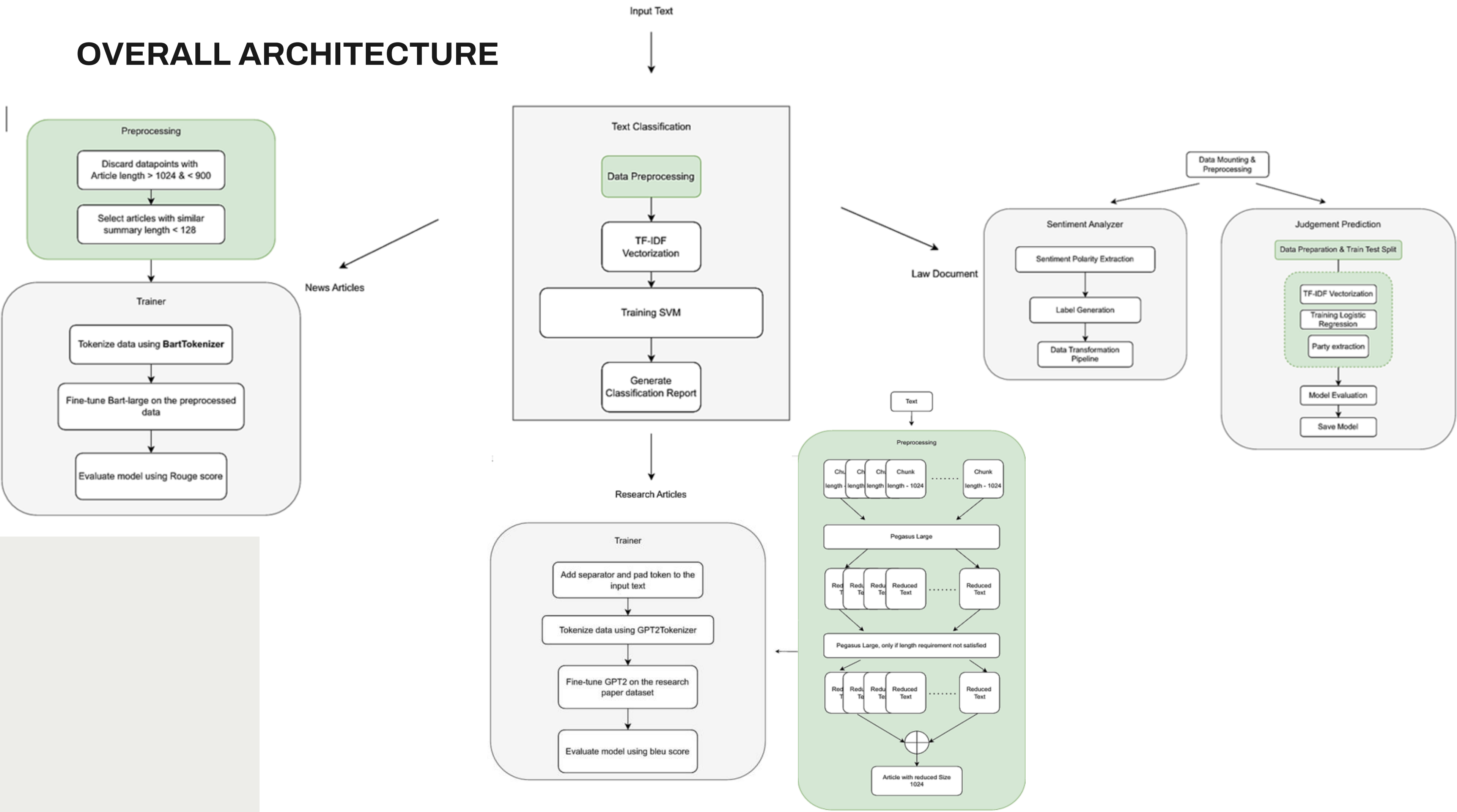
Automated Textual Summarization

By Heera, Sharvita, Shanmukha

Introduction

- **Problem:** Difficulty in processing large texts
- **Solution:** Automated Textual Summarization (ATS) system.
- **Goal:** Generating summaries and analyzing large documents concise summaries using various machine learning models.
- **Target Documents:** news articles, research papers, law papers

OVERALL ARCHITECTURE



Evaluation Metrics

BLEU Score

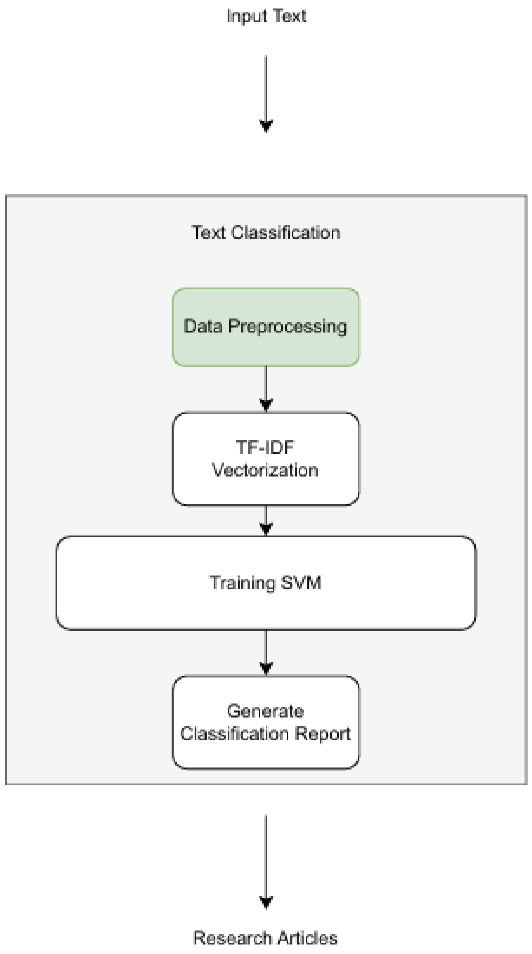
- ❑ Assess readability, accuracy, coverage, and fluency.
- ❑ Scoring system (1 to 5) to derive overall effectiveness.

ROGUE Score:

- ❑ Measures overlap between machine-generated and human-generated summaries.
- ❑ Indicates the effectiveness of content retention.

