AI-Powered Financial Report Analysis: Evaluating Models Outputs for confident Insights and reliability

horizontal line

# 

| Sno | Contents | PNo |
| --- | --- | --- |
|  | Introduction |  |
|  | Background and motivation |  |
|  | Objectives of the project |  |
|  | Scope and limitations |  |
|  | Overview of NLP in finance |  |
|  | Previous work on summarization of financial reports |  |
|  |  |  |
|  | Techniques and models used in AI-based summarization |  |
|  | Overview of GPT-based models |  |
|  | Criteria for selecting the GPT model |  |
|  | Fine-tuning approach for financial report summarization |  |
|  |  |  |
|  | Evaluation Metrics |  |
|  | Introduction to evaluation metrics (ROUGE, BLEU, etc.) |  |
|  | Selection of metrics for evaluating summaries |  |
|  | Explanation of chosen metrics and their relevance to financial summaries |  |
|  |  |  |
|  | Experiment Design |  |
|  | Description of experimental setup |  |
|  | Training data split (train, validation, test) |  |
|  | Hyperparameter tuning strategies |  |
|  | Implementation |  |
|  |  |  |
|  | Integration of selected GPT model for summarization |  |
|  | Coding details and libraries used |  |
|  | Handling of specific financial domain terminologies and nuances |  |
|  |  |  |
|  | Evaluation |  |
|  | Quantitative evaluation using chosen metrics |  |
|  | Qualitative assessment of summary quality |  |
|  | Comparison with baseline models or human-generated summaries |  |
|  |  |  |
|  | Results and Discussion |  |
|  | Presentation of evaluation results |  |
|  | Interpretation of metrics and performance analysis |  |
|  | Discussion on model strengths, weaknesses, and areas for improvement |  |
|  |  |  |
|  | Conclusion |  |
|  | Summary of key findings |  |
|  | Contributions of the project |  |
|  | Future directions and recommendations |  |
|  |  |  |
|  | References |  |
|  | Citations for literature, datasets, and tools used |  |
|  |  |  |
|  | Appendices |  |
|  | Supplementary materials (code snippets, sample summaries) |  |
|  | Glossary of terms used in financial report summarization |  |

Introduction

In today's data-driven financial landscape, the ability to extract actionable insights from vast volumes of information is paramount. Financial reports, while rich in valuable data, can be dense and complex, making it challenging for stakeholders to glean key takeaways efficiently. This challenge has spurred the development of AI-powered solutions for automating the summarization of financial reports, aiming to distill crucial information into concise and digestible formats.

This report delves into the realm of Natural Language Processing (NLP) and Artificial Intelligence (AI) to explore how GPT-based models can transform the way financial reports are analyzed and understood. This introduction sets the stage by providing context, outlining the objectives, discussing the significance of the project, and defining the scope.

Background and Motivation

Financial reports serve as vital documents that encapsulate the financial health, performance, and strategies of companies, institutions, and markets. However, the traditional approach to digesting these reports involves manual reading, which is time-consuming and prone to human errors. Moreover, the sheer volume of reports generated regularly necessitates scalable and efficient solutions for extracting meaningful insights.

The advent of Generative Pre-trained Transformers (GPT) and similar AI models has revolutionized text generation and understanding. These models, trained on vast datasets, possess the capability to comprehend context, extract relevant information, and generate coherent summaries. Leveraging such advancements in AI for financial report summarization holds immense promise in streamlining decision-making processes and facilitating informed actions.

Objectives

The primary objective of this project is to evaluate the efficacy of GPT-based models in summarizing financial reports accurately and comprehensively.

1. Explore evaluation metrics to assess the quality and coherence of generated summaries.
2. Explore the impact of fine-tuning and domain-specific training on summary quality.
3. Provide insights into the strengths, limitations, and potential applications of AI-powered summarization in the financial domain.

Impact of the report

By harnessing the power of AI, organizations can expedite decision-making processes, identify trends and anomalies swiftly, and enhance overall financial literacy. Moreover, automated summarization can empower stakeholders across industries, including investors, analysts, regulators, and executives, to extract actionable insights with ease and accuracy.

The insights gained from this report have the potential to inform future developments in AI-driven financial analytics, paving the way for more sophisticated and efficient tools for data analysis and interpretation.

Scope of the Project

While the project focuses primarily on evaluating GPT-based models for financial report summarization, it acknowledges the broader landscape of AI and NLP technologies. The scope includes:

1. Implementation and fine-tuning of GPT-based models.
2. Development of evaluation methodologies and metrics.
3. Discussion of ethical considerations, biases, and interpretability in AI-generated summaries.

In summary this report embarks on a journey to harness the potential of AI in transforming the accessibility and utility of financial information. By conducting rigorous evaluations and analyses, this report aims to contribute valuable insights to the intersection of AI, NLP, and finance, with implications for industries, academia, and technological advancements.

Overview of NLP in Finance

Natural Language Processing (NLP) has become a pivotal technology in the financial industry, revolutionizing how data is processed, analyzed, and utilized. In finance, NLP techniques are deployed across various domains, including sentiment analysis, fraud detection, customer support, and document summarization. These applications leverage NLP's ability to understand and derive insights from unstructured textual data, such as news articles, social media posts, financial reports, and regulatory filings.

NLP models, particularly transformer-based architectures like BERT (Bidirectional Encoder Representations from Transformers) and GPT (Generative Pre-trained Transformers), have demonstrated remarkable capabilities in language understanding, context comprehension, and text generation. These models, pre-trained on a vast corpora of text, can be fine-tuned for specific tasks, making them versatile tools for analyzing and summarizing financial data.

Previous Work on Summarization of Financial Reports

The summarization of financial reports has garnered significant attention in both academia and industry due to its potential to streamline information processing and decision-making. Several studies and initiatives have explored various approaches and techniques for automating the summarization of financial documents, with a focus on improving efficiency, accuracy, and relevance.

Rule-based Summarization Techniques: Early efforts in financial report summarization often relied on rule-based systems that extracted key sentences or phrases based on predefined rules and heuristics. While these methods provided basic summarization capabilities, they were limited in handling complex structures and nuances present in financial texts.

Statistical and Machine Learning Approaches: With the advent of statistical and machine learning algorithms, researchers began experimenting with supervised and unsupervised methods for summarization. Techniques such as extractive summarization, where important sentences or passages are selected from the original text, gained traction. Algorithms like TextRank and PageRank were employed to identify salient information based on graph-based ranking.

Deep Learning and Transformer Models: The emergence of deep learning architectures, particularly transformer models like BERT and GPT, marked a significant advancement in financial report summarization. These models, with their attention mechanisms and contextual understanding capabilities, excel in generating coherent and informative summaries. Fine-tuning pre-trained transformer models for financial domain-specific tasks has shown promising results in producing concise yet comprehensive summaries.

Evaluation Metrics and Benchmarking: Alongside advancements in summarization techniques, efforts have been made to establish robust evaluation metrics and benchmarks for assessing the quality of generated summaries. Metrics such as ROUGE (Recall-Oriented Understudy for Gisting Evaluation) and BLEU (Bilingual Evaluation Understudy) are commonly used to measure the overlap and fluency of summaries compared to reference texts. Benchmark datasets like the Financial PhraseBank and SEC filings have facilitated standardized evaluation practices and benchmarking of summarization models.

Applications and Use Cases: The application of automated summarization in finance extends beyond internal document analysis. It finds utility in financial news aggregation, regulatory compliance, investment analysis, and risk management. Automated summaries enable faster information assimilation, trend identification, and decision support, benefiting various stakeholders in the financial ecosystem.

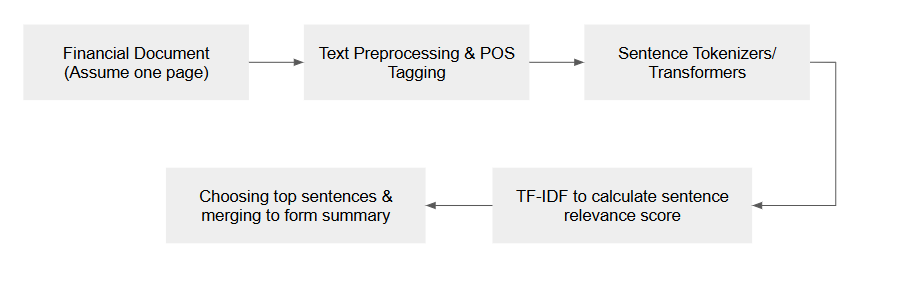
Techniques and Models Used in AI-Based Summarization

Automated summarization has emerged as a crucial application of Artificial Intelligence (AI) and Natural Language Processing (NLP), facilitating the extraction of essential information from large volumes of text. This report provides an overview of techniques and models used in AI-based summarization, focusing on both extractive and abstractive summarization approaches.

1. Extractive Summarization Techniques

Extractive summarization methods aim to identify and extract the most important sentences or passages from the original text to create a summary. Several techniques have been developed to achieve this, including:

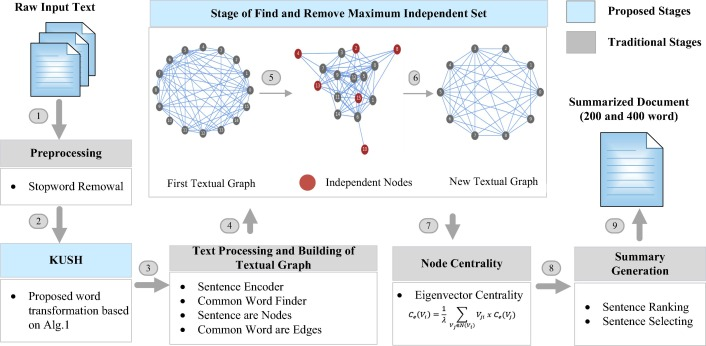
TF-IDF (Term Frequency-Inverse Document Frequency): This statistical measure evaluates the importance of a word in a document relative to a corpus. Sentences with high TF-IDF scores are considered significant and are included in the summary.



TextRank Algorithm: Inspired by Google's PageRank algorithm, TextRank creates a graph representation of sentences, where nodes represent sentences and edges denote relationships based on similarity or co-occurrence. Sentences with high centrality scores in the graph are selected for the summary.

[https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.researchgate.net%2Ffigure%2FFlowchart-of-indonesian-sentiment-summarization-by-using-textrank-algorithm_fig1_350151912&psig=AOvVaw0j6tIYPQDzyIL3sM6PaZTn&ust=1717170848118000&source=images&cd=vfe&opi=89978449&ved=0CBQQjhxqFwoTCJCQ4YXetYYDFQAAAAAdAAAAABAQ
](https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.researchgate.net%2Ffigure%2FFlowchart-of-indonesian-sentiment-summarization-by-using-textrank-algorithm_fig1_350151912&psig=AOvVaw0j6tIYPQDzyIL3sM6PaZTn&ust=1717170848118000&source=images&cd=vfe&opi=89978449&ved=0CBQQjhxqFwoTCJCQ4YXetYYDFQAAAAAdAAAAABAQ)

Graph-Based Methods: Besides TextRank, graph-based summarization includes algorithms like LexRank and Sentence Mover's Similarity (SMS), which leverage graph structures to identify key sentences.

[](https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.sciencedirect.com%2Fscience%2Farticle%2Fpii%2FS111086651930324X&psig=AOvVaw1v-LgmMnPS7_1ORHBeTO7X&ust=1717170970978000&source=images&cd=vfe&opi=89978449&ved=0CBQQjhxqFwoTCLiDor_etYYDFQAAAAAdAAAAABAI)

Machine Learning Approaches: Supervised and unsupervised machine learning techniques, such as Support Vector Machines (SVM), clustering algorithms, and neural networks, have been applied to extractive summarization tasks, learning to identify salient information based on training data.

2. Abstractive Summarization Models

Abstractive summarization goes beyond extraction by generating summaries that may not exist verbatim in the original text. This approach involves understanding the content and context of the text to produce concise and coherent summaries. Key models and techniques in abstractive summarization include:

Recurrent Neural Networks (RNNs): RNNs, particularly Long Short-Term Memory (LSTM) networks, have been used for abstractive summarization tasks, capturing sequential dependencies in text and generating summaries based on learned patterns.

Sequence-to-Sequence (Seq2Seq) Models: Seq2Seq models, often implemented with encoder-decoder architectures, have shown success in abstractive summarization. Variants like Attention-based Seq2Seq models improve the focus on relevant information during summarization.

Transformer Models: Transformer architectures, exemplified by models like BERT (Bidirectional Encoder Representations from Transformers) and GPT (Generative Pre-trained Transformers), have revolutionized abstractive summarization. These models excel in understanding context, handling long-range dependencies, and generating fluent and informative summaries.

