Microsoft News AI Agent

AI Studio Final Project

12/6/2025

BREAK THROUGH TECH

Meet Our Team

Anya Liu
Boston University

Ashriya Tuladhar
Umass Boston

Erwin Coq
Umass Lowell

Jerry Liu
Boston University

Sonia Broni
Tufts University

Sophie Lin
Wellesley College

Our AI Studio Mentors

Dr. Francesca Lazzeri
Microsoft

Challenge Advisor

Andrii Zahorodnii
MIT

Teaching Assistant

Chu Huang
BTTAI

Maxime Nguyen
BTTAI

Break Through Tech AI Mentors

Presentation Agenda

- Introduction and Context for News AI Agent
- Data Understanding, Cleaning
- Feature engineering
- Categorization Models: Training, Evaluation, and Comparison
- LLMs: Evaluation and Comparison
- Agentic AI - App Development
- Future Improvements and Learning

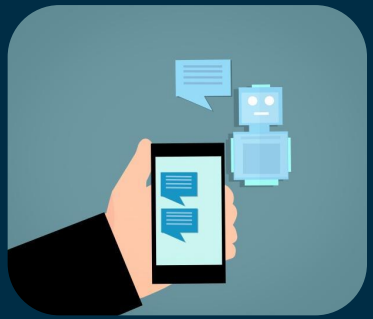
Introduction and Context for News AI Agent

AI Studio Project Overview

- There’s too much information!

- News AI Agent, aims to solve exactly that!
  - Categorization
  - Summarization
  - Q&A / Chatbot

BBC NEWS



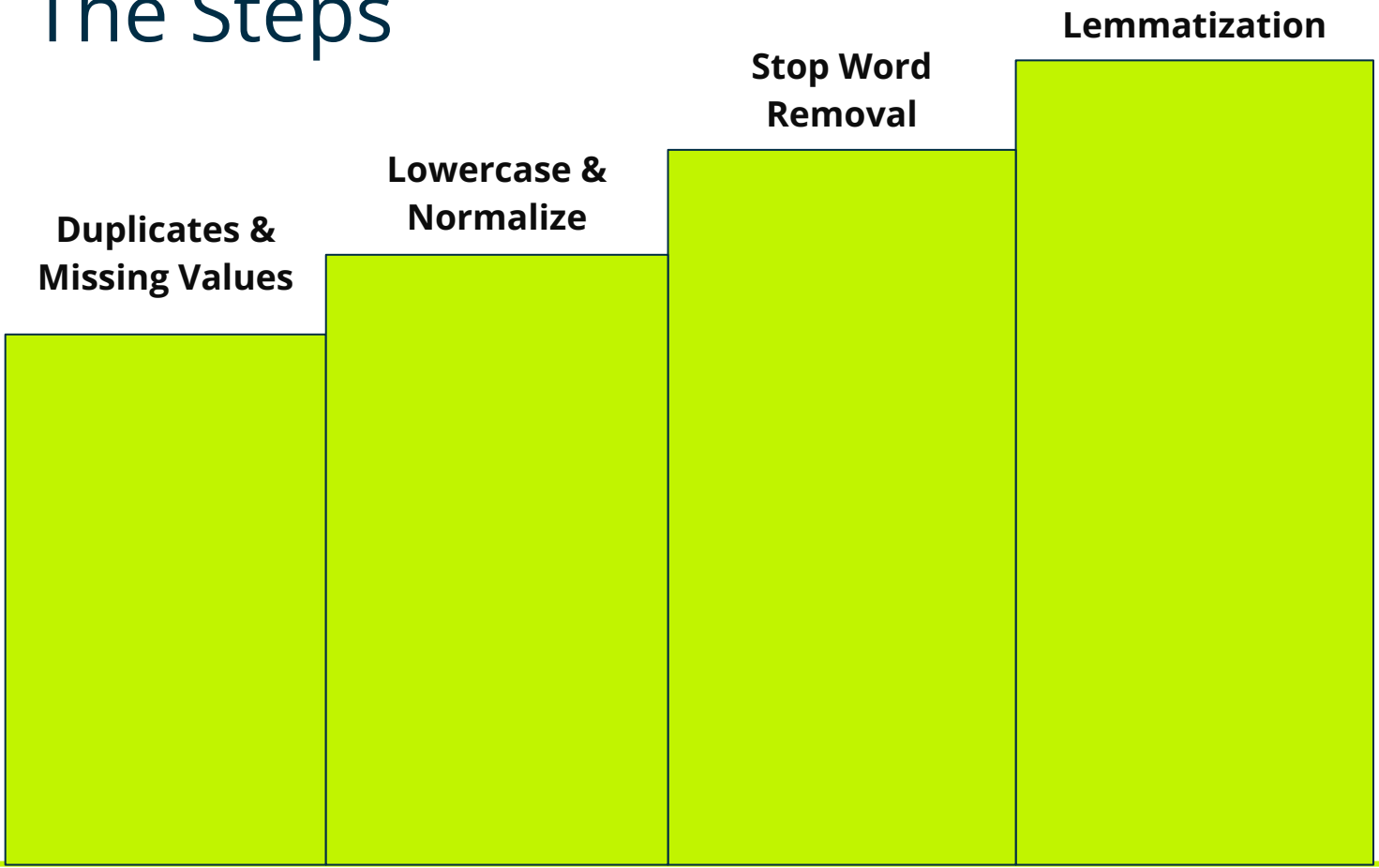
Data: Public domain news articles (BBC-style), processed as text data. Comprised of 2225 articles, each labeled under one of 5 categories: business, entertainment, politics, sport or tech

ML Challenge Type: Supervised classification (topic categorization) + text summarization (NLP) + Generative AI (Chatbot)