F1 Score:

- ❑ Metric used to evaluate the performance of a classification model, particularly in binary classification tasks, by combining precision and recall into a single value



MODULE 1:
TEXT CLASSIFICATION

Data Preprocessing for text classification

```
def GetDataset(self):
    news_articles = pd.read_csv("../data/CNN_Reduced_128_Full.csv")
    research_papers = pd.read_csv("../data/arXiv_Final.csv")
    law_documents = pd.read_csv("../data/LawDocuments.csv")

    research_papers.drop(['Unnamed: 0', 'abstract'], axis=1, inplace=True)
    news_articles.drop(['Unnamed: 0','summary'], axis=1, inplace=True)
    law_documents.drop('Unnamed: 0', axis=1, inplace=True)
    law_documents = law_documents.dropna()

    news_data = news_articles.sample(n=len(research_papers), random_state=42)
    research_data = research_papers.sample(n=len(research_papers), random_state=42)
    law_data = law_documents.sample(n=len(research_papers), random_state=42)

    news_data['label'] = 'News Article'
    research_data['label'] = 'Research Paper'
    law_data['label'] = 'Law Document'

    news_data.rename(columns={'article': 'text'}, inplace=True)
    research_data.rename(columns={'reduced_articles': 'text'}, inplace=True)
    law_data.rename(columns={'case_text': 'text'}, inplace=True)

    combined_data = pd.concat([news_data, research_data, law_data])
    self.dataset = combined_data.sample(n=len(combined_data), random_state=42)

def EncodeLabels(self):
    self.dataset['label'] = self.dataset['label'].map({'News Article': 0, 'Research
        Paper': 1, 'Law Document': 2})
```

Load, clean, and prepare the dataset.

TF-IDF Identification

```
def Vectorize_TFIDF(self):
    X_train_tfidf = self.vectorizer.fit_transform(self.trainTestSplit[0])
    X_test_tfidf = self.vectorizer.transform(self.trainTestSplit[1])
    self.inputTrainTestVectors = X_train_tfidf, X_test_tfidf
```

Training SVM

```
def TrainSVM(self):
    svm_model = SVC(kernel='linear', random_state=42)
    print("Training SVM model")
    svm_model.fit(self.inputTrainTestVectors[0], self.trainTestSplit[2])
    self.model = svm_model
```

Generating Classification Report

```
def EvaluateModel(self):
    y_pred = self.model.predict(self.inputTrainTestVectors[1])
    print("Clasification Report:")
    print(classification_report(self.trainTestSplit[3], y_pred, target_names=['News
        Article', 'Research Paper', 'Law Document']))
```

Results

	precision	recall	f1-score	support
News Article	0.99	1.00	0.99	420
Research Paper	0.99	1.00	0.99	420
Law Document	1.00	0.99	0.99	416
accuracy			0.99	1256
macro avg	0.99	0.99	0.99	1256
weighted avg	0.99	0.99	0.99	1256

Feature vector size: 100

	precision	recall	f1-score	support
News Article	0.99	0.98	0.98	420
Research Paper	0.95	0.98	0.96	420
Law Document	0.98	0.95	0.96	416
accuracy			0.97	1256
macro avg	0.97	0.97	0.97	1256
weighted avg	0.97	0.97	0.97	1256

Feature vector size: 50

	precision	recall	f1-score	support
News Article	1.00	1.00	1.00	420
Research Paper	1.00	1.00	1.00	420
Law Document	1.00	1.00	1.00	416
accuracy			1.00	1256
macro avg	1.00	1.00	1.00	1256
weighted avg	1.00	1.00	1.00	1256

Feature vector size: 500

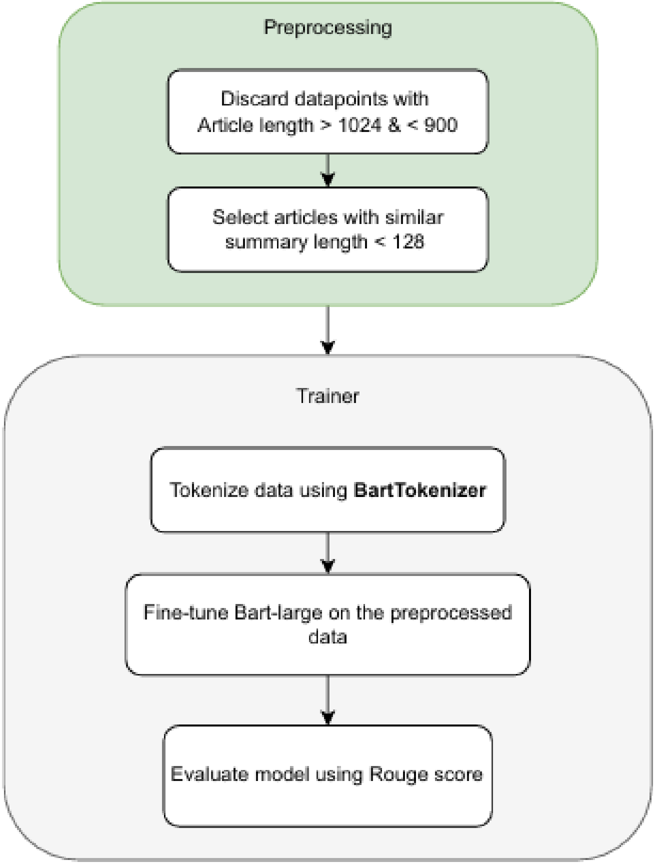
Models Used and Why?

SVM (Support Vector Machine) for Classification

- ❏ Handles high-dimensional, sparse text data effectively , robust to overfitting by focusing on key data points (support vectors), efficient and simpler to train on medium-sized datasets

Why Not Other Models:

- ❏ Naive Bayes: Assumes feature independence, which isn’t realistic for text.
- ❏ Logistic Regression: Struggles with complex boundaries in high-dimensional data.
- ❏ Neural Networks (BERT/LSTM): Require large datasets and high computational resources.