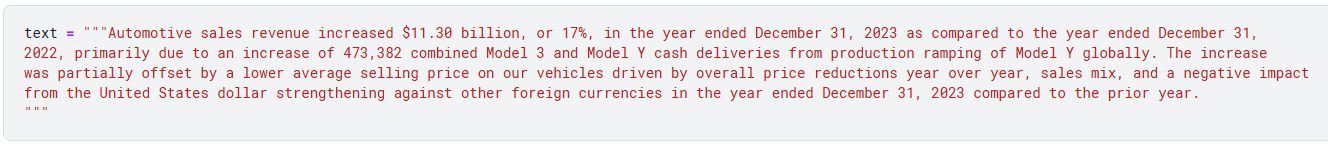
3. Hybrid Approaches

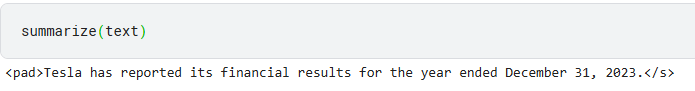
Hybrid approaches combine elements of extractive and abstractive summarization to leverage the strengths of both techniques. For instance, an extractive step may be used to identify important sentences, which are then paraphrased or merged to form an abstractive summary. These approaches aim to improve the coherence and informativeness of generated summaries.

| Aspect | Extractive Summarization | Abstractive Summarization |
| --- | --- | --- |
| Definition | Selects and extracts key sentences or phrases directly from the source text. | Generates new sentences that capture the meaning of the source text. |
| Methodology | Relies on identifying and ranking important sentences based on predefined criteria (e.g., frequency, position, relevance). | Uses advanced NLP models to understand and rephrase the text, often involving deep learning techniques like sequence-to-sequence models. |
| Output | The summary is a subset of the original text, with sentences taken verbatim. | The summary contains newly generated sentences that might not appear in the source text. |
| Coherence | Generally maintains the original context and coherence since sentences are directly taken from the text. | May produce more coherent and fluent summaries as it constructs sentences, but can also risk generating less accurate information. |
| Accuracy | High accuracy in terms of factual correctness, as it uses exact sentences from the text. | May introduce minor inaccuracies or misinterpretations, especially if the model is not well-trained. |
| Complexity | Relatively simpler and less computationally intensive. | More complex and computationally expensive due to the need for sophisticated models and training. |
| Flexibility | Limited to the information explicitly present in the source text. | More flexible and capable of paraphrasing, generalizing, and inferring information. |
| Use Cases | Suitable for tasks where factual correctness and extraction of key points are critical, such as legal and scientific documents. | Ideal for applications needing human-like summaries, such as news articles, storytelling, and conversational AI. |
| Examples of Techniques | TextRank, LexRank, LSA (Latent Semantic Analysis). | Transformer models like BERT, GPT, T5, and sequence-to-sequence models with attention mechanisms. |
| Development | Easier to develop and implement, often requiring less data for training. | Requires extensive training data and computational resources to develop effective models. |
| Pros | Ensures high fidelity to the source text, easy to implement, and typically requires less processing power. | Produces more concise and human-like summaries, can interpret and rephrase the text innovatively. |
| Cons | Can be less coherent and may include redundant or unimportant information. | Risk of generating inaccurate summaries, more challenging to train and fine-tune. |

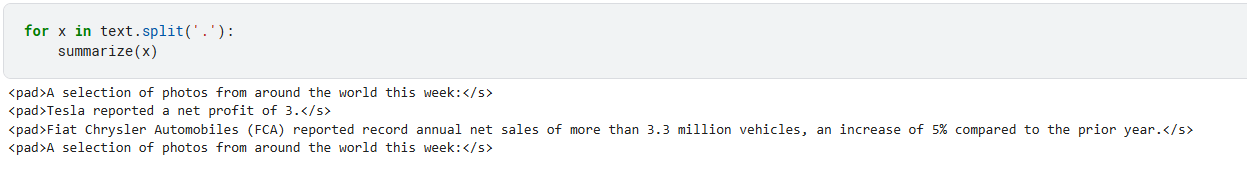
Lets compare the outputs of multiple models in the summarization task.

google/pegasus-xsum



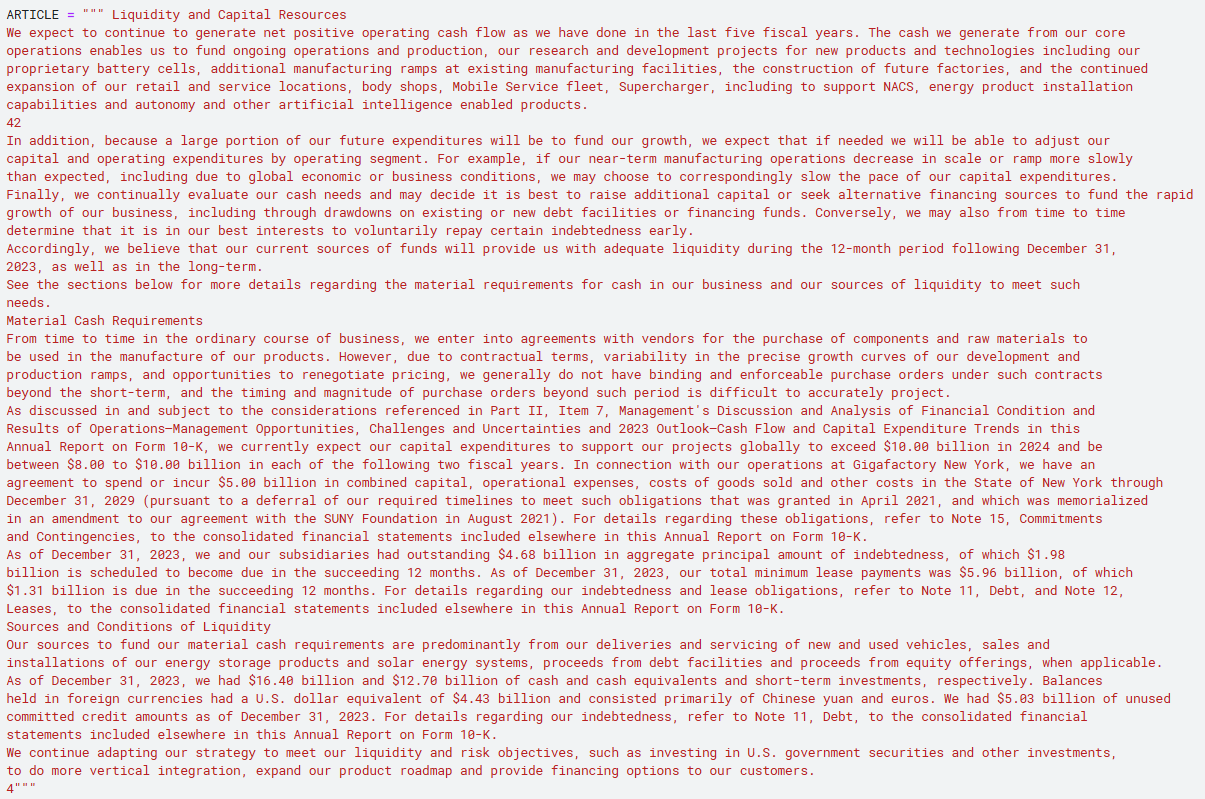


The above is not impressive or useful in any way. However the following is the output that tries to increase the output by splitting the input into multiple sentences. Which introduces hallucinations.

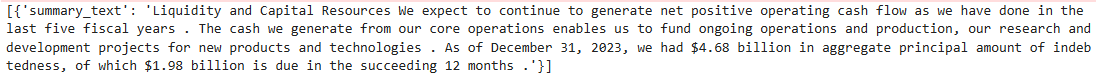


This is an example where it shows that models perform better and need large input tokens in order to provide better summarized outputs. This is more evident in the following examples.

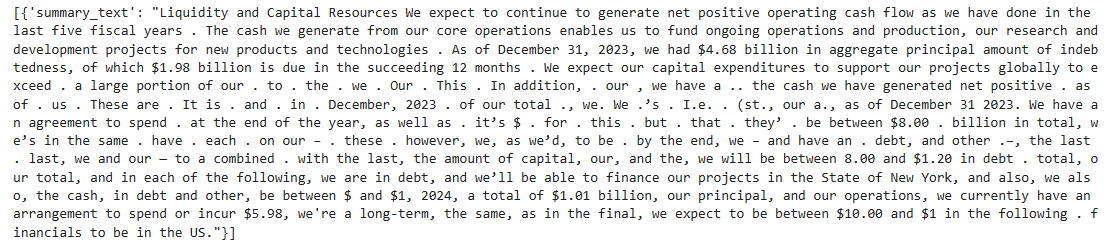
Falconsai/text\_summarization



This summary is when the minimum length is expected to be 30 tokens in length.



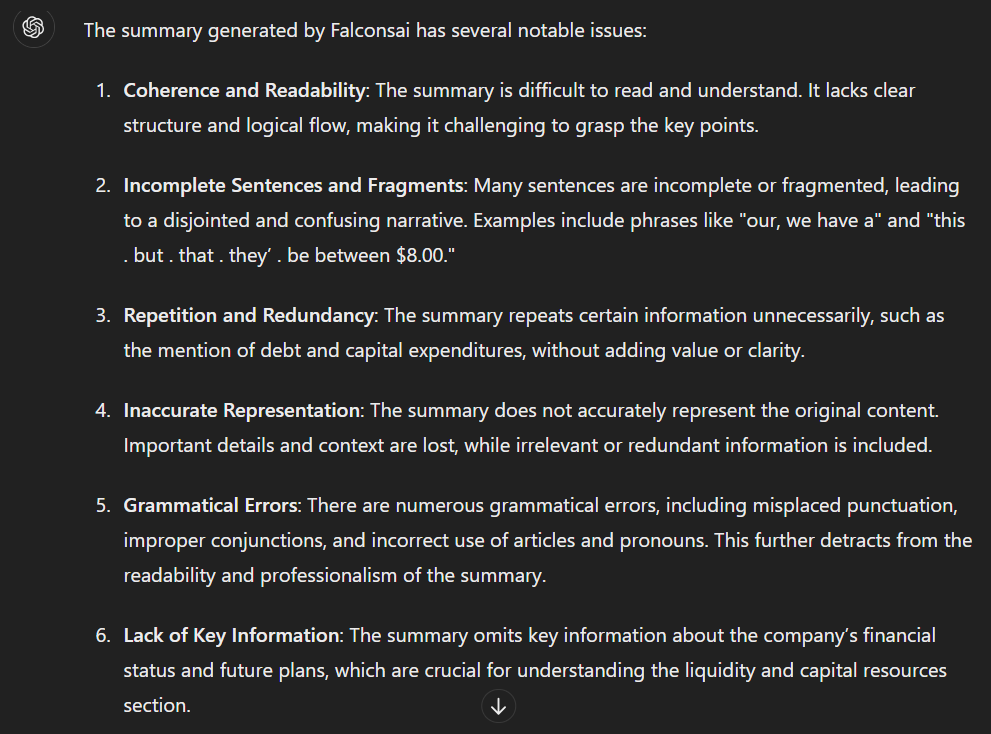
However, when we increase the minimum token length parameter to 500. We receive the following.



Since we will discuss using LLMs to evaluate other LLMs, lets see a quick example of the same here. The following is the evaluation provided by GPT-4o.

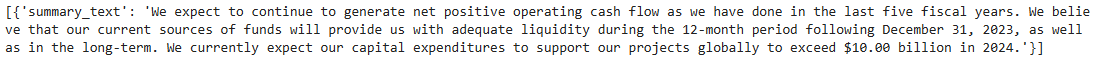
The prompt used was the following:

Task: Critique the following summary generated by Falconsai. Context: Input tokens are enclosed within parentheses. Summarised/output tokens are enclosed within triple quotes.

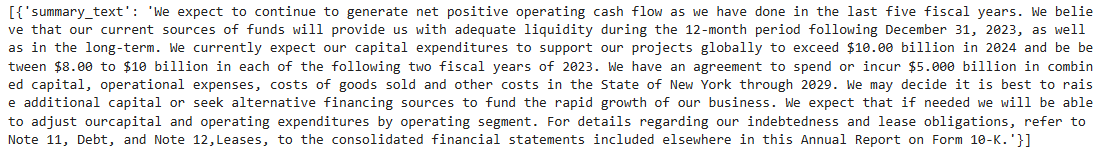


facebook/bart-large-cnn

Same input as the Falconsai as this one supports similar input token length.