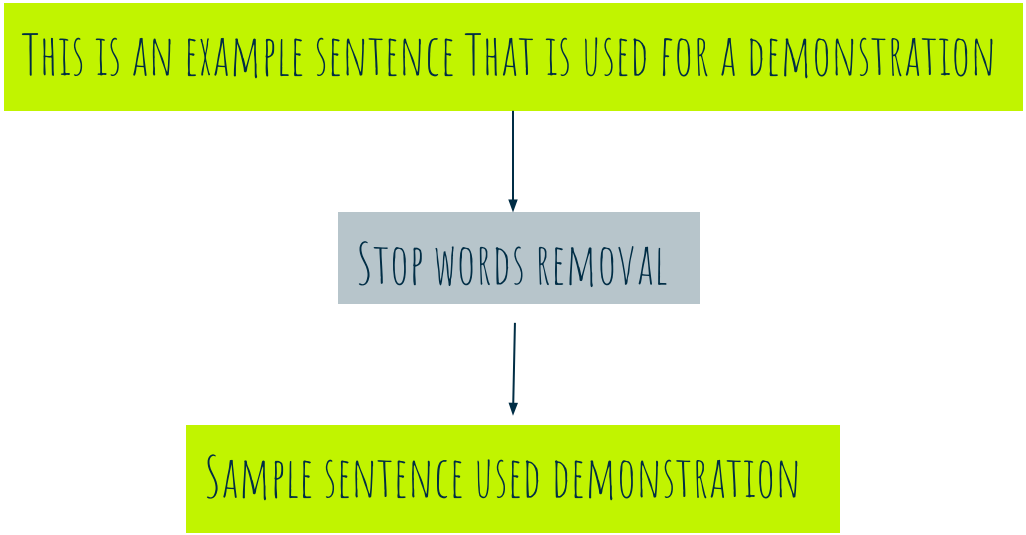
Resources We Leveraged

Data Understanding and Cleaning

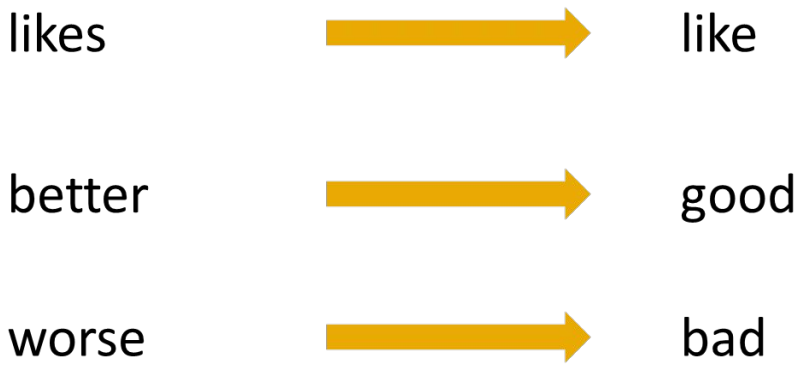
The Steps


Duplicates & Missing Values
Lowercase & Normalize
Stop Word Removal
Lemmatization

Stop Word Removal



Lemmatization



⬇️ Data Size ⬆️ Model Performance

Feature Engineering

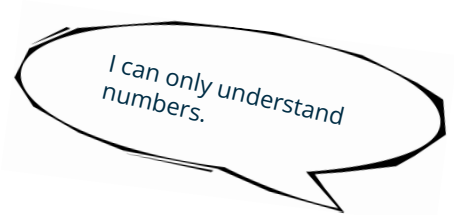
ML Models in our project

1) Categorization model
(requires numerical input,
needs feature engineering)

2) Large Language Model
(can understand human language)

BBC News Data set → raw text data

ArticleId	Text	Category
1833	worldcom ex-boss launches defence lawyers defending fo	business
154	german business confidence slides german business conf	business
1101	bbc poll indicates economic gloom citizens in a majority o	business
1976	lifestyle governs mobile choice faster better or funkier ha	tech
917	enron bosses in $168m payout eighteen former enron dire	business
1582	howard truanted to play snooker conservative leader mic	politics
651	wales silent on grand slam talk rhys williams says wales ar	sport
1797	french honour for director parker british film director sir al	entertainment



News Text data

Feature engineering

Numeric Data

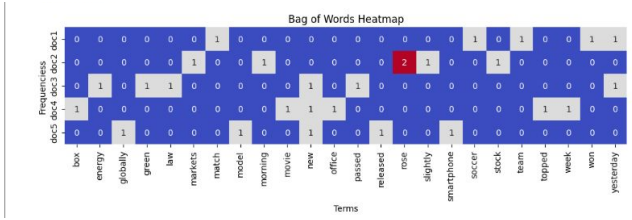
ML Model

1) Bag of words

Text → Frequency of text in document

example:

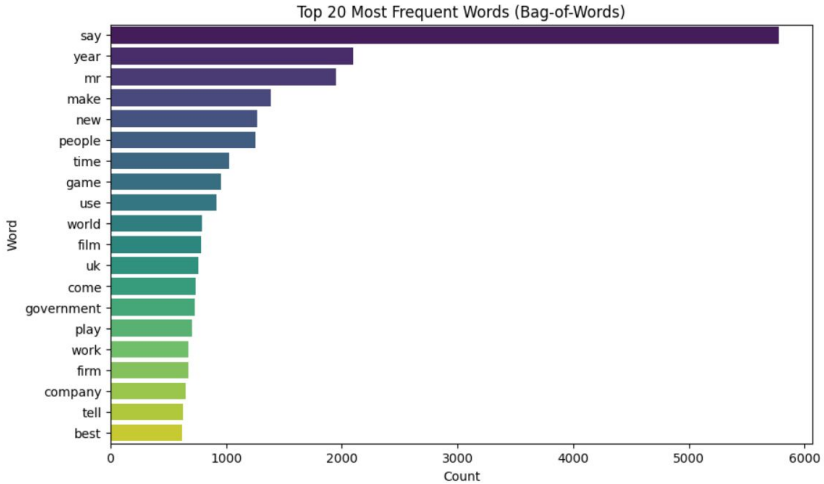
	ArticleId	Text	Category
0	1	The soccer team won their match yesterday.	sport
1	2	Stock markets rose rose slightly this morning.	business
2	3	The new green energy law was passed yesterday.	politics
3	4	A new movie topped the box office this week.	entertainment
4	5	A new smartphone model was released globally.	tech



Call Bag-of-Words Function

```
X_train_bow, X_test_bow, bow_vectorizer = bow(X_train['Text'], X_test['Text'], return_vectorizer=True)
```

Vocabulary size: 18836
Training matrix shape: (1460, 18836)
Testing matrix shape: (735, 18836)