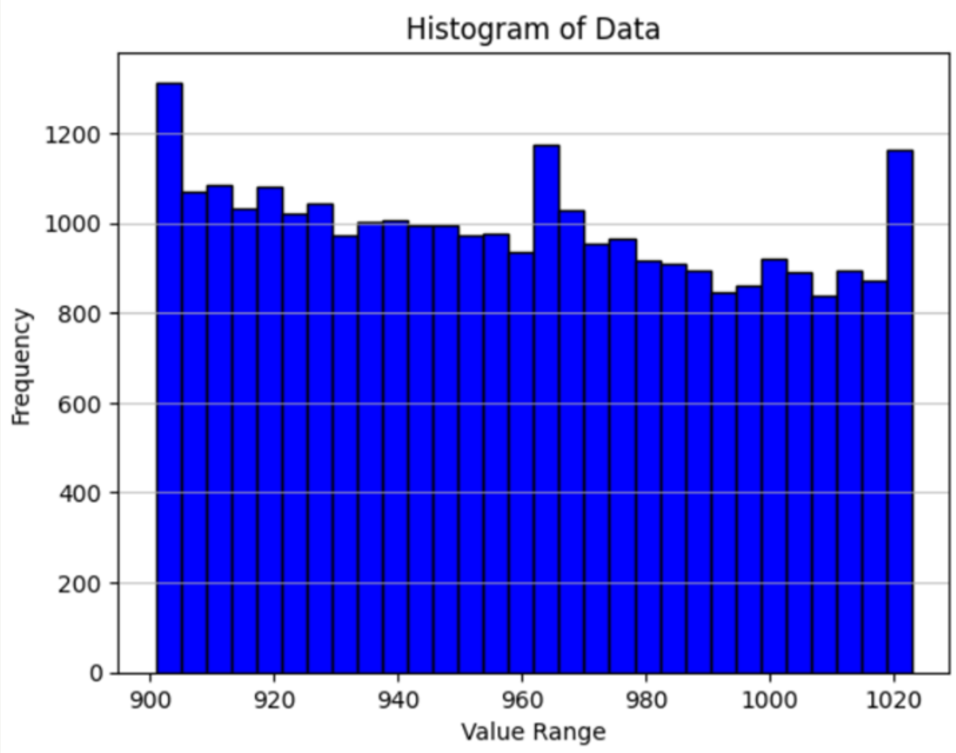
MODULE 2: NEWS ARTICLES

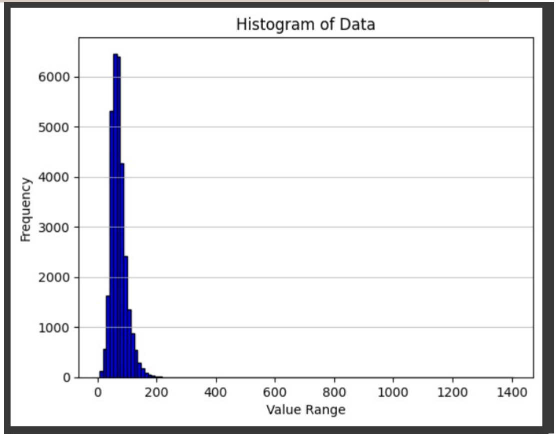
PreProcessing: select articles with similar length

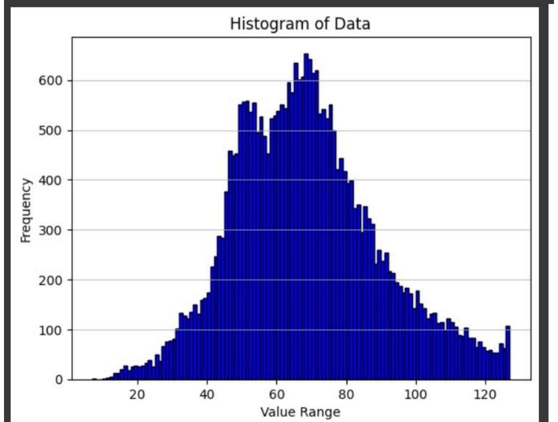
Articles with length outside 900–1024 tokens are excluded to ensure uniform input size for BART, as its architecture is optimized for handling inputs of specific lengths efficiently.

```
class PreprocessCNN:

    def __init__(self):
        self.dataset = None
        self.trainValTest = None
        self.tokenizer = BartTokenizer.from_pretrained("facebook/bart-large")
        self.model = BartForConditionalGeneration.from_pretrained("facebook/bart-large")

    def IsDataValid(self, dataPoint):
        token = self.tokenizer(dataPoint['article'], return_tensors="pt")['input_ids']
        size = len(token.squeeze())
        return size > 900 and size < 1024

    def LoadDataset(self):
        dataset = load_dataset("cnn_dailymail", "3.0.0")
        train_data = dataset["train"].shuffle(seed=42)
        val_data = dataset['validation'].shuffle(seed=42)
        test_data = dataset['test'].shuffle(seed=42)

        self.trainValTest = train_data, val_data, test_data

    def IsSummaryLengthGreaterThan128(self, row):
        token = self.tokenizer(row['summary'], return_tensors="pt")['input_ids']
        return len(token.squeeze()) < 128
```

Histogram of article size







Trainer: Tokenize data using BART Tokenizer

Converts articles and summaries into tokenized formats compatible with the BART model, preparing them for training

```
def tokenize_function(self, batch):
    inputs = self.tokenizer(
        batch['article'],
        max_length=1024,
        truncation=True,
        padding="max_length"
    )
    labels = self.tokenizer(
        batch['summary'],
        max_length=128,
        truncation=True,
        padding="max_length"
    )
    inputs['labels'] = labels['input_ids']
    return inputs
```

Trainer: Fine-tune BART-large on the preprocessed data

Adjusts the pre-trained BART model to perform the specific task of summarization, improving its performance on the target dataset.

```
def Train(self):

    train_dataset = SummarizationDataset(self.trainValEncodings[0])
    val_dataset = SummarizationDataset(self.trainValEncodings[1])

    train_loader = DataLoader(train_dataset, batch_size=8, shuffle=True)
    val_loader = DataLoader(val_dataset, batch_size=8)

    self.trainLoader = train_loader
    self.valLoader = val_loader

    optimizer = AdamW(self.model.parameters(), lr=5e-5)

    self.model.train()
    for epoch in range(5):
        loop = tqdm(train_loader, leave=True)
        for batch in loop:
            batch = {k: v.to(self.device) for k, v in batch.items()}

            outputs = self.model(**batch)
            loss = outputs.loss

            optimizer.zero_grad()
            loss.backward()
            optimizer.step()

            loop.set_description(f"Epoch {epoch}")
            loop.set_postfix(loss=loss.item())
```

ROUGE Evaluation

ROUGE metrics assess the overlap between generated summaries and reference summaries, providing an objective measure of summarization quality.

```
from rouge_score import rouge_scorer
import numpy as np

# Initialize ROUGE scorer
scorer = rouge_scorer.RougeScorer(['rouge1', 'rouge2', 'rougeL'], use_stemmer=True)

# Calculate ROUGE for each generated summary
rouge1_scores, rouge2_scores, rougeL_scores = [], [], []

for gen, ref in zip(generated_summaries, ground_truths):
    scores = scorer.score(gen, ref)
    rouge1_scores.append(scores['rouge1'].fmeasure)
    rouge2_scores.append(scores['rouge2'].fmeasure)
    rougeL_scores.append(scores['rougeL'].fmeasure)

# Compute average ROUGE scores
avg_rouge1 = np.mean(rouge1_scores)
avg_rouge2 = np.mean(rouge2_scores)
avg_rougeL = np.mean(rougeL_scores)

print(f"ROUGE-1: {avg_rouge1:.4f}")
print(f"ROUGE-2: {avg_rouge2:.4f}")
print(f"ROUGE-L: {avg_rougeL:.4f}")
```

```
ROUGE-1: 0.3809
ROUGE-2: 0.1590
ROUGE-L: 0.2560
```