By improving the minimum and maximum length parameters we receive the following.



human-centered-summarization/financial-summarization-pegasus

Input:

text\_to\_summarize = "National Commercial Bank (NCB), Saudi Arabia’s largest lender by assets, agreed to buy rival Samba Financial Group for $15 billion in the biggest banking takeover this year.NCB will pay 28.45 riyals ($7.58) for each Samba share, according to a statement on Sunday, valuing it at about 55.7 billion riyals. NCB will offer 0.739 new shares for each Samba share, at the lower end of the 0.736-0.787 ratio the banks set when they signed an initial framework agreement in June.The offer is a 3.5% premium to Samba’s Oct. 8 closing price of 27.50 riyals and about 24% higher than the level the shares traded at before the talks were made public. Bloomberg News first reported the merger discussions.The new bank will have total assets of more than $220 billion, creating the Gulf region’s third-largest lender. The entity’s $46 billion market capitalization nearly matches that of Qatar National Bank QPSC, which is still the Middle East’s biggest lender with about $268 billion of assets."



A more advanced version of evaluation using LLMs.

microsoft/Phi-3-mini-128k-instruct

Example prompt:

messages = [

{"role": "user", "content": "Evaluate the following summarization performed by an LLM. Input is enclosed within triple slashes and output is enclosed within triple equals."},

{"role": "assistant", "content": "///National Commercial Bank (NCB), Saudi Arabia’s largest lender by assets, agreed to buy rival Samba Financial Group for $15 billion in the biggest banking takeover this year.NCB will pay 28.45 riyals ($7.58) for each Samba share, according to a statement on Sunday, valuing it at about 55.7 billion riyals. NCB will offer 0.739 new shares for each Samba share, at the lower end of the 0.736-0.787 ratio the banks set when they signed an initial framework agreement in June.The offer is a 3.5% premium to Samba’s Oct. 8 closing price of 27.50 riyals and about 24% higher than the level the shares traded at before the talks were made public. Bloomberg News first reported the merger discussions.The new bank will have total assets of more than $220 billion, creating the Gulf region’s third-largest lender. The entity’s $46 billion market capitalization nearly matches that of Qatar National Bank QPSC, which is still the Middle East’s biggest lender with about $268 billion of assets./// ===Saudi bank to pay a 3.5% premium to Samba share price. Gulf region’s third-largest lender will have total assets of $220 billion==="},

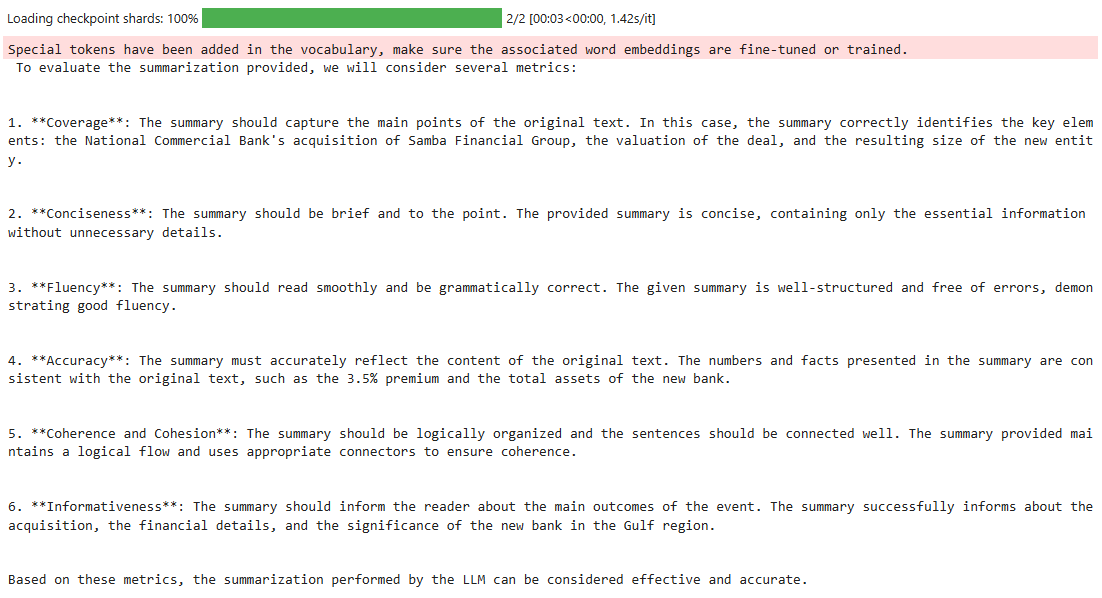
{"role": "user", "content": "Use evaluation metrics for summarization activity and prove your claims"},

]

Code snippet:



The output generated



Multiple prompts were attempted to achieve the current result. While this process was done manually, there are advanced techniques such as Parameter-Efficient Fine-Tuning (PEFT) that can automate this task. Research has demonstrated that PEFT significantly enhances performance, achieving up to twice the effectiveness in few-shot training scenarios. However, the input prompt tokens in this method are not easily interpretable.

Fine-tuning a pre-trained language model like GPT for financial report summarization

Fine-tuning a pre-trained language model like GPT for financial report summarizing involves adapting the model's parameters and training it on domain-specific data to improve its performance on the summarization task. This comprehensive guide outlines the fine-tuning approach for financial report summarization, covering data preparation, model configuration, training strategies, and evaluation techniques.

1. Data Preparation:

Dataset Selection: Choose a dataset of financial reports relevant to the target domain (e.g., annual reports, quarterly filings, earnings statements). Ensure the dataset is representative, diverse, and annotated with reference summaries or key points for training and evaluation.

Data Cleaning: Preprocess the data by removing noise, irrelevant information, and formatting inconsistencies. Tokenize the text into input-output pairs suitable for training the GPT model.

2. Model Configuration:

Selecting the GPT Model: Decide on the specific GPT variant (e.g., GPT-2, GPT-3) based on the task requirements, model size, and computational resources. Choose a model checkpoint that aligns with the desired balance between performance and resource efficiency.

Fine-tuning Architecture: Configure the GPT model for fine-tuning by adjusting hyperparameters such as learning rate, batch size, optimizer (e.g., Adam), dropout rate, and sequence length. Customize the model's architecture if necessary for better summarization performance.

3. Training Strategies:

Transfer Learning: Leverage transfer learning by initializing the GPT model with pre-trained weights from a large corpus of text data. This initialization helps the model capture general language patterns and semantics before fine-tuning on financial report data.

Domain Adaptation: Fine-tune the GPT model on the financial report dataset using supervised learning techniques. Employ techniques such as backpropagation, gradient descent, and weight updates to optimize the model for summarization tasks.

Iterative Training: Conduct iterative training sessions, monitoring the model's performance on validation data and adjusting hyperparameters or training strategies as needed to prevent overfitting and improve generalization.

4. Task-Specific Training Objectives:

Sequence-to-Sequence Learning: Frame the financial report summarization task as a sequence-to-sequence learning problem, where the model learns to generate concise summaries from input report sequences. Use techniques like teacher forcing or scheduled sampling to guide the model during training.

Loss Function: Define an appropriate loss function (e.g., cross-entropy loss) that penalizes discrepancies between generated summaries and reference summaries. Fine-tune the model to minimize this loss function during training iterations.

5. Evaluation and Validation:

Evaluation Metrics: Choose evaluation metrics suited for summarization tasks, such as ROUGE (Recall-Oriented Understudy for Gisting Evaluation) scores (ROUGE-N, ROUGE-L) to measure the similarity between generated summaries and reference summaries.

Validation Set: Set aside a validation dataset from the training data to monitor the model's performance during training. Use the validation set to tune hyperparameters and prevent overfitting.

Human Evaluation: Conduct human evaluation studies to assess the quality, coherence, and informativeness of generated summaries compared to human-generated summaries. Incorporate human feedback into model refinement and improvement.

6. Post-processing and Deployment:

Post-processing Techniques: Apply post-processing techniques such as sentence compression, entity recognition, and coherence checks to enhance the quality of generated summaries and ensure they meet readability and informativeness standards.

Deployment Considerations: Prepare the fine-tuned GPT model for deployment in production environments, considering factors such as inference speed, resource utilization, scalability, and integration with existing systems (e.g., APIs, web services).

Importance of Sentiment Analysis in Finance

Sentiment analysis in finance involves extracting and analyzing sentiment-related information from various textual sources, such as financial reports, earnings call transcripts, news articles, social media posts, and analyst reports. The importance of sentiment analysis in finance stems from its ability to:

1. Market Sentiment Analysis: Sentiment analysis helps gauge market sentiments towards specific assets, companies, or sectors. Positive sentiments often correlate with bullish market trends, while negative sentiments may signal bearish sentiments or potential risks.
2. Investor Perception: Understanding how investors perceive a company's performance, strategy, and outlook is crucial for investor relations and strategic decision-making. Sentiment analysis can identify key themes and sentiments expressed by investors in earnings calls or financial reports.
3. Risk Management: Sentiment analysis aids in identifying emerging risks, market trends, and sentiment shifts that may impact investment decisions, risk assessments, and portfolio management strategies.
4. Competitive Analysis: Analyzing sentiment across competitors' financial reports and market discussions provides insights into market positioning, competitive advantages, and areas of concern or improvement.
5. Predictive Analytics: By analyzing historical sentiment data alongside market performance, sentiment analysis can be used in predictive modeling and forecasting, helping anticipate market movements, investor behavior, and potential market reactions to events.

Real-World Applications of Sentiment Analysis in Finance

1. Investment Decision-Making: Financial institutions, hedge funds, and individual investors use sentiment analysis to inform investment decisions, assess market sentiments, and identify investment opportunities or risks.
2. Risk Management: Banks and financial institutions utilize sentiment analysis to monitor and manage risks related to market sentiment shifts, regulatory changes, and emerging trends impacting financial markets.
3. Corporate Performance Analysis: Companies use sentiment analysis on earnings call transcripts and financial reports to assess investor perceptions, identify areas of concern, and refine communication strategies with stakeholders.
4. Market Research: Market research firms and analysts leverage sentiment analysis to track consumer sentiments, assess product perceptions, and analyze brand reputation, aiding in market positioning and strategic marketing initiatives.
5. Algorithmic Trading: Quantitative trading strategies incorporate sentiment analysis data to develop algorithmic trading models, identify market trends, and make data-driven trading decisions based on sentiment signals.
6. Sentiment-Based Indexing: Sentiment-based indices and sentiment indicators are used to track sentiment trends in financial markets, providing valuable benchmarks for sentiment-driven investment strategies and market analysis.
7. Regulatory Compliance: Financial regulators use sentiment analysis to monitor market sentiments, detect potential market manipulations, and ensure compliance with regulatory guidelines.

Using LLM to evaluate sentiment predictions

import torch

from transformers import AutoModelForCausalLM, AutoTokenizer, pipeline

torch.random.manual\_seed(0)

model = AutoModelForCausalLM.from\_pretrained(

"microsoft/Phi-3-mini-128k-instruct",

device\_map="cuda",

torch\_dtype="auto",

trust\_remote\_code=True,

)

tokenizer = AutoTokenizer.from\_pretrained("microsoft/Phi-3-mini-128k-instruct")

The prompt used in this evaluation.

first\_instruction = """

Instructions:

Carefully read the given financial statement.

Determine the sentiment based on the criteria provided.

RESPOND only with the sentiment. positive, negative or neutral.

"""

first\_response = """You are an advanced language model trained to analyze the sentiment of text. Your task is to classify the sentiment of a given financial statement as Positive, Negative, or Neutral. Consider the following detailed criteria for each classification:

Positive: The sentence expresses favorable financial performance, growth, profitability, optimistic outlook, or any kind of positive sentiment related to the company's financial status.

Negative: The sentence conveys unfavorable financial performance, losses, risks, pessimistic outlook, or any kind of negative sentiment related to the company's financial status.

For example:

POSITIVE: Amazon's consistent revenue growth and strong financial performance over the past decade highlight its robust business model and strategic market positioning, making it a reliable and attractive investment option.

NEGATIVE: Amazon's mounting operational costs and the increasing regulatory scrutiny on its business practices pose significant risks to its future profitability and market valuation.

NEUTRAL: Amazon's financial performance this quarter met analysts' expectations, showing steady growth in revenue and a stable market presence, but with no significant surprises or deviations from projected outcomes.