2) TF-IDF

Call TF-IDF Function

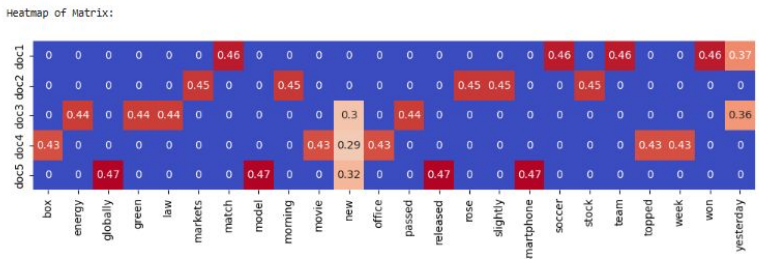
```
X_train_tfidf, X_test_tfidf, vectorizer = tf_idf(X_train, X_test, return_vectorizer=True)
feature_names = vectorizer.get_feature_names_out()
```

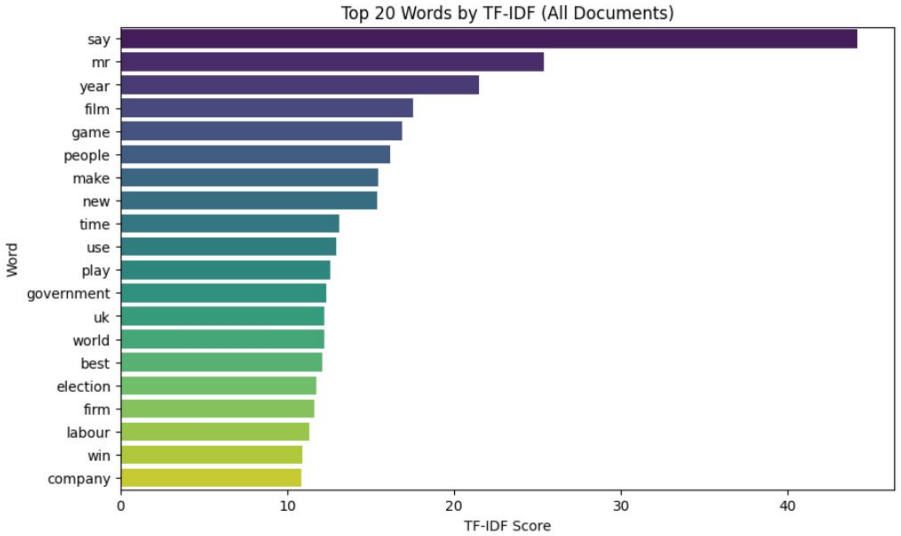
Text → Frequency x IDF scores

[Rare word - high score
and vice versa]

Vocabulary size: 221705
Training matrix shape: (1460, 221705)
Testing matrix shape: (735, 221705)

example:

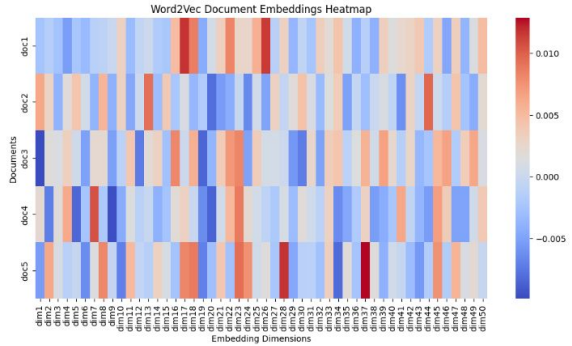




3) Word2Vec

Text → vectors that capture meaning and semantics

example:



Two word2vec models:

1) CBOW (predict the word from the surrounding words)
2) SKIP-GRAM (predict surrounding words from the word)

Call Word2Vec Function

```
X_train_wv, X_test_wv, wv_model = word2vec_embeddings(X_train['Text'], X_test['Text'])
```

```
Training matrix shape: (1460, 100)
Testing matrix shape: (735, 100)
```

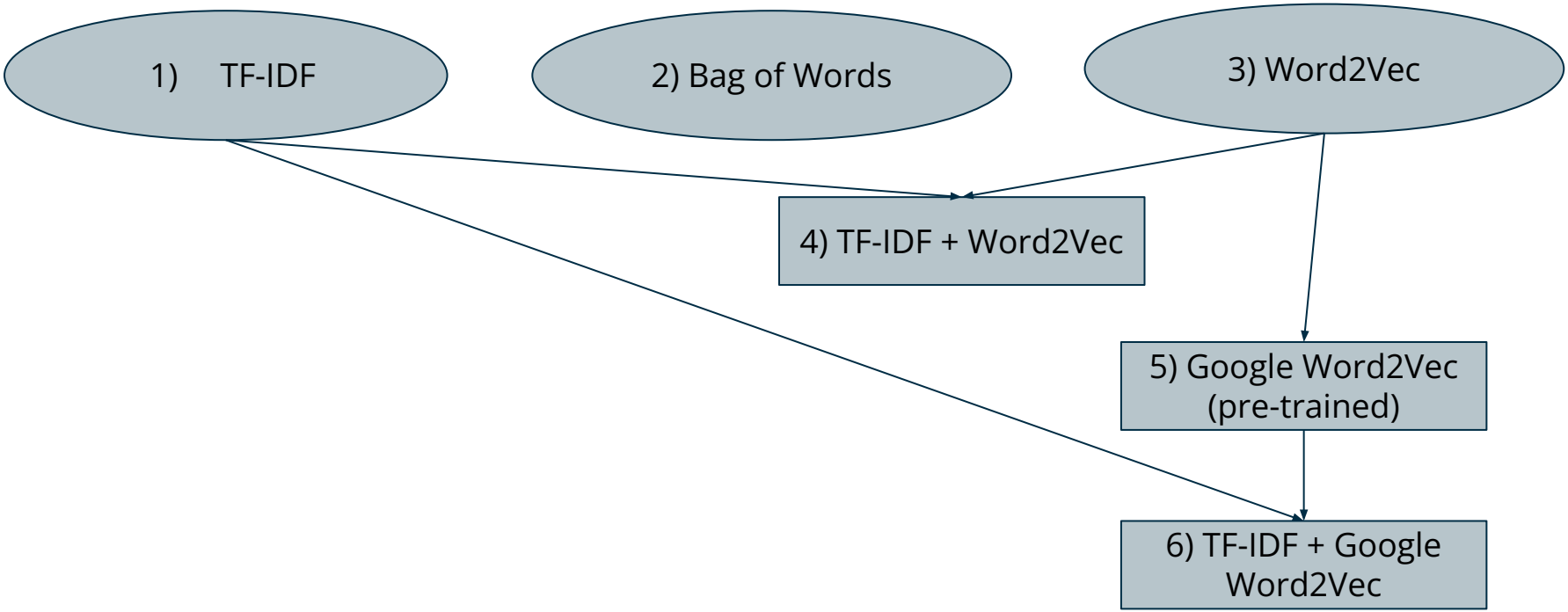
```
# Analogy: "actor" is to "film" as ____ is to "music"
# Returns top 5 words that best fit this analogy
wv_model.wv.most_similar(positive=["actor", "music"], negative=["film"], topn=5)
```

```
[('hip', 0.7542892694473267),
 ('interactive', 0.7435777187347412),
 ('dance', 0.742928683757782),
 ('mtv', 0.733330249786377),
 ('rap', 0.7274619340896606)]
```

```
# Should see "queen" on large data sets
wv_model.wv.most_similar(positive=["king", "woman"], negative=["man"], topn=5)
```

```
[('friend', 0.8636079430580139),
 ('diary', 0.8501811027526855),
 ('cast', 0.8480105996131897),
 ('passion', 0.8426082730293274),
 ('reunite', 0.8407309055328369)]
```