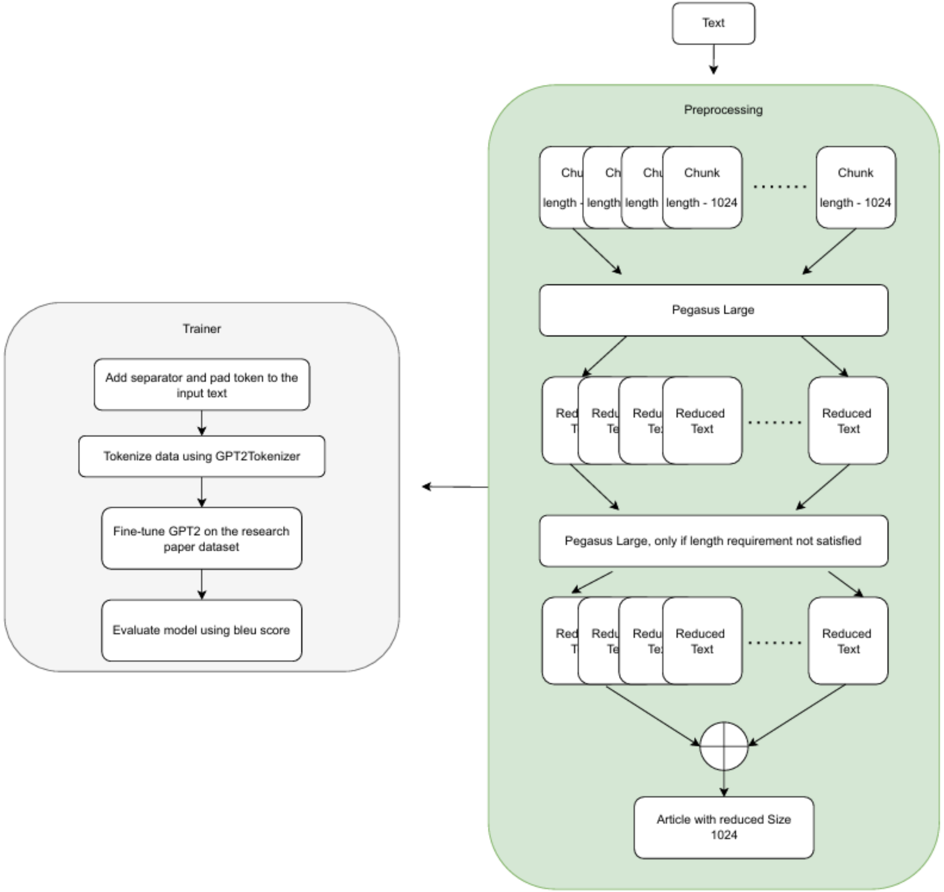
Models Used and Why?

BART-Large for News Articles

DEF: Hybrid model for extractive and abstractive tasks; reconstructs text effectively.

WHY: BART (Bidirectional and Auto-Regressive Transformers) combines the strengths of encoder-decoder models, The encoder processes the input comprehensively, while the decoder generates summaries with coherence and fluency.

T5 NOT USED: T5 only took input size of 512 which is too less and we were not getting good evaluation scores. Bart allows you to use 1025 input size and hence the summaries were more accurate and evaluation scores were higher.



MODULE 3: RESEARCH ARTICLES

Data Preprocessing

```
def SkipRow(self, text):
    input = self.tokenizer(text, return_tensors="pt")
    if len(input['input_ids'][0]) > 6140:
        return True
    return False

def Preprocess(self):
    self.model.eval()
    arXiv_Reduced = {"reduced_articles": [],
                    "article_id": [],
                    "abstract":[]}

    for i in range(0,len(self.arXivdF["article"])):

        if self.SkipRow(self.arXivdF["article"][i]):
            print(f"Skipping row: {i}")
            continue

        arXiv_Reduced["reduced_articles"].append( self.ReduceTextLength(self.arXivdF["article"][i]))
        arXiv_Reduced["article_id"].append(i)
        arXiv_Reduced["abstract"].append(self.arXivdF["abstract"][i])

        print(f"{i} row processed")


        if len(arXiv_Reduced['reduced_articles']) % 50 == 0:
            reduced_df = pd.DataFrame(arXiv_Reduced)
            reduced_df.to_csv(f"/content/drive/MyDrive/ATS/data/arXiv_{i}_ReducedText.csv", index=False)
```

Data Preprocessing

```
def ReduceTextLength(self, text):
    # Tokenize Text
    with torch.inference_mode():
        # model.to("cuda")
        inputs = self.tokenizer(text, return_tensors="pt")

        i = 0
        chucks = []
        while i*1024 <= len(inputs["input_ids"][0]):
            chunk = inputs["input_ids"][0, i*1024:(i+1)*1024 ]
            chucks.append(torch.unsqueeze(chunk, dim=0))
            i = i+1

        summarized_text = ''
        sum_len = 0
        for chunk in chucks:
            # Generate a summary
            # chunk = chunk.to("cuda")
            summary_ids = self.model.generate(
                chunk,
                max_length=256,
                min_length=200,
                length_penalty=1.0,
                num_beams=4,
                early_stopping=True
            )
            sum_len += len(summary_ids[0])

            # Decode the summary
            summary = self.tokenizer.decode(summary_ids[0], skip_special_tokens=True)
            summarized_text += summary
        print(f"Summary Token length {sum_len}")
        return summarized_text
```

```python
def Train(self):
    training_args = TrainingArguments(
        output_dir="/content/drive/MyDrive/ATS/results",
        evaluation_strategy= IntervalStrategy.STEPS,  # Evaluate at the end of each epoch
        eval_steps = 50,
        learning_rate=5e-5,
        per_device_train_batch_size=2,
        per_device_eval_batch_size=2,
        num_train_epochs=10,
        fp16=True,
        # logging_dir="./logs",
        report_to="none",
        metric_for_best_model = 'eval_loss',
        load_best_model_at_end=True,
        save_safetensors=False
    )

    self.trainer = Trainer(
        model=self.model,
        args=training_args,
        train_dataset=self.trainVal[0],
        eval_dataset=self.trainVal[0],
        tokenizer=self.tokenizer,
        callbacks=[EarlyStoppingCallback(early_stopping_patience=3)]
    )
    self.trainer.train()

def SaveModel(self):
    self.trainer.save_model("../model/ResearchPaperModel")
    self.tokenizer.save_pretrained("../model/ResearchPaperModel")
```

Step	Training Loss	Validation Loss
50	No log	3.197822
100	No log	2.847369
150	No log	2.429829
200	No log	2.144670
250	No log	2.031564
300	No log	2.000165
350	No log	1.987154
400	No log	1.973409
450	No log	1.965023
500	2.387300	1.957156
550	2.387300	1.952332
600	2.387300	1.943125
650	2.387300	1.938780
700	2.387300	1.931382
750	2.387300	1.928047
800	2.387300	1.923416
850	2.387300	1.918845
900	2.387300	1.918777

Evaluations

```
average_bleu = calculate_bleu(predictions, references)
print("Average BLEU Score:", average_bleu)  # **Final BLEU score output**
```

```
Average BLEU Score: 0.016586875530968
```

Why low BLEU Score?