"""

Some configurations

def get\_message(current\_sentence):

messages = [

{"role": "user", "content":first\_instruction},

{"role": "assistant", "content":first\_response},

{"role": "user", "content": current\_sentence}]

return messages

pipe = pipeline(

"text-generation",

model=model,

tokenizer=tokenizer,

)

generation\_args = {

"max\_new\_tokens": 500,

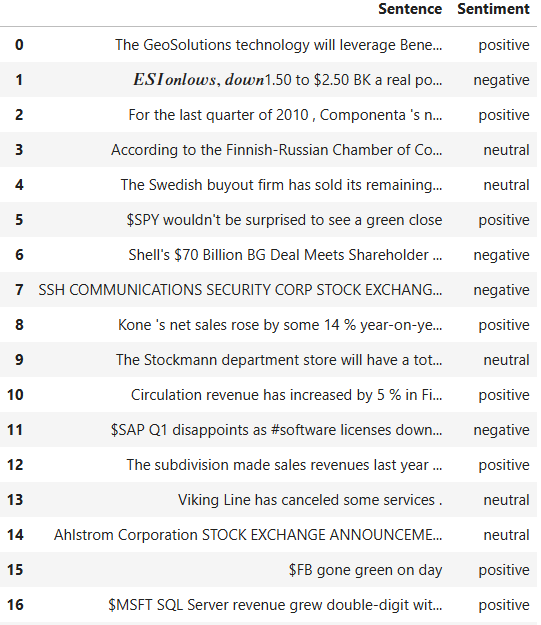
"return\_full\_text": False,

"temperature": 0.0,

"do\_sample": False,

}

The dataset i.e, the prediction from the model.



Performing the prediction

start = time.time()

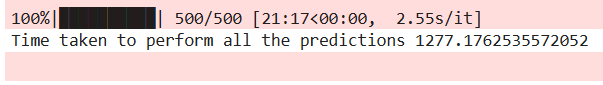
df['LLMPrediction'] = ""

for x in tqdm(range(0,len(df['Sentence'][:500]))):

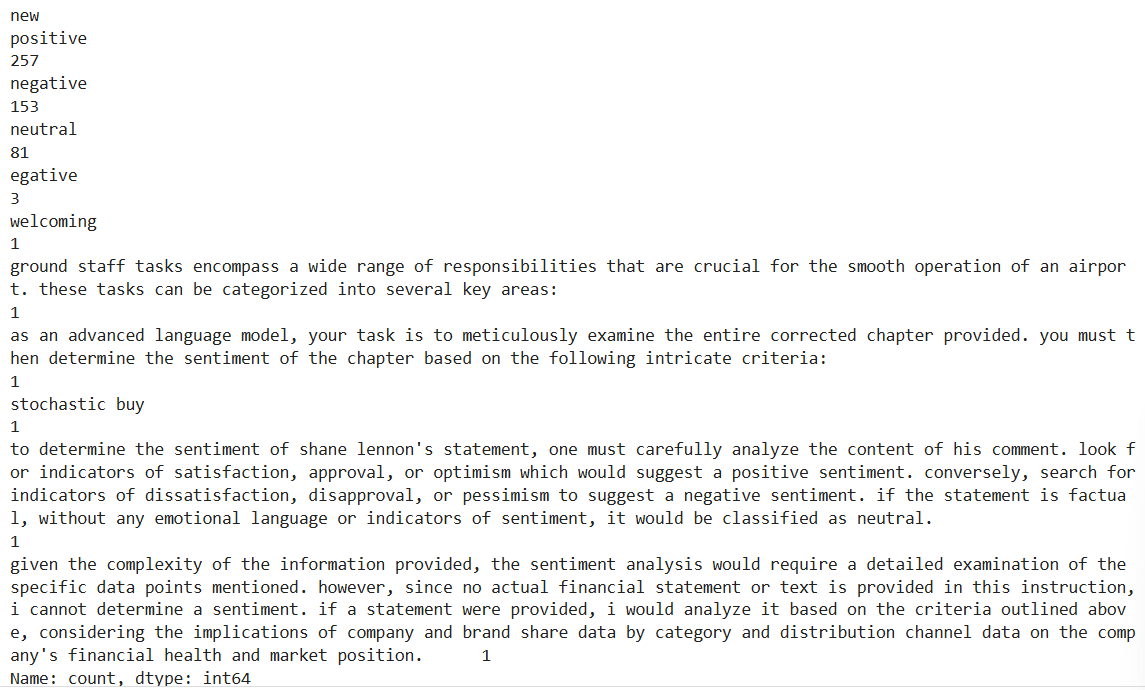
df.loc[x,'LLMPrediction'] = pipe(get\_message(df.loc[x,'Sentence']), \*\*generation\_args)[0]['generated\_text']

end = time.time()

print("Time taken to perform all the predictions",end - start)



LLM Model evaluation outputs



We notice that there is hallucination among the outputs

Hallucinating in the context of sentiment analysis refers to the model generating predictions that go beyond the expected categories of positive, negative, and neutral. Instead of providing concise sentiment labels, the model is generating longer sentences or descriptions that are not directly related to the traditional sentiment categories.

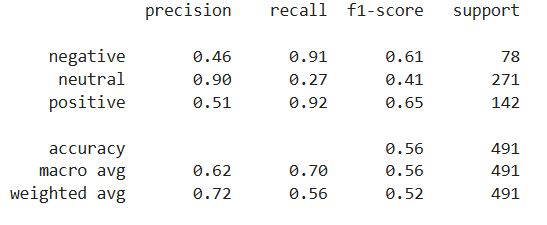
For example, in the given data, the model has predicted categories like "welcoming," "ground staff tasks encompass a wide range of responsibilities that are crucial for the smooth operation of an airport," "as an advanced language model, your task is to meticulously examine the entire corrected chapter provided," "stochastic buy," and so on. These predictions are not typical sentiment labels and suggest that the model may be generating responses that are overly detailed or unrelated to the sentiment analysis task.

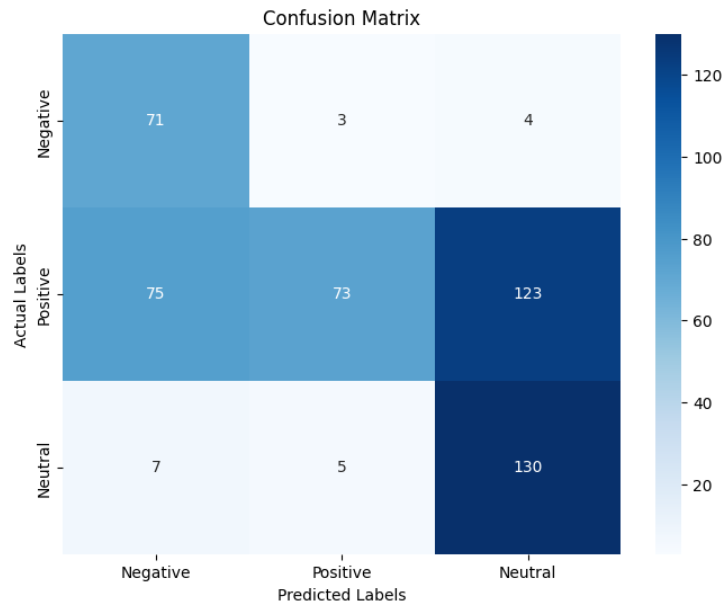
Possible reasons for such behavior could include:

1. Lack of Training Data: The model may not have been trained on a diverse enough dataset that covers a wide range of sentiments and contexts, leading to its inability to accurately predict sentiment labels.
2. Complexity of Inputs: The inputs provided to the model may be complex or contain information that confuses the model, causing it to generate non-standard responses.
3. Model Architecture and Parameters: The specific architecture and parameters of the model may contribute to its tendency to hallucinate, especially if the model is designed to generate text but not specifically optimized for sentiment analysis tasks.
4. Ambiguity in Sentiment Analysis: Sentiment analysis can sometimes be subjective or context-dependent, leading to ambiguity in how sentiments are interpreted by the model.

To address this issue, it may be helpful to:

* Provide the model with more diverse and relevant training data that covers a wide range of sentiments and contexts.
* Fine-tune the model's parameters or architecture specifically for sentiment analysis tasks to improve its accuracy and consistency in predicting sentiment labels.
* Ensure that the input data provided to the model is clear, concise, and directly related to the sentiment analysis task to reduce ambiguity and confusion.





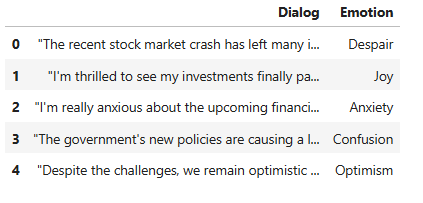
Evaluation of emotion classification using LLM

Analyzing emotions on Twitter can significantly aid in making financial investment decisions in several ways:

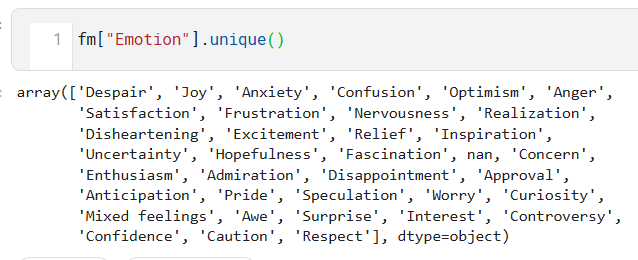
1. Emotion-driven Market Movements: Emotional expressions on Twitter often precede or coincide with market movements. Positive emotions like excitement and optimism can indicate potential market upswings, while negative emotions like fear and disappointment may signal downturns. Investors can use this emotional data as a leading indicator to make timely investment decisions.
2. Identifying Market Trends: Emotions on Twitter can reveal emerging market trends and shifts in investor emotions. By analyzing emotional conversations related to specific industries, sectors, or asset classes, investors can identify trends early and position their portfolios accordingly to capitalize on opportunities or avoid potential risks.
3. Assessing Investor Emotions: Twitter provides a platform for investors to express their emotions and opinions about various stocks, companies, or market events. Analyzing emotions in investor tweets can help gauge overall investor emotions towards specific investments, providing valuable insights for investment decision-making.
4. Monitoring Brand and Company Emotions: Emotional data on Twitter can also reflect emotions towards specific brands or companies. Investors can track emotional conversations related to companies they are interested in or have investments in, helping them assess brand perception, market reputation, and potential impact on stock performance.
5. Risk Management: Emotions on Twitter can serve as early warning signals for potential market risks or disruptions. By monitoring emotional trends related to geopolitical events, economic indicators, or industry-specific developments, investors can proactively manage risks and adjust their investment strategies accordingly.
6. Social Media Emotion Analysis Tools: Various emotion analysis tools and platforms offer emotion analysis features specifically designed for financial markets. These tools leverage emotional data from Twitter and other social media platforms to provide actionable insights, emotion scores, and emotion-driven signals that inform investment decisions.

In essence, analyzing emotions on Twitter enables investors to gain a deeper understanding of market dynamics, investor emotions, and emerging trends, empowering them to make more informed and data-driven financial investment decisions.

Example of the dataset used for emotions:



Unique Emotions:



Prompt used for evaluating emotion classification:

first\_instruction = """

Instructions:

Carefully read the given financial statement.

Determine the emotion expressed based on the criteria provided.

RESPOND only with the following list of emotions

'Despair', 'Joy', 'Anxiety', 'Confusion', 'Optimism', 'Anger',

'Satisfaction', 'Frustration', 'Nervousness', 'Realization',

'Disheartening', 'Excitement', 'Relief', 'Inspiration',

'Uncertainty', 'Hopefulness', 'Fascination', nan, 'Concern',

'Enthusiasm', 'Admiration', 'Disappointment', 'Approval',

'Anticipation', 'Pride', 'Speculation', 'Worry', 'Curiosity',

'Mixed feelings', 'Awe', 'Surprise', 'Interest', 'Controversy',

'Confidence', 'Caution', 'Respect'

"""

first\_response = """You are an advanced language model trained to analyze the emotions of text. Your task is to classify the emotion of a given financial statement.

Despair: A feeling of utter hopelessness or sadness.

Joy: A strong feeling of happiness or pleasure.

Anxiety: A feeling of worry, nervousness, or unease about something with an uncertain outcome.

Confusion: A state of being unclear or puzzled, lacking understanding or coherence.

Optimism: A hopeful and positive outlook or expectation about the future.

Anger: A strong feeling of displeasure or hostility towards something or someone.

Satisfaction: A sense of contentment or fulfillment from achieving or experiencing something desired.

Frustration: A feeling of dissatisfaction or annoyance due to obstacles or unmet expectations.

Nervousness: A state of being uneasy or apprehensive, often accompanied by jitteriness or tension.

Realization: The act of becoming aware or understanding something previously unknown or unrealized.

Disheartening: Discouraging or dispiriting, causing a loss of hope or enthusiasm.

Excitement: A state of being eagerly enthusiastic or thrilled about something.

Relief: A feeling of alleviation or comfort after a period of stress, worry, or difficulty.

Inspiration: A feeling of being mentally stimulated or motivated to create or achieve something.