Ensembling Feature -Engineering Techniques:



4) Word2Vec + TF-IDF

```
# Should see "queen" on large data sets
wv_if_model.wv.most_similar(positive=["king", "woman"], negative=["man"], topn=5)
```

```
[('self', 0.8277500867843628),
 ('bpi', 0.8193415999412537),
 ('panel', 0.8163647651672363),
 ('pundit', 0.8142743706703186),
```

5) Google Word2Vec

```
# Explore the pretrained vectors
word = "king"
similar_words = wordvec2_model_loaded.most_similar(word, topn=10)
print(f"Words most similar to '{word}':")
for w, score in similar_words:
    print(f"\t{w}: {score:.4f}")

# king - man + woman = ?
result = wordvec2_model_loaded.most_similar(positive=['king', 'woman'], negative=['man'], topn=1
print("\nking - man + woman =", result)
```

```
Words most similar to 'king':
        kings: 0.7138
        queen: 0.6511
        monarch: 0.6413
        crown_prince: 0.6204
        prince: 0.6160
        sultan: 0.5865
        ruler: 0.5798
        princes: 0.5647
        Prince_Paras: 0.5433
        throne: 0.5422

king - man + woman = [('queen', 0.7118193507194519)]
```

Google Word2Vec pre-trained on about 100 billion words

https://www.kaggle.com/datasets/sugataghosh/google-word2vec

6) Google Word2Vec + TF-IDF

```
# testing the function

# Create and fit a TF-IDF vectorizer
tfidf_model_temp = TfidfVectorizer()
tfidf_model_temp.fit(df_clean['cleaned_txt'])


print("Testing the function")
sample_text = df_clean['cleaned_txt'].iloc[0]
print(sample_text[:200])  # show the first 200 characters to confirm which article we're testing

# Test the google_w2v_tfidf function
sample_vector = google_w2v_tfidf(sample_text, tfidf_model_temp, wordvec2_model_loaded)

# Check the output
print("Vector shape:", sample_vector.shape)
print("First 10 values:", sample_vector[:10])
```

```
Testing the function
worldcom ex-boss launch defence lawyer defend former worldcom chief bernie ebbers battery fraud charge call company whistleblower first witness
Vector shape: (300,)
First 10 values: [-0.03358119  0.02605228  0.00470108  0.00632206 -0.03344466  0.01062823
  0.04462273 -0.0800718   0.10644963  0.03359389]
```

Categorization: Traditional ML Models

Training, Evaluation, and Comparison

Model Overview

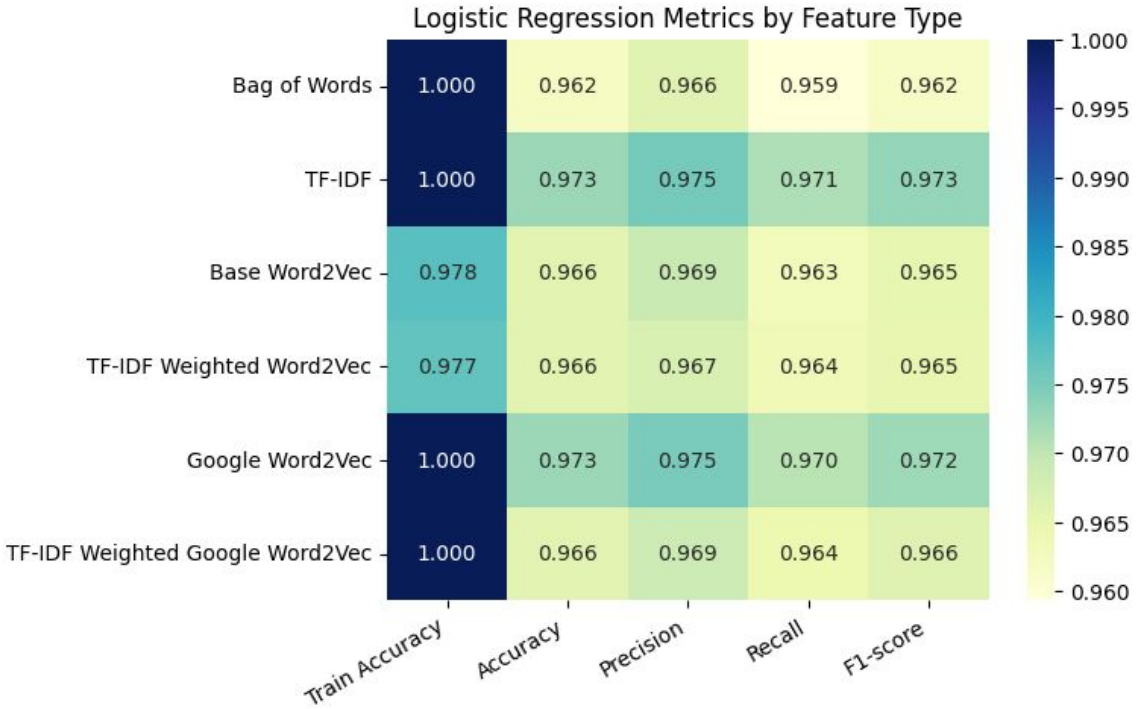
Model Name	Trained On	Key Hyper-Parameters	Notes
Logistic Regression	TF-IDF Bag-of-Words Word2Vec TF-IDF + Word2Vec Google Word2Vec (pretrained) TF-IDF + Google Word2Vec	C = 0.1 StandardScaler used	Regularized, good for all feature types
Naive Bayes	Bag of Words TF-IDF	alpha = 0.1 fit_prior = true	Works best with discrete counts, not embeddings
Random Forest	TF-IDF Bag-of-Words Word2Vec TF-IDF + Word2Vec Google Word2Vec (pretrained) TF-IDF + Google Word2Vec	n_estimators=200, max_features='sqrt', min_samples_leaf=3, min_samples_split=10	Ensemble of trees, handles high-dim features