- BLEU evaluates exact word matches.
- Length Discrepancy
- Not enough training data, took around 24+ hours to generate 2k data points

Result

```
[40] average_bleu = calculate_bleu(predictions, references)
     print("Average BLEU Score:", average_bleu)  # **Final BLEU score output**

Average BLEU Score: 0.01716652213901337

[41] predictions[5]

'in this paper we describe a new method for evaluating transverse spin correlations and quantum spin - fluctuation corrections about the hf - level broken - symmetry state, in terms of magnon mode energies and spectral functions obtained in the random phase approximation. in this paper we describe a new method for evaluating transverse spin correlations and quantum spin - fluctuation corrections about the hf - level broken - symmetry state, in terms of magnon mode energies and spectral functions obtained in the random phase approximation.the effective energy of the single - mode magneton with a constant spin is then taken to be 0.6 g, thus making one of the leading orders of the hamiltonian can be used to calculate the effective spin correlations of the doped model. this leads to, and in particular a good approximation can be constructed on the basis of the following two - dimensional form : one'

references[5]

'a numerical method is described for evaluating transverse spin correlations in the random phase approximation . quantum spin - fluctuation corrections to sublattice magnetization are evaluated for the antiferromagnetic ground state of the half - filled hubbard model in two and three dimensions in the whole . extension to the case of defects in the af is also discussed for spin vacancies and low impurities . in the , the vacancy - induced enhancement in the spin fluctuation correction is obtained for the spin - vacancy problem in two dimensions , for vacancy concentration up to the percolation threshold . for low- , the overall spin fluctuation correction is found to be strongly suppressed , although surprisingly spin fluctuations are locally enhanced at the low sites . 2'
```

Models Used and Why?

Pegasus-Large for Research paper

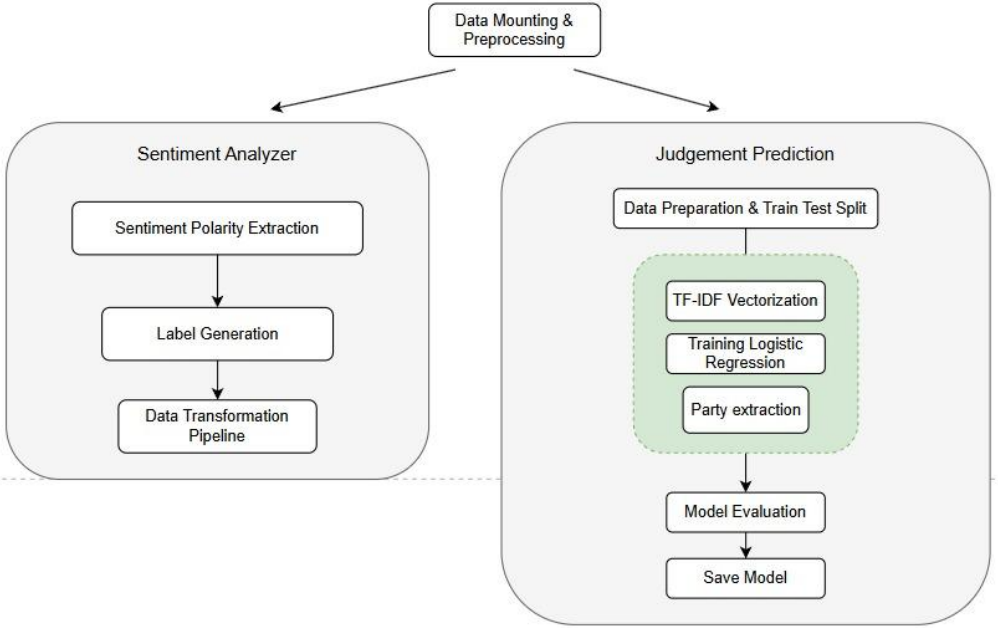
DEF: Hybrid model for extractive and abstractive tasks; reconstructs text effectively.

WHY: Zero-shot Capabilities: Since it's a general-purpose summarization model, you can input raw text (like a research paper) and expect it to generate a decent summary.

Pegasus was trained on datasets where entire sentences are masked, forcing the model to learn how to generate coherent summaries from surrounding context. This makes it well-suited for summarization even without fine-tuning. Pegasus-large has been trained on a wide range of datasets, so it can generalize well to many types of text, including academic writing.

GPT2's autoregressive design can generate meaningful summaries by synthesizing input information, which becomes more accurate and domain-specific after fine-tuning.

T5 NOT USED: Models like T5 are less optimized for the technical requirements of research paper summarization, and their higher computational cost was not justified for this use case.



MODULE 4: LEGAL DOCUMENTS

WorkFlow:

1. Data mounting and Preprocessing
2. **Sentiment Analysis**

   It is to to automatically determine the overall emotional tone or attitude expressed within a text, classifying it as positive, negative, or neutral based on the words and phrases used, essentially "reading between the lines" to understand the sentiment conveyed in the document.