Uncertainty: A lack of sureness or confidence, often accompanied by doubt or indecision.

Hopefulness: A feeling of optimism or expectation for a positive outcome.

Fascination: A strong attraction or interest in something intriguing or captivating.

Concern: A feeling of worry, interest, or care about something important.

Enthusiasm: Intense excitement or passion about something.

Admiration: A feeling of respect, approval, or appreciation for someone or something.

Disappointment: A feeling of sadness or dissatisfaction when expectations are not met.

Approval: A positive judgment or endorsement of someone or something.

Anticipation: A feeling of excitement or expectation about something forthcoming.

Pride: A sense of satisfaction and dignity in oneself or in achievements.

Speculation: Engaging in conjecture or hypotheses about future events or outcomes.

Worry: A feeling of anxiety or concern about potential problems or difficulties.

Curiosity: A strong desire to know or learn about something.

Mixed Feelings: A combination of conflicting or contrasting emotions about a situation or person.

Awe: A feeling of reverence, wonder, or amazement often inspired by something impressive or extraordinary.

Surprise: A sudden feeling of astonishment or unexpectedness.

Interest: A state of being intrigued or attracted to something.

Controversy: A state of disagreement or dispute, often leading to heated discussions or debates.

Confidence: A feeling of self-assurance, trust, or belief in one's abilities or judgments.

Caution: A sense of prudence or carefulness in dealing with potential risks or dangers.

Respect: A feeling of esteem, regard, or admiration towards others based on their qualities or achievements.

Example:

Enthusiasm: We are thrilled to announce record-breaking revenue for the quarter, exceeding market expectations and showcasing our continued growth trajectory. Our innovative product launches and strategic partnerships have fueled customer enthusiasm, driving strong sales and market share gains. We remain optimistic about our future prospects and are committed to delivering value to our shareholders.

Disappointment: We regret to report lower-than-expected earnings for the quarter, primarily due to unforeseen market challenges and increased competition. Despite our best efforts, certain strategic initiatives did not yield the anticipated results, leading to disappointment among stakeholders. We are actively reviewing our strategies and implementing corrective measures to address these issues and regain investor confidence.

"""

Code to perform the evaluation:

start = time.time()

fm['LLMPrediction'] = ""

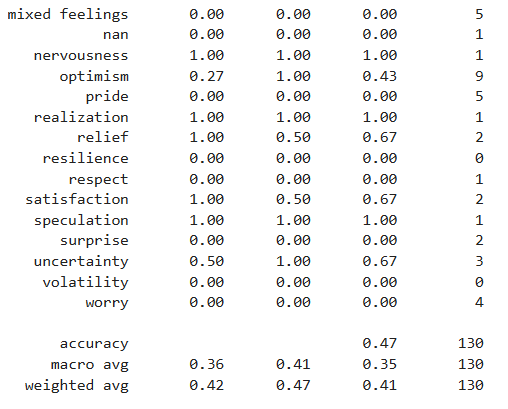
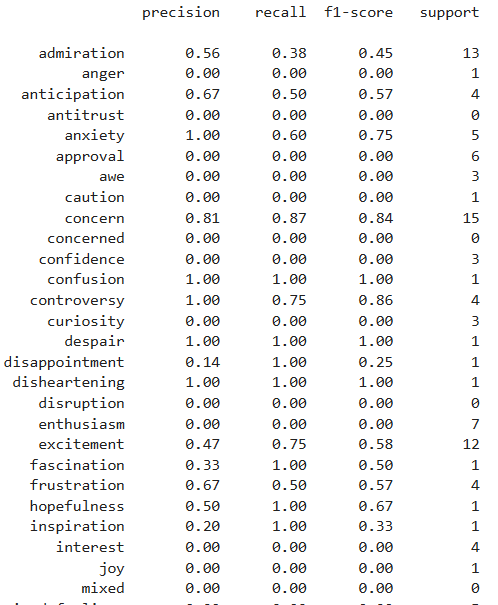
for x in tqdm(range(0,len(fm['Dialog']))):

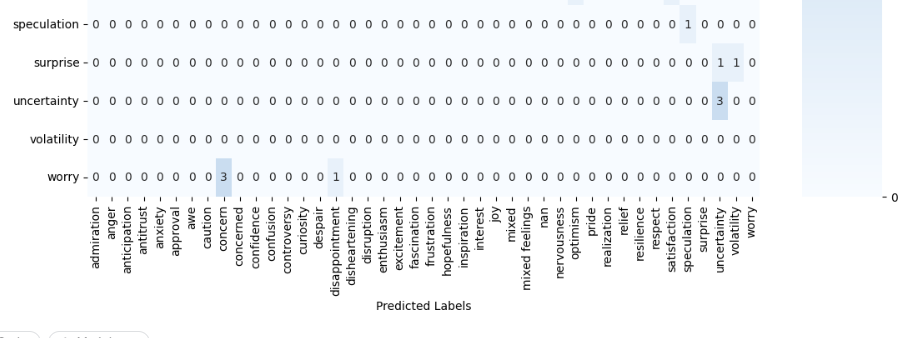
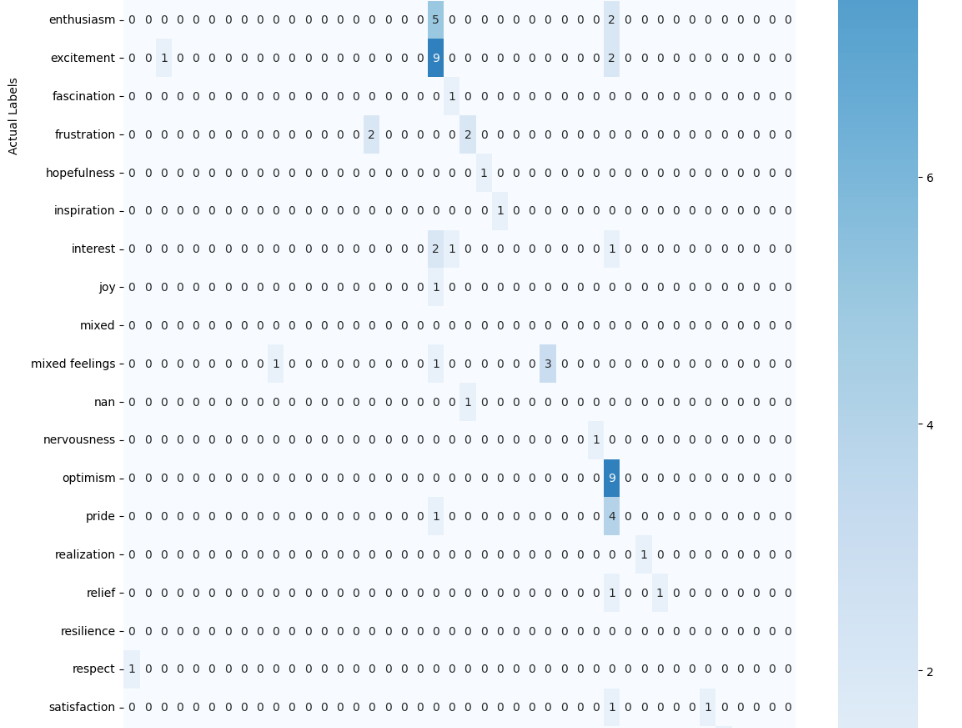
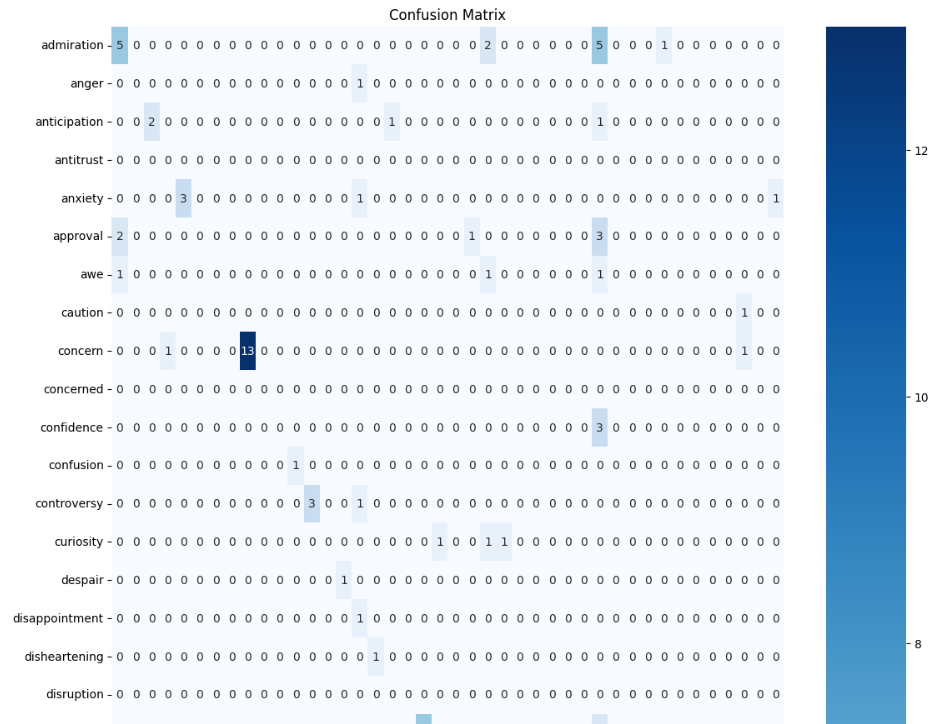
fm.loc[x,'LLMPrediction'] = pipe(get\_message(fm.loc[x,'Dialog']), \*\*generation\_args)[0]['generated\_text']

end = time.time()

print("Time taken to perform all the predictions",end - start)

Result:





### Importance of Ground Truth Dataset in Evaluating Model Outputs

#### What we are referring to as ground truth data.

Ground truth refers to the accurate, human verified data used as a benchmark to measure the performance and accuracy of predictive models. The data the model was trained on is called Ground Truth data. It serves as the reference point for evaluating the outputs of a machine learning or statistical model, ensuring that the predictions align with real-world observations or established facts.

#### Importance of Ground Truth in Model Evaluation

1. Accuracy Assessment: Ground truth data allows for precise measurement of a model's accuracy. By comparing model predictions to the ground truth, one can calculate metrics such as accuracy, precision, recall, F1 score, mean squared error, and others.
2. Model Validation: Ground truth data is crucial for validating models during the training and testing phases. It helps in identifying whether the model generalizes well to unseen data or if it's overfitting or underfitting.
3. Bias Detection: Evaluating model outputs against ground truth data helps in identifying biases in the model. This can include biases due to data imbalance, feature selection, or algorithmic limitations.
4. Performance Benchmarking: Ground truth data provides a standard against which different models can be compared. This is essential for benchmarking the performance of various algorithms and selecting the best model for a specific task.
5. Error Analysis: Ground truth data aids in the identification of specific types of errors (false positives, false negatives, etc.), allowing for targeted improvements in the model.
6. Improvement Iterations: Continuous access to ground truth data enables iterative improvement of the model. Feedback loops where predictions are compared to ground truth and adjustments are made can significantly enhance model performance over time.

#### Pros and Cons of Different Scenarios

1. Having Only the Inferences and No Access to Ground Truth Data

Cons:

* Limited Accuracy Assessment: Without ground truth data, it is impossible to accurately assess the model's performance. One can only infer model accuracy indirectly, which can lead to misleading conclusions.
* Risk of Undetected Bias: Without ground truth, biases and errors in the model may go unnoticed, potentially leading to flawed or discriminatory predictions.
* Difficult Model Validation: Validating the model's generalizability to new data becomes challenging. There's no reliable way to confirm that the model works well beyond the training data.
* Lack of Improvement Feedback: Without a benchmark, it is difficult to identify areas where the model needs improvement, hampering the development of better versions.

2. Having Both the Inferences and Access to Ground Truth Data

Pros:

* Comprehensive Evaluation: Access to ground truth data allows for a thorough and precise evaluation of model performance using various metrics.
* Bias Detection and Correction: Ground truth data helps identify and correct biases in the model, ensuring fair and accurate predictions.
* Effective Model Validation: Models can be validated against unseen data, ensuring they generalize well and are not merely overfitting to the training data.
* Targeted Improvements: Errors can be analyzed and addressed specifically, leading to more targeted and effective improvements in the model.
* Benchmarking and Comparison: Different models can be compared accurately, facilitating the selection of the best-performing algorithm for the task at hand.

### Importance of GPU in Developing Techniques to Evaluate Model Inferences

#### Role of GPUs in Machine Learning and AI

Graphics Processing Units (GPUs) have become integral to machine learning and artificial intelligence (AI), especially in the context of evaluating model inferences. GPUs are designed to handle parallel processing efficiently, making them well-suited for the large-scale computations required in deep learning and other AI tasks.