Logistic Regression Evaluation

Best feature type:

TF-IDF

- Train Accuracy: 1.0000
- Test Accuracy: 0.9726
- Test Precision: 0.9753
- Test Recall: 0.9714
- Test F1-score: 0.9731

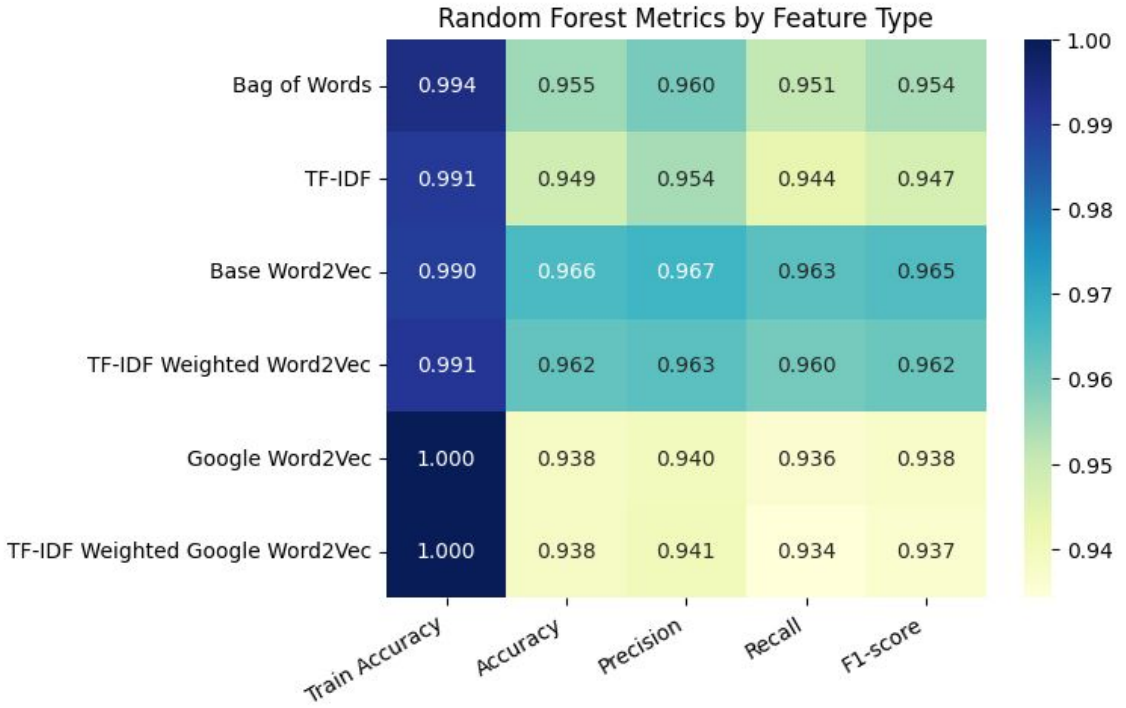


Random Forest Evaluation

Best feature type:

Base Word2Vec

- Train Accuracy: 0.9897
- Test Accuracy: 0.9658
- Test Precision: 0.9673
- Test Recall: 0.9632
- Test F1-score: 0.9649

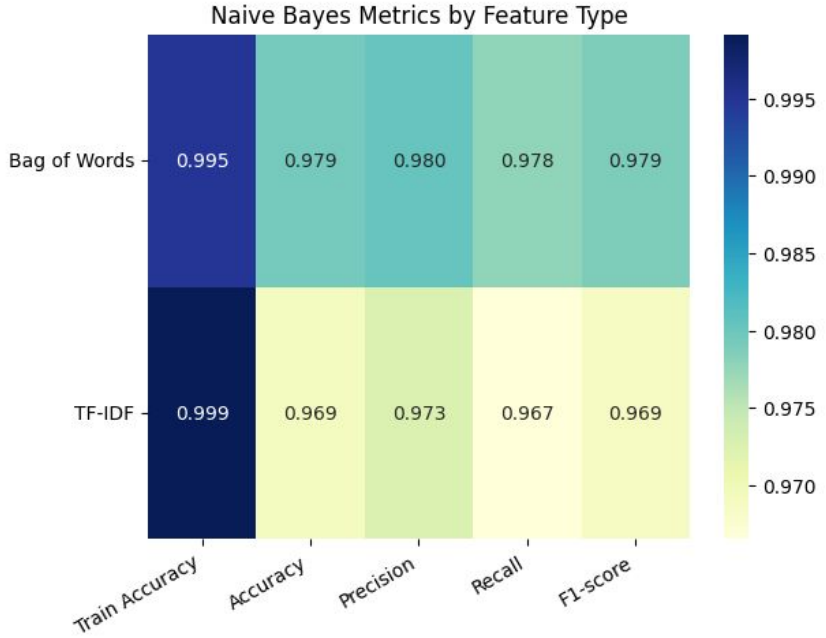


Naive Bayes Evaluation

Best feature type:

Bag-of-Words

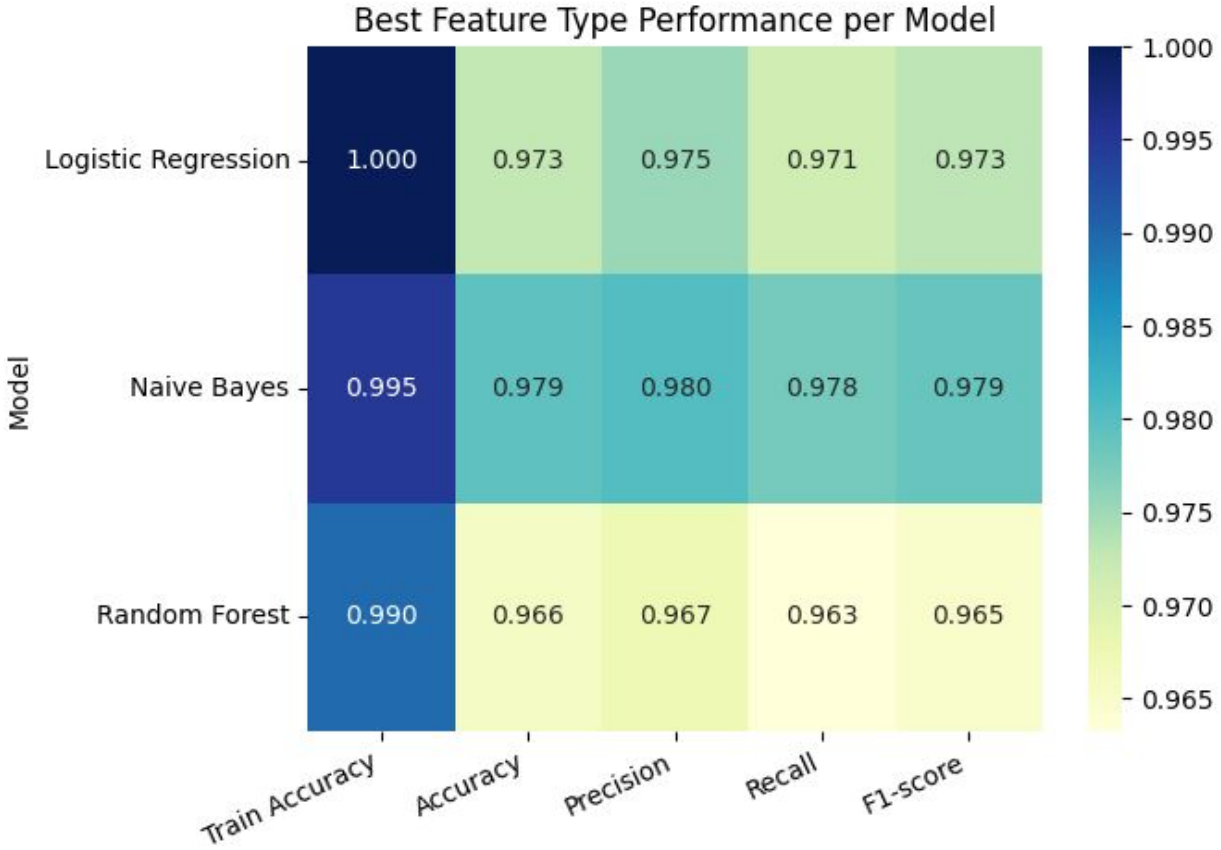
- Train Accuracy: 0.9949
- Test Accuracy: 0.9795
- Precision: 0.9803
- Recall: 0.9778
- F1-score: 0.9788



Finding the Best Model

Naive Bayes with Bag-of-Words achieved the highest overall performance across all key metrics:

- train accuracy of 0.9949,
- test accuracy of 0.9795,
- precision of 0.9803,
- recall of 0.9778, and
- F1-score of 0.9788.



LLMs: Evaluation and Comparison

BART, T5, Open LLaMA, GPT

Evaluating Summarization Models:
What Works for Our Agent?

Goal

- Find a summarization model that is **accurate, fast, instruction-following, and cost-efficient**
- Explore whether models can also handle **Q&A**

Golden Dataset (Our Benchmark)

- Article → human-written summaries (multiple summary types)
- Question → human-written answers for Q&A
- Provided the ground truth for comparing model outputs

Evaluation Method

- Generated model summaries/answers for every article
- Compared predictions to golden references using:
  - BLEU & ROUGE (automated metrics)
  - Human scoring (1– 3) for semantic quality

Summarization – BART

What Worked

CNN → coherent multi-sentence summaries

XSum → clean one-sentence headlines

Handles long articles via chunking

Limitations

Does **not follow instructions**

- CNN always outputs multiple sentences
- XSum always outputs one sentence

Cannot perform **Q&A** (always outputs a summary)

Golden Summary	CNN Summary	XSum Summary
SoFi's stock dropped after they announced a new $1.5 billion share sale, which surprised a lot of people. They want more cash to keep growing the business and bring back things like crypto trading. The stock is still doing well this year, but it looks like they raised money now because the price was high.	SoFi Technologies, a fintech-turned-bank, announced a $1.5 billion share sale on Thursday. The company plans to invest in its existing businesses, relaunch crypto trading, and expand its product offerings. SoFi's second capital raise in two quarters wasn't widely expected.	SoFi's stock is down about 7% after the fintech firm raised $1.5bn in a share sale.

Avg Metrics:	BLEU	0.069	BLEU	0.0041
	Rouge-L	0.2993	Rouge-L	0.1879
	Human Evaluation	1.7 / 3	Human Evaluation	1.5 / 3

Summarization – T5

Best Balance of **Speed, Cost, and Instruction Following**

Why FLAN-T5-Small?

- Instruction-tuned → understands "1 sentence vs 2–3 sentences," causal summaries
- Fast, lightweight, and cheap to run locally (no API cost)
- Integrated well with chunking + merging pipeline

Limitations:

- Limited Q&A ability
- Not conversational

Golden Summary	Model Summary
Parent DaimlerChrysler experienced a slump in profits in 2004 due to Mercedes' poor performance, though it expects to see improvements in profitability through major investments.	Mercedes-Benz has reported a fall in profits at its parent company.

Avg Summarization Metrics:

BLEU	0.1293
Rouge-L	0.1814
Human Evaluation	2.5/3

Question	Golden Answer	Model Answer
What are the 7 benefits mentioned?	The seven benefits are: 1. Exercise… 2. It lowers your risk for many health…	Get more active throughout your day.