1. Data Prep and Splitting
2. Model Saving
3. Party Extraction & Judgement Prediction

Justice dataset

	A	B	C	D	E	F	G	H	I	J	K	L
1		ID	name	href	docket	term	first_party	second_party	facts	facts_len	majority_vote	minority_vote
2	0	50606	Roe v. Wade	https://api.oyez.org/c	70-18	1971	Jane Roe	Henry Wade	<p>In 1970, Jane Rc	501	7	2
3	1	50613	Stanley v. Illinois	https://api.oyez.org/c	70-5014	1971	Peter Stanley, Sr.	Illinois	<p>Joan Stanley hac	757	5	2
4	2	50623	Giglio v. United Stat	https://api.oyez.org/c	70-29	1971	John Giglio	United States	<p>John Giglio was	495	7	0
5	3	50632	Reed v. Reed	https://api.oyez.org/c	70-4	1971	Sally Reed	Cecil Reed	<p>The Idaho Proba	378	7	0
6	4	50643	Miller v. California	https://api.oyez.org/c	70-73	1971	Marvin Miller	California	<p>Miller, after conc	305	5	4
7	5	50644	Kleindienst v. Mande	https://api.oyez.org/c	71-16	1971	Richard G. Kleindier	Ernest E. Mandel, et	<p>Ernest E. Mandel <p>The Graduate St <p>Mandel, along w	2282	6	3
8	6	50655	Samo v. Illinois Crim	https://api.oyez.org/c	70-7	1971	Samo	Illinois Crime Investi	<p>The Illinois Crim <p>On February 8, 1 <p>On March 21, 19	1424	5	2
9	7	50656	Argersinger v. Hamli	https://api.oyez.org/c	70-5015	1971	Argersinger	Hamlin	<p>Jon Argersinger v	347	9	0
10	8	50657	Eisenstadt v. Baird	https://api.oyez.org/c	70-17	1971	Eisenstadt	Baird	<p>William Baird ga	420	6	1
11	9	50663	Gooding v. Wilson	https://api.oyez.org/c	70-26	1971	Gooding	Wilson	<p>A Georgia state c	612	5	2
12	10	50671	Furman v. Georgia	https://api.oyez.org/c	69-5003	1971	Furman	Georgia	<p>Furman was burg	477	5	4
13	11	50683	Moose Lodge No. 10	https://api.oyez.org/c	70-75	1971	Moose Lodge No. 10	Irvis	<p>K. Leroy Irvis, a b	415	6	3
14	12	50688	Branzburg v. Hayes	https://api.oyez.org/c	70-85	1971	Branzburg	Hayes	<p>After observing a	745	5	4

M	N	O	P
first_party_winner	decision_type	disposition	issue_area
TRUE	majority opinion	reversed	
TRUE	majority opinion	reversed/remanded	Civil Rights
TRUE	majority opinion	reversed/remanded	Due Process
TRUE	majority opinion	reversed/remanded	Civil Rights
TRUE	majority opinion	vacated/remanded	First Amendment
TRUE	majority opinion	reversed	First Amendment
FALSE	per curiam		Criminal Procedure
TRUE	majority opinion	reversed	Criminal Procedure
FALSE	majority opinion	affirmed	Privacy
FALSE	majority opinion	affirmed	First Amendment
TRUE	per curiam	reversed/remanded	Criminal Procedure

- There are 3302 case data recorded for training
- *Acknowledgements*
  Mohammad Alali, Shaayan Syed, Mohammed Alsayed, Smit Patel, Hemanth Bodala

Data Preprocessing

```
print(legal_df.columns) #legal_df DataFrame cols check
print("Columns in justice_df:", justice_df.columns.tolist()) #justice_df cols check
```

```
Index(['case_id', 'case_outcome', 'case_title', 'case_text'], dtype='object')
Columns in justice_df: ['Unnamed: 0', 'ID', 'name', 'href', 'docket', 'term', 'first_party', 'second_party', 'facts', 'facts_len', 'majority_vote',
```

```
#[DATA PREPROCESSING]
first_party_winner_column = 'first_party_winner'  #use correct col name for first party & update to actual col name
justice_df['facts'] = justice_df['facts'].fillna('')  #replace NaNs in 'facts' with empty strings

if first_party_winner_column in justice_df.columns: #if first_party_winner_col exists
    justice_df[first_party_winner_column] = justice_df[first_party_winner_column].fillna(0)  #replace NaNs with 0
    justice_df[first_party_winner_column] = justice_df[first_party_winner_column].astype(int) #convert to int
else:
    raise KeyError(f"Column '{first_party_winner_column}' does not exist in justice_df.")
```

Sentiment Analysis

```python
#[SENTIMENT ANALYSER]
from textblob import TextBlob
def get_sentiment(text):
    analysis = TextBlob(text)
    return analysis.sentiment.polarity
def classify_sentiment(polarity):
    if polarity > 0:
        return "Positive"
    elif polarity < 0:
        return "Negative"
    else:
        return "Neutral"
justice_df['facts_sentiment'] = justice_df['facts'].apply(get_sentiment)
justice_df['sentiment_label'] = justice_df['facts_sentiment'].apply(classify_sentiment)
print(justice_df[['facts', 'facts_sentiment', 'sentiment_label']].head())
```

```
                                               facts  facts_sentiment  \
0  <p>In 1970, Jane Roe (a fictional name used in...     3.571429e-03
1  <p>Joan Stanley had three children with Peter ...    -6.944444e-02
2  <p>John Giglio was convicted of passing forged...    -4.545455e-02
3  <p>The Idaho Probate Code specified that "male...     0.000000e+00
4  <p>Miller, after conducting a mass mailing cam...     9.251859e-18

  sentiment_label
0        Positive
1        Negative
2        Negative
3         Neutral
4        Positive
```

Data Prep & Splitting

```
#[PREP FOR DATA TRAINING]
import joblib
X = justice_df['facts']
y = justice_df[first_party_winner_column]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model_pipeline = make_pipeline(TfidfVectorizer(), LogisticRegression())

#[MODEL TRAINING]
model_pipeline.fit(X_train, y_train)
y_pred = model_pipeline.predict(X_test)
print(classification_report(y_test, y_pred))

#[MODEL SAVING FOR FUTURE REFERENCES]
model_filename = '/content/drive/MyDrive/datasets/lawDoc_proj/legal_model.pkl'
joblib.dump(model_pipeline, model_filename)
print(f"Model saved to {model_filename}")
```

Party Extraction & Justice Prediction - I

```python
#[JUDGEMENT PREDICTION]
import re
import joblib
loaded_model = joblib.load(model_filename)

def predict_judgment(facts):
    prediction = loaded_model.predict([facts])
    return prediction[0]
```

```python
def extract_parties(facts):
    match = re.search(r"([\w\s,]+)\s+v(?:\.|s\.?)\s+([\w\s,]+)", facts, re.IGNORECASE)
    if match:
        first_party = match.group(1).strip()
        second_party = match.group(2).strip()
        return first_party, second_party

    match = re.search(r"In the case of ([\w\s]+) and ([\w\s]+)", facts, re.IGNORECASE)
    if match:
        first_party = match.group(1).strip()
        second_party = match.group(2).strip()
        return first_party, second_party

    match = re.search(r"([\w\s]+)\s+against\s+([\w\s]+)", facts, re.IGNORECASE)
    if match:
        first_party = match.group(1).strip()
        second_party = match.group(2).strip()
        return first_party, second_party

    return None, None
```