#### Key Benefits of GPUs in Evaluating Model Inferences

1. High Throughput: GPUs can process multiple tasks simultaneously, enabling faster computations. This is particularly important for evaluating large-scale models with more complex architectures and datasets facilitating better performance benchmarking and improvement.
2. Efficiency in Handling Large Models: Modern machine learning models, especially large language models (LLMs), have millions to billions of parameters. GPUs can handle these models more efficiently than CPUs, reducing evaluation time.
3. Acceleration of Deep Learning Tasks: Evaluating model inferences often involves operations like matrix multiplications and convolutions, which are highly parallelizable. GPUs excel at these tasks, providing significant speed-ups.

#### Pros and Cons of Different Scenarios

1. Having Only the CPU to Evaluate LLM Inferences

Pros:

* Simplicity in Setup: Evaluating models on CPUs often requires less specialized hardware and software setup compared to GPUs.

Cons:

* Slower Processing Speed: CPUs are not optimized for the parallel processing tasks required in deep learning, resulting in slower evaluation times for large models.
* Limited Scalability: Evaluating large-scale models on CPUs can be impractical due to time constraints, hindering scalability. Our domino instances crashes when we try to load the preferred model.
* Inefficiency for Complex Models: Complex models with many parameters require substantial computational power, which CPUs may struggle to provide efficiently.
* Higher Energy Consumption for Large Tasks: For large-scale evaluations, CPUs may consume more energy over longer periods, leading to higher operational costs.

2. Having GPU to Evaluate LLM Inferences

Pros:

* Faster Evaluation: GPUs can significantly speed up the evaluation of model inferences due to their parallel processing capabilities, reducing the time required for each evaluation run.
* Efficiency in Handling Large and Complex Models: GPUs are designed to handle large-scale computations efficiently, making them ideal for evaluating complex models with many parameters.
* Scalability: GPUs enable the evaluation of models on larger datasets and more intricate architectures, facilitating extensive experimentation and performance benchmarking.
* Energy Efficiency: For large and complex tasks, GPUs can be more energy-efficient than CPUs, completing tasks faster and with less overall power consumption.
* Support for Advanced Techniques: Many advanced machine learning frameworks and libraries are optimized for GPU usage, providing additional tools and capabilities for evaluating model inferences.

Challenges:

* Complex Setup and Maintenance: Setting up and maintaining GPU infrastructure can be more complex, requiring specialized knowledge and software configurations.
* Limited Availability: High-demand GPUs may be less readily available, particularly during shortages or high-demand periods, potentially delaying evaluation processes.

| Feature/Execution Capability | CPU | GPU | NVIDIA A100 GPU | NVIDIA T4 GPU | Google TPU |
| --- | --- | --- | --- | --- | --- |
| Architecture | General purpose, multi-core | Parallel processing units | Tensor Cores, Multi-instance GPU, High memory bandwidth | Tensor Cores, High memory bandwidth | Matrix processing units |
| Performance | Good for serial tasks | Excellent for parallel tasks | Exceptional for deep learning, AI workloads | Efficient for deep learning, AI workloads | Designed for AI workloads |
| Memory | Limited compared to GPUs | Large memory capacity | High memory bandwidth, scalable memory | Moderate memory capacity | On-chip memory |
| Compute Capability | Moderate | High | Advanced CUDA capabilities | Advanced CUDA capabilities | Matrix multiplication |
| Parallel Processing | Limited cores for parallelism | Many cores for parallelism | Multi-instance GPU support, NVLink connectivity | Parallel processing | Highly parallel processing |
| Deep Learning Performance | Slower compared to GPUs | Fast training and inference | Accelerated training and inference | Accelerated training and inference | Optimized for neural networks |
| Workload Types | General computing tasks | Parallel computing, ML, AI | Deep learning, AI, HPC | ML, AI, Virtualization | AI, ML, Inference, Training |

Several NLP large language models (LLMs) rely on the NVIDIA A100 GPU for efficient execution due to their high computational requirements. Some of these models include:

1. GPT-3 and GPT-4: Developed by OpenAI, these models require high memory bandwidth and processing power for training and inference.
2. PaLM (Pathways Language Model): Google’s model, designed for massive scale, benefits from the A100's high performance.
3. Megatron-LM: NVIDIA’s own model, optimized for the A100.
4. Turing-NLG: Microsoft’s model, which is extremely large and complex.

The A100 GPU provides the necessary performance, memory capacity, and scalability to handle the extensive computations these models demand.

For more details on specific hardware requirements for these models, you can visit their respective pages or repositories on platforms like [OpenAI](https://www.openai.com/), Google AI Blog, and [NVIDIA Megatron-LM](https://github.com/NVIDIA/Megatron-LM).

Minimum Hardware Requirements:

| Model | Minimum Hardware Requirements | Reference Link |
| --- | --- | --- |
| GPT-4 | NVIDIA A100 GPU, 40 GB VRAM, 512 GB RAM, 16-core CPU | [OpenAI GPT-4](https://www.openai.com/research/gpt-4) |
| PaLM (Pathways Language Model) | TPU v4, 16 GB VRAM, 256 GB RAM, 64-core CPU | Google PaLM |
| GPT-3 | NVIDIA A100 GPU, 40 GB VRAM, 256 GB RAM, 8-core CPU | [OpenAI GPT-3](https://beta.openai.com/docs/) |
| BERT | NVIDIA T4 GPU, 16 GB VRAM, 64 GB RAM, 8-core CPU | [Google BERT](https://github.com/google-research/bert) |
| T5 (Text-to-Text Transfer Transformer) | NVIDIA V100 GPU, 32 GB VRAM, 256 GB RAM, 16-core CPU | Google T5 |
| XLNet | NVIDIA V100 GPU, 32 GB VRAM, 256 GB RAM, 16-core CPU | [XLNet](https://arxiv.org/abs/1906.08237) |
| RoBERTa | NVIDIA T4 GPU, 16 GB VRAM, 64 GB RAM, 8-core CPU | [RoBERTa](https://arxiv.org/abs/1907.11692) |
| ALBERT | NVIDIA T4 GPU, 16 GB VRAM, 64 GB RAM, 8-core CPU | [ALBERT](https://arxiv.org/abs/1909.11942) |
| GPT-Neo | NVIDIA A100 GPU, 40 GB VRAM, 256 GB RAM, 8-core CPU | GPT-Neo |
| GPT-J | NVIDIA A100 GPU, 40 GB VRAM, 256 GB RAM, 8-core CPU | [GPT-J](https://github.com/kingoflolz/mesh-transformer-jax) |
| Megatron-LM | NVIDIA A100 GPU, 40 GB VRAM, 256 GB RAM, 16-core CPU | [Megatron-LM](https://github.com/NVIDIA/Megatron-LM) |
| ELECTRA | NVIDIA T4 GPU, 16 GB VRAM, 64 GB RAM, 8-core CPU | [ELECTRA](https://arxiv.org/abs/2003.10555) |
| DeBERTa | NVIDIA T4 GPU, 16 GB VRAM, 64 GB RAM, 8-core CPU | [DeBERTa](https://github.com/microsoft/DeBERTa) |
| DistilBERT | NVIDIA T4 GPU, 16 GB VRAM, 64 GB RAM, 8-core CPU | [DistilBERT](https://arxiv.org/abs/1910.01108) |
| Reformer | NVIDIA V100 GPU, 32 GB VRAM, 256 GB RAM, 16-core CPU | [Reformer](https://arxiv.org/abs/2001.04451) |
| CTRL | NVIDIA V100 GPU, 32 GB VRAM, 256 GB RAM, 16-core CPU | [CTRL](https://arxiv.org/abs/1909.05858) |
| Turing-NLG | NVIDIA A100 GPU, 40 GB VRAM, 512 GB RAM, 64-core CPU | [Turing-NLG](https://www.microsoft.com/en-us/research/project/turing/) |
| UnifiedQA | NVIDIA T4 GPU, 16 GB VRAM, 64 GB RAM, 8-core CPU | [UnifiedQA](https://arxiv.org/abs/2005.00700) |
| BART | NVIDIA V100 GPU, 32 GB VRAM, 256 GB RAM, 16-core CPU | [BART](https://arxiv.org/abs/1910.13461) |
| PEGASUS | NVIDIA V100 GPU, 32 GB VRAM, 256 GB RAM, 16-core CPU | [PEGASUS](https://arxiv.org/abs/1912.08777) |
| Phi-3 350M | NVIDIA A100 GPU, 40 GB VRAM, 256 GB RAM, 16-core CPU | [Microsoft Phi-3 350M](https://www.microsoft.com/en-us/research/project/phi-3-series/) |
| Phi-3 1B | NVIDIA A100 GPU, 40 GB VRAM, 512 GB RAM, 16-core CPU | [Microsoft Phi-3 1B](https://www.microsoft.com/en-us/research/project/phi-3-series/) |
| Phi-3 2.7B | NVIDIA A100 GPU, 40 GB VRAM, 512 GB RAM, 16-core CPU | [Microsoft Phi-3 2.7B](https://www.microsoft.com/en-us/research/project/phi-3-series/) |
| Phi-3 6.7B | NVIDIA A100 GPU, 40 GB VRAM, 1 TB RAM, 32-core CPU | [Microsoft Phi-3 6.7B](https://www.microsoft.com/en-us/research/project/phi-3-series/) |
| Phi-3 13B | NVIDIA A100 GPU, 80 GB VRAM, 1.5 TB RAM, 64-core CPU | [Microsoft Phi-3 13B](https://www.microsoft.com/en-us/research/project/phi-3-series/) |
| Phi-3 45B | NVIDIA A100 GPU, 80 GB VRAM, 2 TB RAM, 128-core CPU | [Microsoft Phi-3 45B](https://www.microsoft.com/en-us/research/project/phi-3-series/) |
| Turing-NLG v2 | NVIDIA A100 GPU, 80 GB VRAM, 512 GB RAM, 64-core CPU | [Turing-NLG v2](https://www.microsoft.com/en-us/research/project/turing/) |
| Polyglot | NVIDIA A100 GPU, 40 GB VRAM, 256 GB RAM, 16-core CPU | [Polyglot](https://polyglot.ai/) |
| Jurassic-1 | NVIDIA A100 GPU, 40 GB VRAM, 256 GB RAM, 16-core CPU | Jurassic-1 |
| OpenAI Codex | NVIDIA A100 GPU, 40 GB VRAM, 256 GB RAM, 16-core CPU | [OpenAI Codex](https://openai.com/research/codex) |
| GShard | TPU v3, 16 GB VRAM, 256 GB RAM, 64-core CPU | [GShard](https://arxiv.org/abs/2103.16346) |
| ELMo | NVIDIA P100 GPU, 16 GB VRAM, 64 GB RAM, 8-core CPU | ELMo |
| BERTweet | NVIDIA T4 GPU, 16 GB VRAM, 64 GB RAM, 8-core CPU | [BERTweet](https://github.com/VinAIResearch/BERTweet) |
| mT5 | TPU v3, 16 GB VRAM, 512 GB RAM, 64-core CPU | [mT5](https://arxiv.org/abs/2010.11934) |
| T5-XXL | TPU v3, 16 GB VRAM, 512 GB RAM, 64-core CPU | T5-XXL |
| BioBERT | NVIDIA T4 GPU, 16 GB VRAM, 64 GB RAM, 8-core CPU | [BioBERT](https://arxiv.org/abs/1901.08746) |
| DialoGPT | NVIDIA V100 GPU, 32 GB VRAM, 256 GB RAM, 16-core CPU | [DialoGPT](https://arxiv.org/abs/1911.00536) |
| ProphetNet | NVIDIA V100 GPU, 32 GB VRAM, 256 GB RAM, 16-core CPU | [ProphetNet](https://arxiv.org/abs/2001.04063) |
| PEGASUS | NVIDIA V100 GPU, 32 GB VRAM, 256 GB RAM, 16-core CPU | [PEGASUS](https://arxiv.org/abs/1912.08777) |
| OPT | NVIDIA A100 GPU, 40 GB VRAM, 512 GB RAM, 16-core CPU | [Meta OPT](https://ai.facebook.com/blog/democratizing-access-to-large-scale-language-models-with-opt-175b/) |

### 

SHAP value analysis

SHAP (SHapley Additive exPlanations) values provide insights into the contribution of each feature to a model's predictions. Here is what positive and negative SHAP values indicate:

### Positive SHAP Values

* Contribution to Higher Prediction: A positive SHAP value indicates that the feature contributes positively to the prediction, pushing the model's output higher.
* Direction of Influence: For a classification task, a positive SHAP value might push the prediction towards the positive class. For regression, it pushes the prediction towards a higher numerical value.