Open LLaMA

What We Tried:

- Ran OpenLLaMA 3B + TinyLLaMA locally
- **Goal:** test open-source instruction-tuned small models

Challenges:

- Very slow latency on available compute
- Summaries were incoherent
- Poor instruction-following
- Unable to fine-tune on our Golden Dataset

Prompt: Give a one-sentence summary of the article.
Summary:
tory expert denies defeatism the conservatives campaign director has denied a report claiming he warned michael howard the party could not win the next general election. the times on monday said australian lynton crosby told the party leader to focus on trying to increase the tories commons presence by 25 to 30 seats. but mr crosby said in a statement: i have never had any such conversation... and i do not hold that view. mr howard later added there was not one iota of truth in the report. the strategist helped australia s pm john howard win four elections. mr howard appointed mr crosby as his elections chief last october. mr crosby s statement said: the conservative party has been making an impact on the issues of lower tax and controlled immigration over the past week. it addeed, however ,that these are just two examples out many others where we can make progress for our country if only they were given more time .

Summarization + Chatbot – GPT

Why GPT?

- Could interpret complex user intent
- Handles **analytical questions**, comparisons, causal summary
- Has **natural conversational ability**

When Do We Use It?

Used for analytical Q&A, multi-turn conversation
Cost-aware fallback, not the primary model

OpenAI API Call

Prompt:

> Instruction: "You are given an article as context. Read and analyze the article carefully, then use its information to provide a specific, well-supported, and concise answer to the following question. "
> Answer the question: {question}\n\n

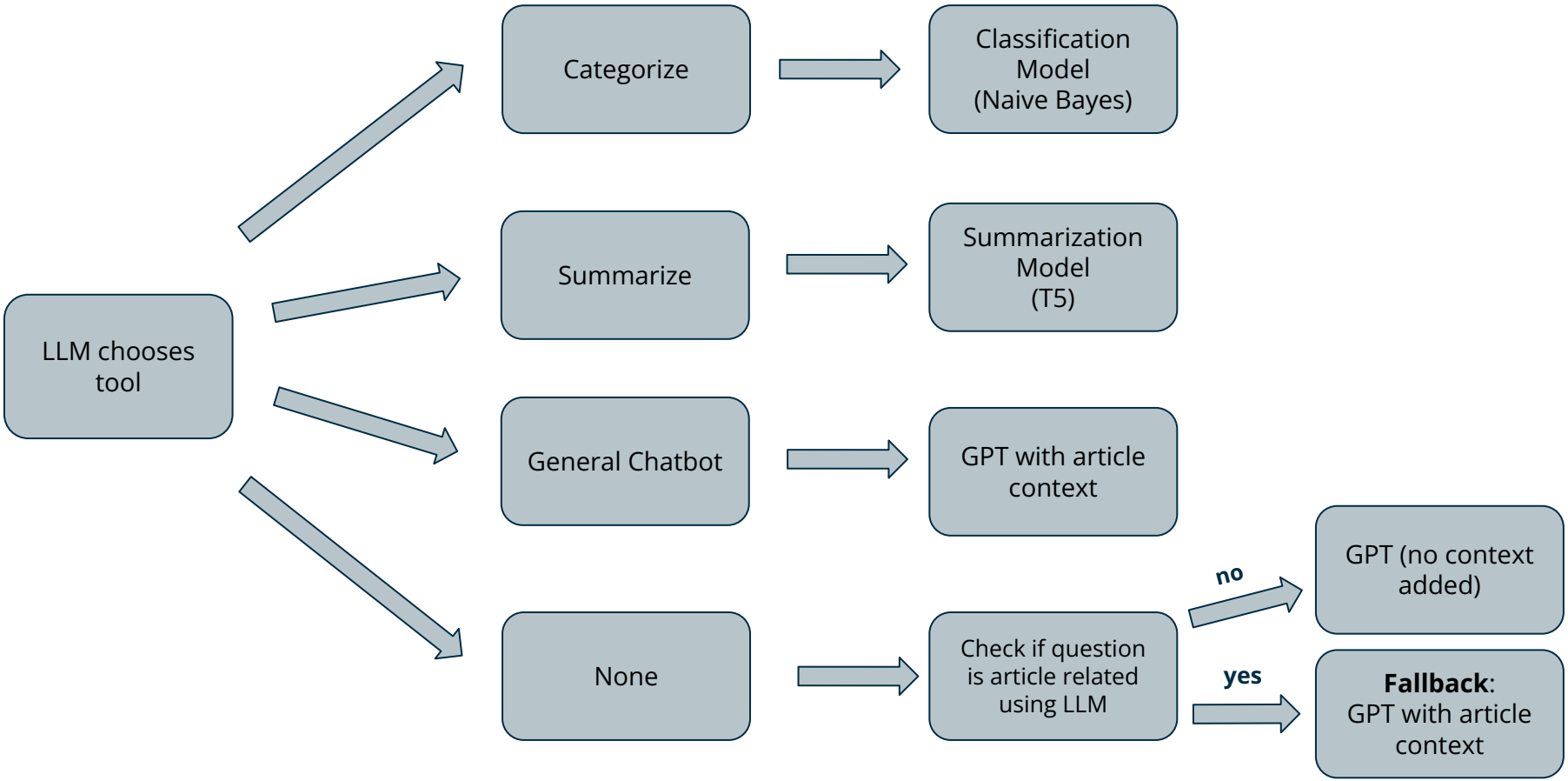
Average Metrics

GPT 4-o:	0.0549
Human Evaluation	2.5 / 3
GPT 4-o mini:	0.0356
Human Evaluation:	1 / 3