Party Extraction & Justice Prediction - II

```
#[INPUT CASE FOR PREDICTIONS]
new_case_facts = "After observing and interviewing a number of people synthesizing and using drugs in a two-county area in Kentucky, Branzburg, a rep

predicted_judgment = predict_judgment(new_case_facts)
first_party, second_party = extract_parties(new_case_facts)
label_mapping = {0: "Second Party Wins", 1: "First Party Wins"}
predicted_outcome = label_mapping[predicted_judgment]

if first_party and second_party:
    print(f"Parties involved: \nFirst Party: '{first_party}' \nSecond Party: '{second_party}'")
else:
    print("Could not extract party names from the facts.")
print(f"The predicted judgment is: {predicted_outcome}")
```

```
Parties involved:
First Party: 'Similarly, in the companion cases of In re Pappas and United States'
Second Party: 'Caldwell, two different reporters, each covering activity within the Black Panther organization, were called to testify before grand ju
The predicted judgment is: Second Party Wins
```

The Input Case - 1

After observing and interviewing a number of people synthesizing and using drugs in a two-county area in Kentucky, Branzburg, a reporter, wrote a story which appeared in a Louisville newspaper. On two occasions he was called to testify before state grand juries which were investigating drug crimes. Branzburg refused to testify and potentially disclose the identities of his confidential sources. Similarly, in the companion cases of In re Pappas and United States v. Caldwell, two different reporters, each covering activity within the Black Panther organization, were called to testify before grand juries and reveal trusted information. Like Branzburg, both Pappas and Caldwell refused to appear before their respective grand juries.

Test case

Input Case - 2 ***Rhinebeck Central School District & Thomas Mawhinney (March 22, 2006)***

On March 18, 2004, the United States Attorney's Office for the Southern District of New York and the Section moved to intervene in A.B. v. Rhinebeck Central School District and Thomas Mawhinney, a sexual harassment case brought against the Rhinebeck Central School District and the former high school principal Thomas Mawhinney. The case was filed in the United States District Court for the Southern District of New York on May 9, 2003, by four current and former high school students and a school employee. The plaintiffs alleged that the school district and Mawhinney violated state and federal laws, including Title IX. The United States filed an intervention brief and complaint-in-intervention alleging that Mawhinney sexually harassed the four plaintiff students as well as other female high school students during his ten-year tenure as principal and that the school district violated Title IX by acting with deliberate indifference to known sexual harassment of these students.

Decision

$152,500 to compensate the student victims and to pay their attorney's fees. (First Party)

```
new_case_facts = "On March 18, 2004, the United States Attorney's Office for the Southern District of New York

predicted_judgment = predict_judgment(new_case_facts)
first_party, second_party = extract_parties(new_case_facts)
label_mapping = {0: "Second Party Wins", 1: "First Party Wins"}
predicted_outcome = label_mapping[predicted_judgment]

print(f"The predicted judgment is: {predicted_outcome}")
```

```
The predicted judgment is: First Party Wins
```

Models Used and Why?

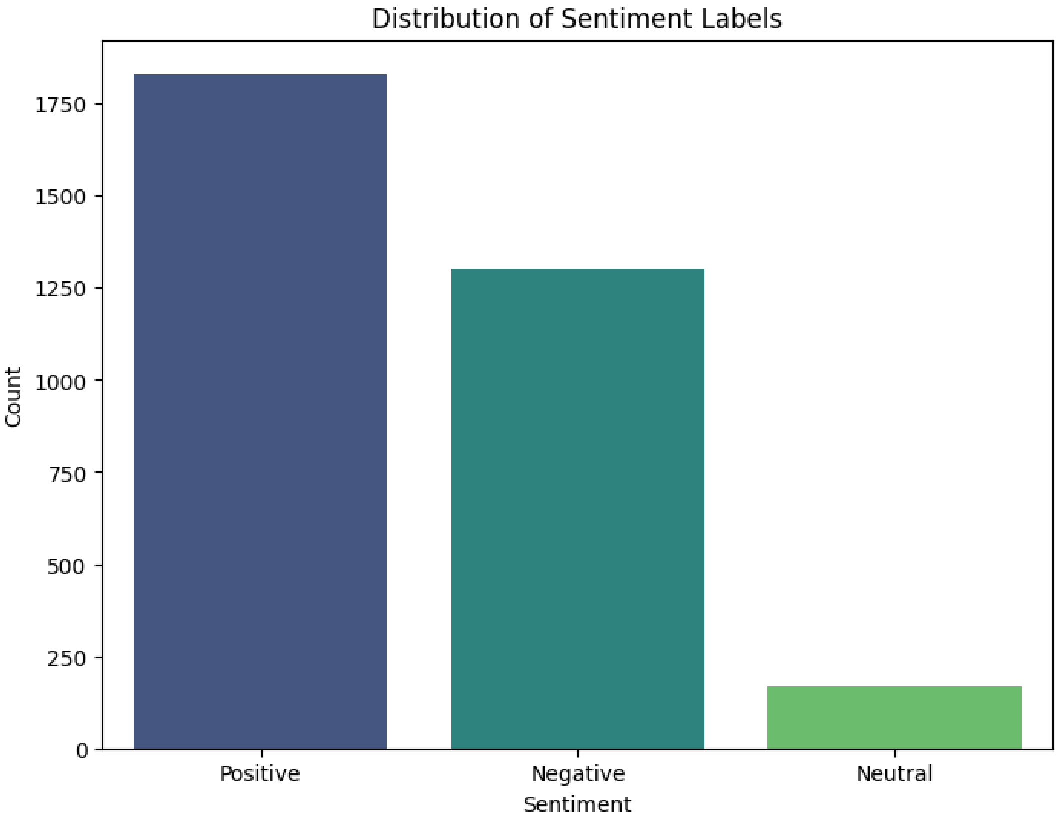
Logistic Regression

DEF: Logistic Regression is a classification algorithm that predicts binary outcomes by modeling probability of each class and learns to classify cases based on these features.