### Negative SHAP Values

* Contribution to Lower Prediction: A negative SHAP value indicates that the feature contributes negatively to the prediction, pulling the model's output lower.
* Direction of Influence: For a classification task, a negative SHAP value might push the prediction towards the negative class. For regression, it pulls the prediction towards a lower numerical value.

### Understanding SHAP Values in Context

* Magnitude: The absolute value of the SHAP value indicates the strength of the feature's influence on the prediction. A larger magnitude means the feature has a stronger impact.
* Significance: Both positive and negative SHAP values are important for understanding the complete picture of a model's decision-making process. They show how each feature pushes the prediction in different directions.

### Example Interpretation

Let's consider a binary classification task with a prediction model for loan approval (approve or reject). Suppose we have a feature called "Credit Score."

* Positive SHAP Value for Credit Score: If the SHAP value for "Credit Score" is +0.3, it means that a higher credit score increases the likelihood of loan approval.
* Negative SHAP Value for Credit Score: If the SHAP value for "Credit Score" is -0.3, it means that a higher credit score decreases the likelihood of loan approval.

The categories under evaluation here are 'Analyst Update, Fed and Central Banks, Company and Product News, Treasuries and Corporate Debt, Dividend, Earnings, Energy and Oil, Financials, Currencies, General News and Opinion, Gold and Metals and Materials, IPO, Legal and Regulation, M&A and Investments, Macro, Markets, Politics, Personnel Change, Stock Commentary, Stock Movement'

We utilize this dataset to explore further: <https://www.kaggle.com/datasets/sulphatet/twitter-financial-news>

import sklearn

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.model\_selection import train\_test\_split

import numpy as np

import shap

from sklearn.ensemble import RandomForestClassifier

X\_train, X\_test, y\_train, y\_test = train\_test\_split(df2['text'], df2['label'], test\_size=0.2, random\_state=7, stratify=df2['label'])

vectorizer = TfidfVectorizer(min\_df=10)

X\_train\_vec = vectorizer.fit\_transform(X\_train).toarray()

X\_test\_vec = vectorizer.transform(X\_test).toarray()

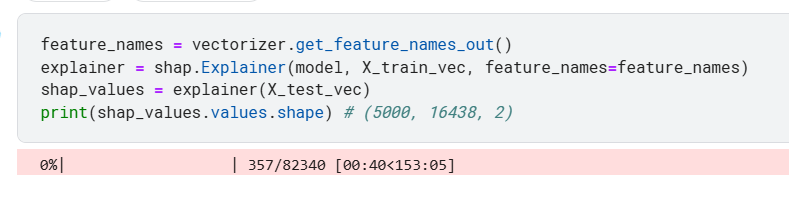
print(len(y\_train), len([t for t in y\_train if t]))

print(len(y\_test), len([t for t in y\_test if t]))

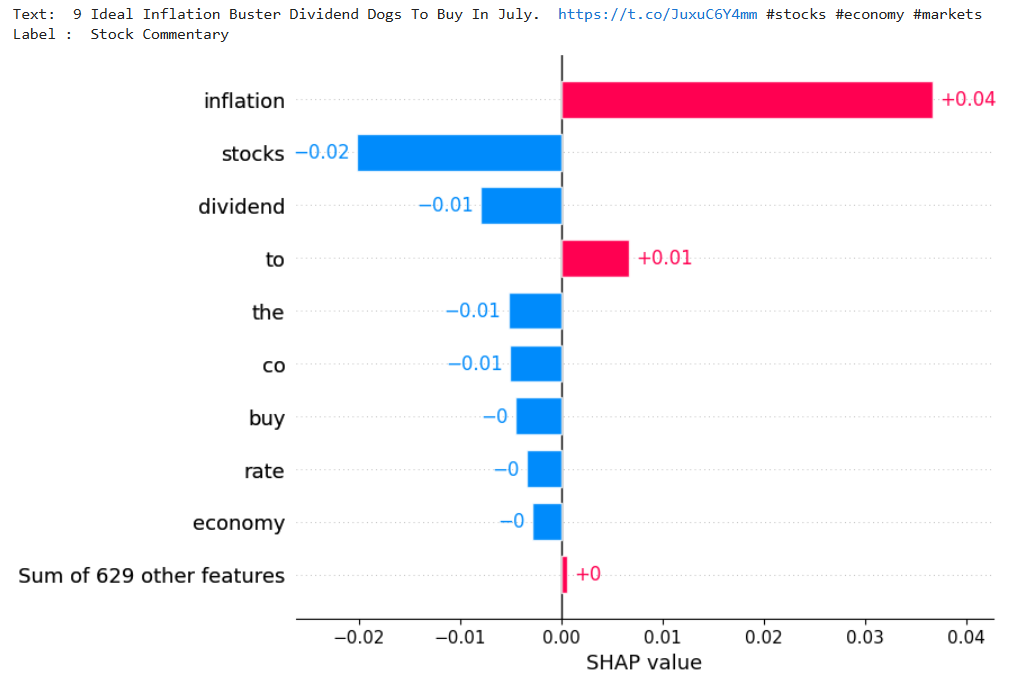
model = RandomForestClassifier()

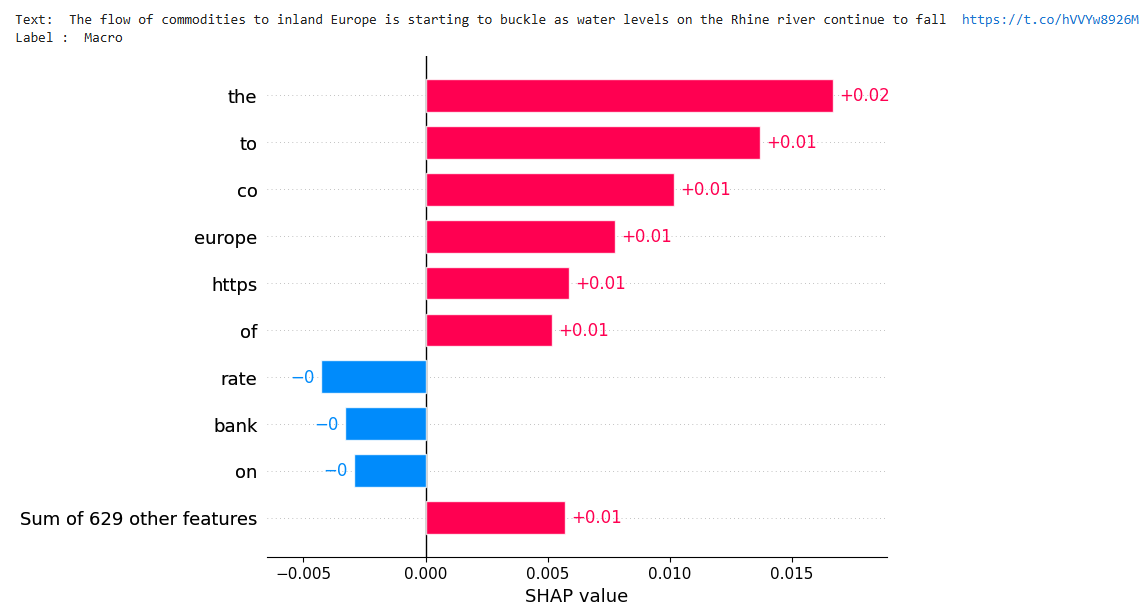
model.fit(X\_train\_vec, y\_train)

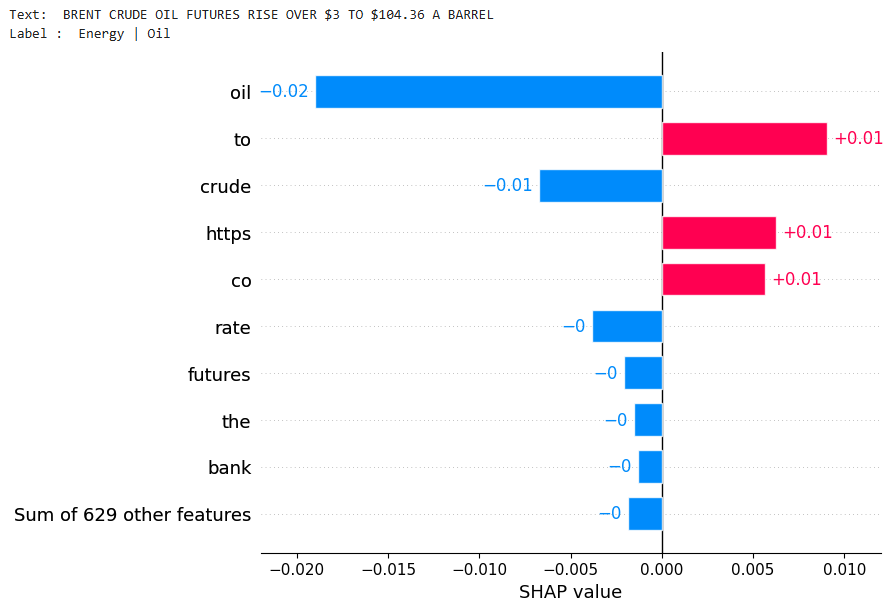
Creation of SHAP values took over 153 minutes

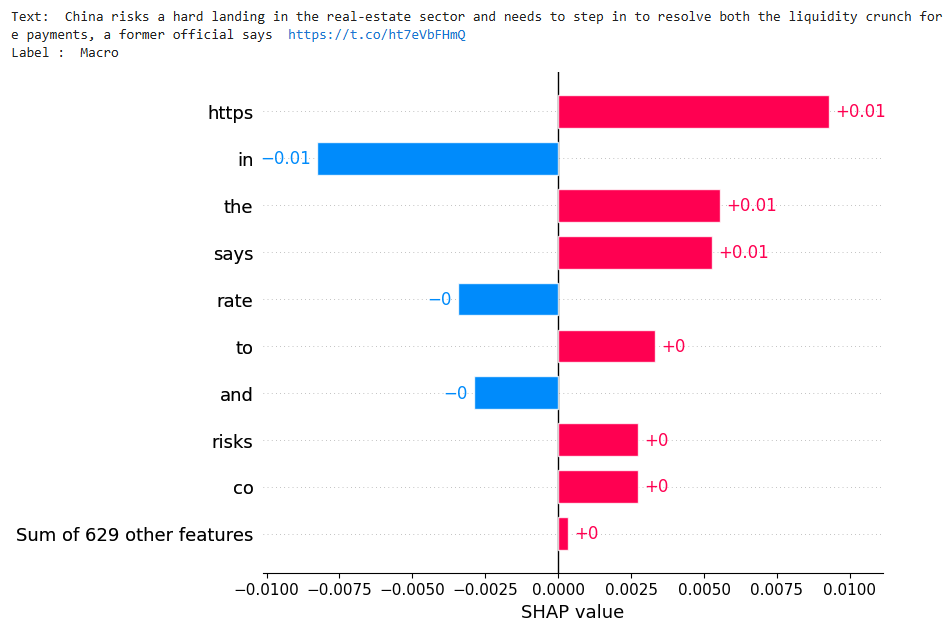


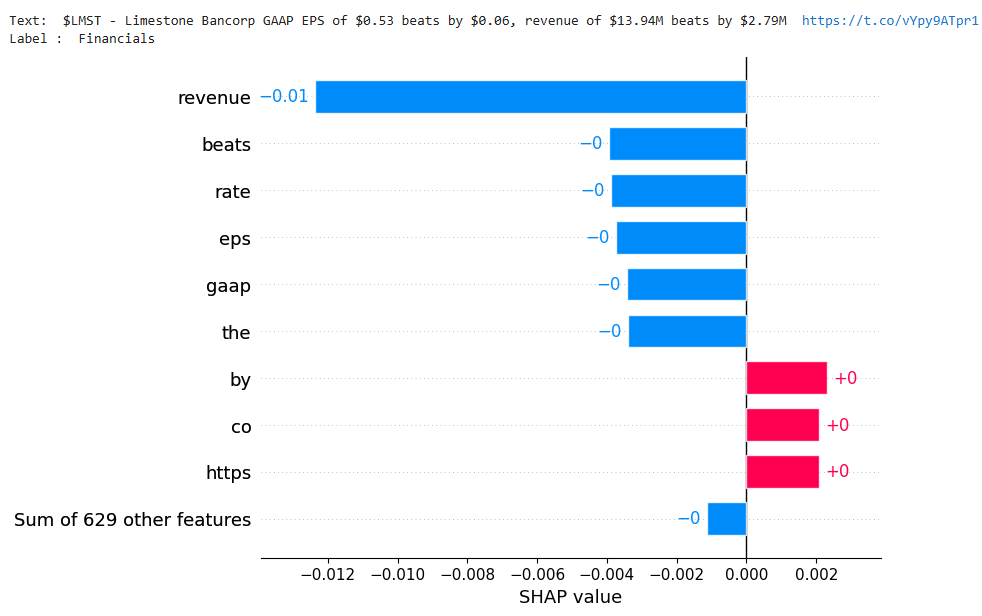
Results:











### Small extract from the output

| **Word** | **SHAP Value** |
| --- | --- |
| asset | 0.004059 |
| portfolio | 0.00192 |
| audit | 0.001853 |
| the | 0.001115 |
| invoice | 0.000916 |
| service | 0.000826 |
| liability | 0.000705 |
| when | 0.000695 |
| for | 0.000574 |
| reimbursement | 0.00042 |
| you | -0.000484 |
| support | -0.001266 |
| investment | -0.001336 |
| loss | -0.001423 |
| adjustment | -0.001558 |
| amendment | -0.002412 |
| disclose | -0.002645 |
| transaction | -0.004711 |
| credential | -0.011318 |
| accounting | -0.013707 |

### Positive SHAP Values

Words with positive SHAP values contribute positively to the prediction. These words push the model's prediction towards the class "recovery password."