Transforming it into an AI Agent

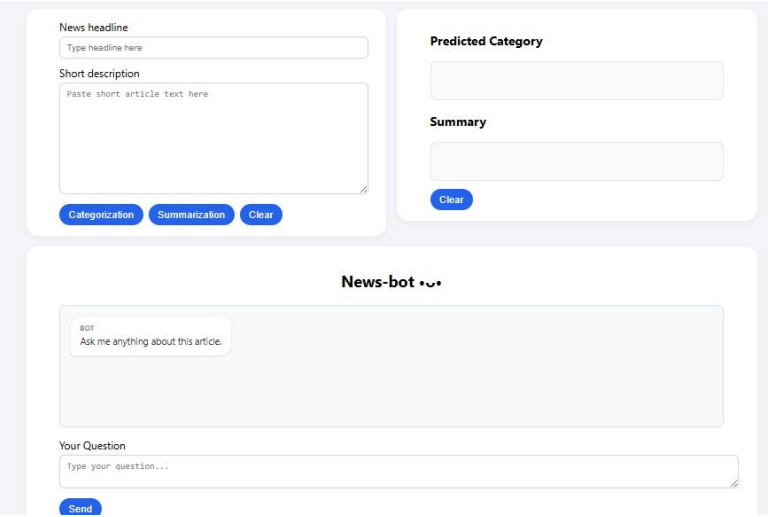
AI Agent

- Let LLM decide which tools best fit a user's request
    - More flexibility
        - user does not need to manually call each model
    - Allows them to focus on their request and letting the LLM autonomously choose the optimal way of executing
- Implementations
    - Inserting the model functions directly into the LLM
        - too slow, parsing long article twice
    - Two Part Implementation with fallback
        - Improved prompts with feedback, leading to more accuracy with LLM choice of tools


LLM chooses tool
Categorize
Classification Model (Naive Bayes)
Summarize
Summarization Model (T5)
General Chatbot
GPT with article context
None
Check if question is article related using LLM
no
GPT (no context added)
yes
Fallback: GPT with article context

Bringing it All Together
App Development

User Interface



Tech Stack

Front End

Backend	Python, Flask, scikit learn, OpenAI, Word2Vec, load_dotenv, pandas, NumPy, etc

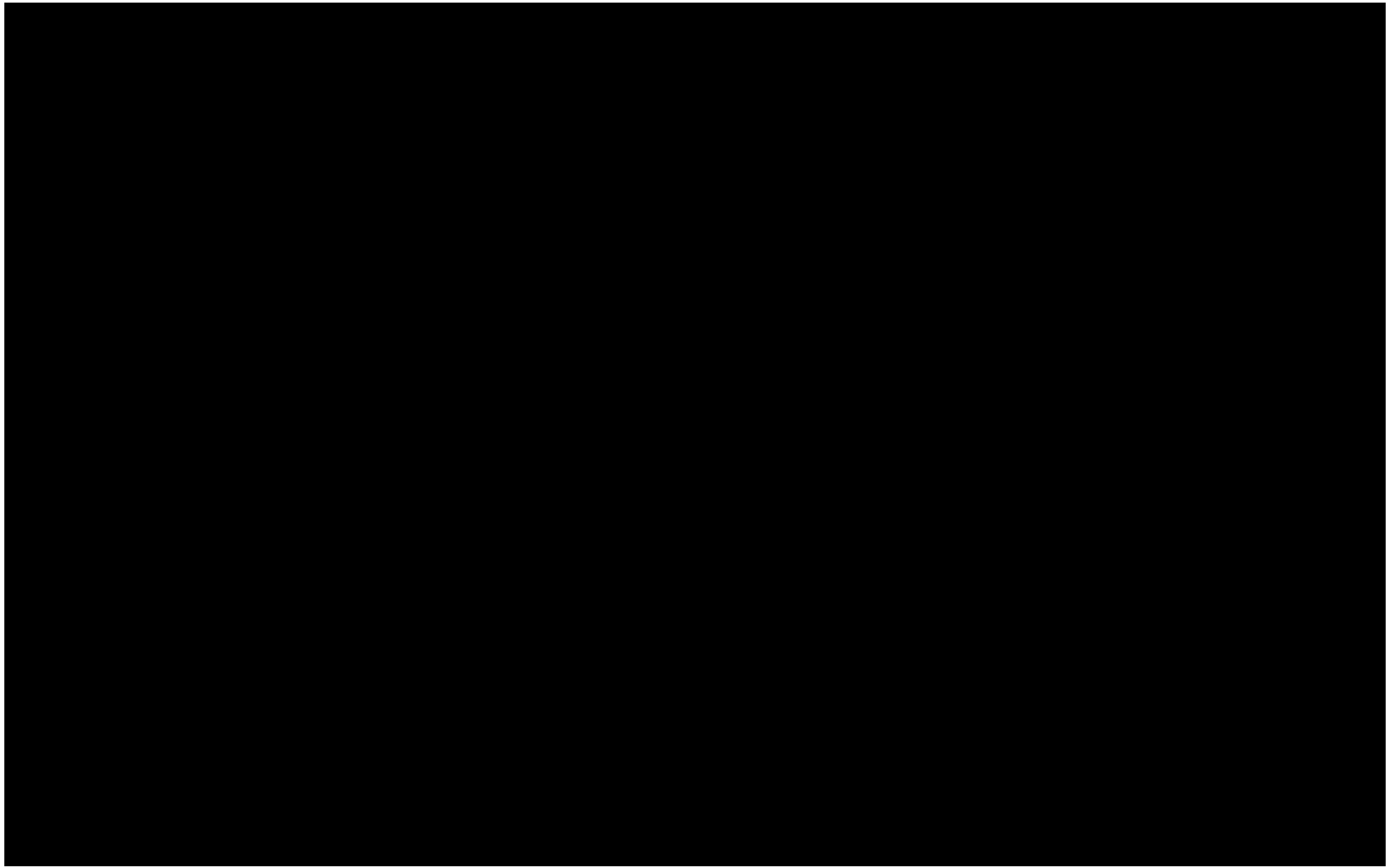
Workflow:

User → React UI → Flask API → ML Models → Flask Response → React Output

Three Teams of 2 each owned the features related to the following models:

1) Categorization Feature
2) Summarization Feature
3) Chatbot Feature

Demo



Next Steps

Potential Next Steps

- Improve AI Agent
  - Guard Rails
  - Lower Latency
- Model Deployment
  - Deploy the classification behind a clean production-ready API endpoint
  - Handle updates and monitoring
- Containerization
  - Docker
  - Ensures consistency across testing and production

Potential Next Steps (continued)

- User Interface
  - More intuitive, easier to use, and visually appealing
    - User doesn't need to copy and paste article, use online model
- RAG (Retrieval-Augmented Generation)
  - Reduces hallucinations and improves accuracy by grounding LLM responses in given article
  - Reduce speed because divides article into chunks

Thank you!

Questions?