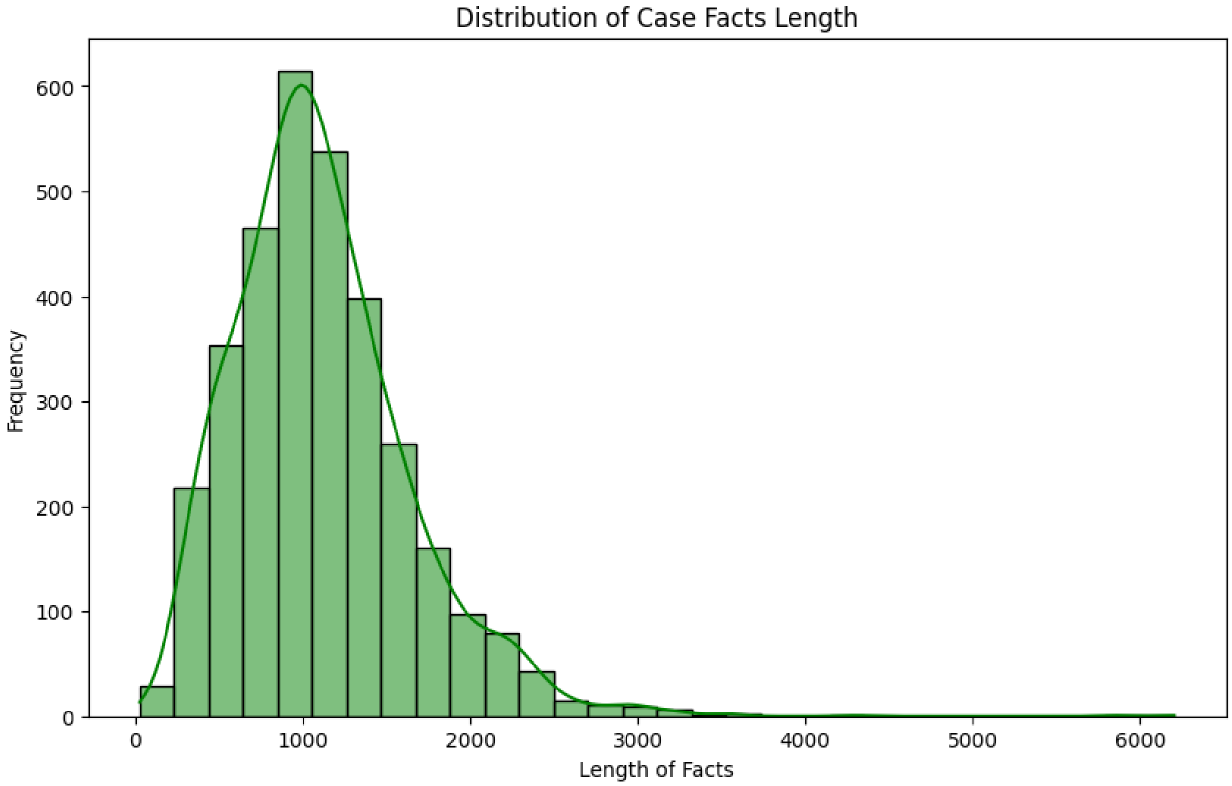
WHY:

- ❑ is one of the simplest models for binary classification
- ❑ its coefficients directly show the importance of each feature in the decision-making process
- ❑ TF-IDF vectorizer outputs sparse feature matrices and Logistic Regression is suitable for sparse data
- ❑ handles datasets with many features (words) and relatively fewer samples (cases)
- ❑ is computationally efficient and requires fewer resources
- ❑ prevents overfitting by penalizing large coefficients as it supports L1 and L2 regularization
- ❑ provides probabilities for each class, giving confidence scores for predictions
- ❑ serves as a good baseline model for classification tasks that can be developed upon in future

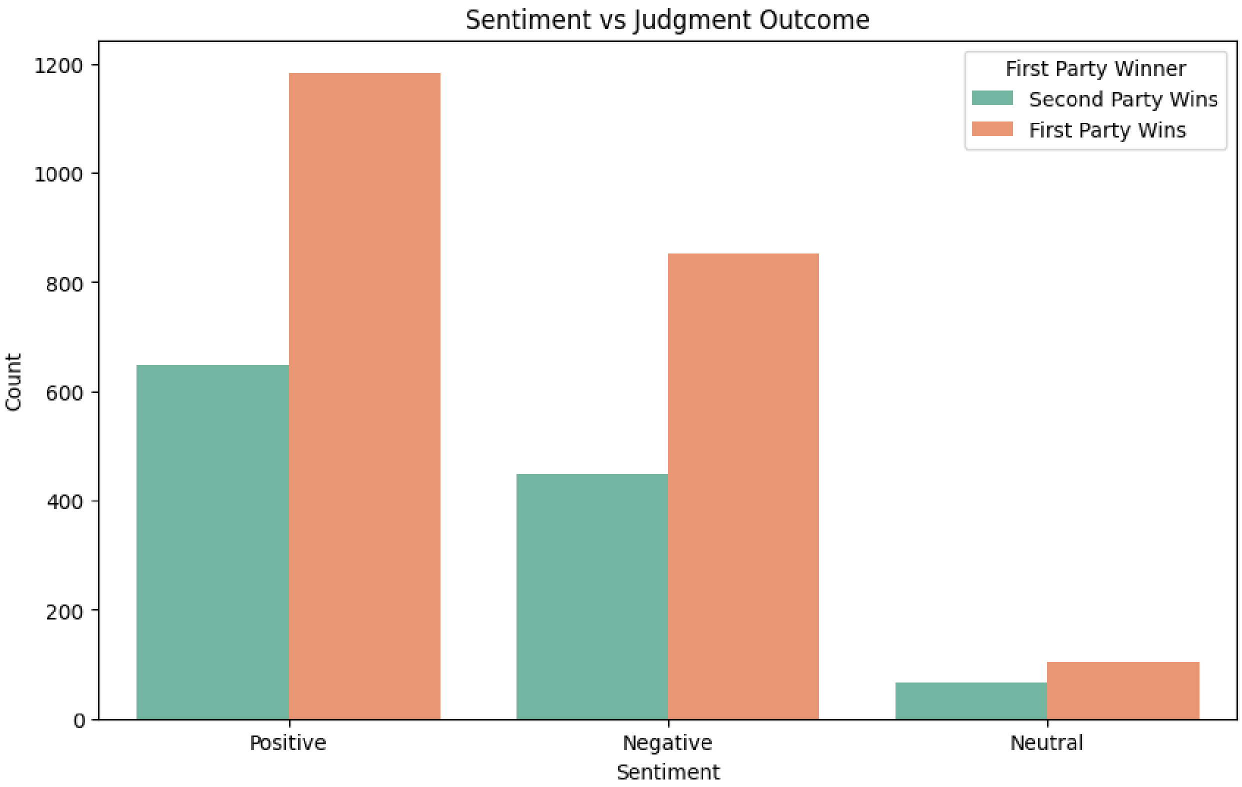
Distribution of Sentiments



Distribution of Case Facts Lengths



Sentiment vs Judgement



THANK YOU

Do you have any questions?