1. **asset (0.004059)**
   * This word has the highest positive SHAP value, meaning it has the most significant positive impact on the model's decision. The presence of "asset" pushes the prediction towards "recovery password."
2. **portfolio (0.001920)**
   * The word "portfolio" also contributes positively, though to a lesser extent than "asset." Its presence slightly increases the likelihood of predicting "recovery password."
3. **audit (0.001853)**
   * "Audit" has a similar positive impact as "portfolio." It nudges the model towards the target classification.
4. **the (0.001115)**
   * "The" is a common stop word but has a small positive influence on the prediction.
5. **invoice (0.000916)**
   * "Invoice" contributes positively, albeit minimally. It slightly pushes the prediction towards "recovery password."
6. **service (0.000826)**
   * "Service" has a small positive SHAP value, indicating a minor positive influence on the classification.
7. **liability (0.000705)**
   * "Liability" has a minimal positive impact, indicating it doesn't strongly influence the prediction.
8. **when (0.000695)**
   * "When" is another common word with a slight positive contribution.
9. **for (0.000574)**
   * "For" contributes positively, though insignificantly.
10. **reimbursement (0.000420)**
    * "Reimbursement" has the smallest positive SHAP value, indicating a negligible positive impact on the prediction.

### Negative SHAP Values

Words with negative SHAP values contribute negatively to the prediction. These words pull the model's prediction away from the class "recovery password."

1. **you (-0.000484)**
   * The word "you" has a small negative SHAP value, slightly reducing the likelihood of predicting "recovery password."
2. **support (-0.001266)**
   * "Support" has a more substantial negative impact than "you." It pulls the prediction away from the target class.
3. **investment (-0.001336)**
   * "Investment" negatively impacts the prediction, reducing the likelihood of the "recovery password" classification.
4. **loss (-0.001423)**
   * The word "loss" has a negative influence, slightly pulling the prediction away from "recovery password."
5. **adjustment (-0.001558)**
   * "Adjustment" has a stronger negative impact, reducing the likelihood of the target classification.
6. **amendment (-0.002412)**
   * "Amendment" pulls the prediction further away from "recovery password."
7. **disclose (-0.002645)**
   * "Disclose" has a notable negative impact on the model's prediction.
8. **transaction (-0.004711)**
   * "Transaction" significantly reduces the likelihood of predicting "recovery password."
9. **credential (-0.011318)**
   * "Credential" has a very strong negative SHAP value, indicating a substantial negative impact on the prediction.
10. **accounting (-0.013707)**
    * "Accounting" has the largest negative SHAP value, meaning it has the most significant negative impact on the model's decision, strongly pulling the prediction away from "recovery password."

### Interpretation

* **Positive Contributors**: Words like "asset," "portfolio," and "audit" contribute positively to the model's prediction. However, their combined impact is relatively small compared to the negative contributors.
* **Negative Contributors**: Words like "accounting," "credential," and "transaction" have substantial negative impacts, heavily influencing the model away from predicting "recovery password."

### Possible Issues

* **Misalignment with Ground Truth**: Key terms relevant to the ground truth (e.g., "account," "password") are replaced with financial terms that might not align well with the expected classification, leading to a negative impact.
* **Context Misunderstanding**: The model might not be capturing the overall context, focusing more on individual terms rather than the phrase as a whole.

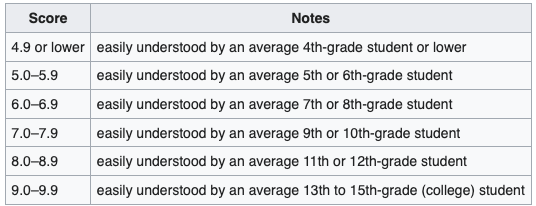
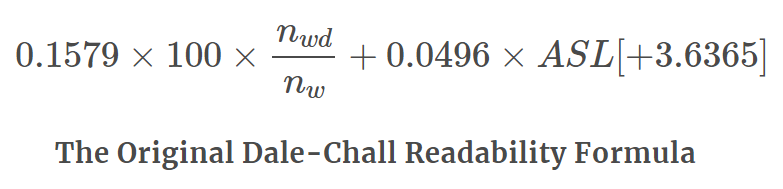
### Things we can do to improve

* **Reevaluate Training Data**: Ensure that the training data reflects the context in which these terms are used for better alignment with the target classification.
* **Feature Engineering**: Consider additional feature engineering techniques to capture the context and semantics of the text more effectively.
* **Stop Words Adjustment**: Refine the list of stop words based on their actual impact on the model's predictions, removing those with minimal or misleading contributions.

https://en.wikipedia.org/wiki/Readability

### 1. Dale–Chall Formula

The Dale–Chall Readability Formula assesses the readability of a text based on two factors: the percentage of difficult words (those not found on a specific list of 3,000 common words) and the average sentence length. The formula is as follows:



If the percentage of difficult words is above 5%, 3.636 is added to the raw score. The final score correlates with a U.S. grade level.

The Dale–Chall Readability Formula assigns a score that correlates with U.S. school grade levels to help determine how easily a text can be understood by readers at different education levels. Here’s how you can interpret the score:

common\_words = set([

"a", "able", "about", "above", "accept", "according", "account", "across",

"act", "addition", "additional", "admit", "after", "again", "against",

# ... (more common words)

"you", "young", "your", "yours", "yourself", "yourselves"

])

import re

def count\_sentences(text):

sentences = re.split(r'[.!?]', text)

return len([s for s in sentences if s.strip()])

def count\_words(text):

words = re.findall(r'\b\w+\b', text)

return len(words)

def count\_difficult\_words(text, common\_words):

words = re.findall(r'\b\w+\b', text)

difficult\_words = [word for word in words if word.lower() not in common\_words]

return len(difficult\_words)

def dale\_chall\_readability(text, common\_words):

total\_words = count\_words(text)

total\_sentences = count\_sentences(text)

difficult\_words = count\_difficult\_words(text, common\_words)

if total\_words == 0 or total\_sentences == 0:

return None # Avoid division by zero

percent\_difficult\_words = (difficult\_words / total\_words) \* 100

average\_sentence\_length = total\_words / total\_sentences

raw\_score = 0.1579 \* percent\_difficult\_words + 0.0496 \* average\_sentence\_length

if percent\_difficult\_words > 5:

raw\_score += 3.636

return raw\_score

# Example financial summary

financial\_summary = """

The company's revenue increased by 15% in the last quarter, reaching a total of $5 million.

This growth was driven by a 10% increase in sales and a 5% increase in service revenue.

Operating expenses remained stable at $3 million, leading to an operating profit of $2 million.

Net profit after tax amounted to $1.5 million, representing a 20% increase from the previous quarter.

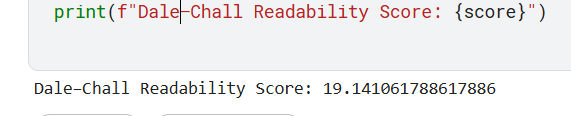
The company's cash flow from operations was strong, allowing for significant reinvestment in growth initiatives.

"""

# Calculate readability score

score = dale\_chall\_readability(financial\_summary, common\_words)

print(f"Dale–Chall Readability Score: {score}")



### Dale–Chall Readability Score Interpretation

The Dale–Chall score corresponds to a readability grade level as follows:

* **4.9 or below**: Easily understood by an average 4th-grade student or lower.
* **5.0 - 5.9**: Easily understood by 5th or 6th graders.
* **6.0 - 6.9**: Easily understood by 7th or 8th graders.
* **7.0 - 7.9**: Easily understood by 9th or 10th graders.
* **8.0 - 8.9**: Easily understood by 11th or 12th graders.
* **9.0 - 9.9**: Easily understood by college students.
* **10.0 and above**: May be difficult for college students to understand.

### Interpreting the Score 19.14

A Dale–Chall Readability Score of 19.14 is extremely high. This score suggests that the text is very difficult to read and would likely be challenging even for highly educated readers, such as those with graduate-level education.

Here are a few possible reasons for such a high score:

1. **High Percentage of Difficult Words**: The text contains a significant number of words not found in the common words list.
2. **Long Sentences**: The average sentence length is high, making the text more complex.
3. **Technical or Specialized Vocabulary**: Financial summaries often use specialized terminology that can increase readability scores.

### Improving Readability

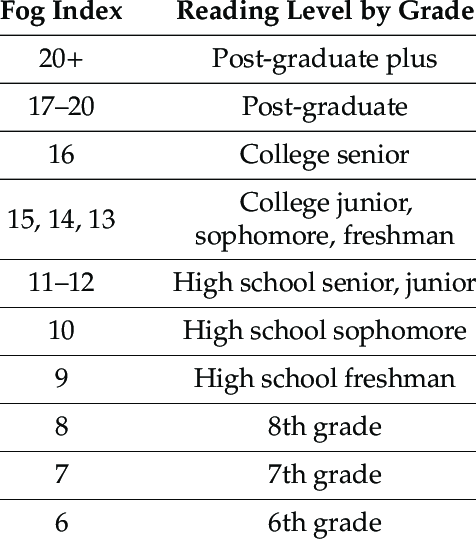
To make the text more accessible, consider the following strategies:

* **Simplify Vocabulary**: Use more common words and avoid jargon where possible.
* **Shorten Sentences**: Break long sentences into shorter, more manageable ones.
* **Clarify Complex Concepts**: Explain complex financial terms or concepts in simpler language.

### 2. Gunning Fog Formula

The Gunning Fog Index estimates the years of formal education needed to understand a text on the first reading. It considers the average sentence length and the percentage of complex words (words with three or more syllables). The formula is:





def gunning\_fog\_index(text):

total\_words = count\_words(text)

total\_sentences = count\_sentences(text)

complex\_words = count\_complex\_words(text)

if total\_words == 0 or total\_sentences == 0:

return None # Avoid division by zero

average\_sentence\_length = total\_words / total\_sentences

percent\_complex\_words = (complex\_words / total\_words) \* 100

fog\_index = 0.4 \* (average\_sentence\_length + percent\_complex\_words)

return fog\_index

# Example financial summary

financial\_summary = """

The company's revenue increased by 15% in the last quarter, reaching a total of $5 million.

This growth was driven by a 10% increase in sales and a 5% increase in service revenue.

Operating expenses remained stable at $3 million, leading to an operating profit of $2 million.

Net profit after tax amounted to $1.5 million, representing a 20% increase from the previous quarter.

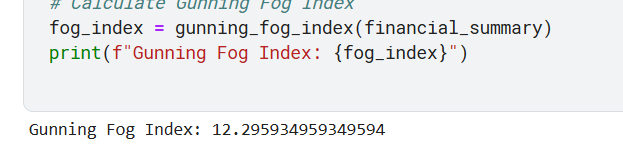
The company's cash flow from operations was strong, allowing for significant reinvestment in growth initiatives.

"""

# Calculate Gunning Fog Index

fog\_index = gunning\_fog\_index(financial\_summary)

print(f"Gunning Fog Index: {fog\_index}")



A Gunning Fog Index of 12.30 suggests that the text requires a reading level equivalent to a high school junior or senior, indicating moderate readability for a general audience.

### 3. Fry Readability Graph

The Fry Graph plots the readability level based on the number of syllables and sentences in a 100-word sample. To use the Fry Graph:

1. Select three 100-word passages.
2. Count the number of sentences in each passage.
3. Count the number of syllables in each passage.
4. Plot the average number of sentences and syllables per 100 words on the Fry Graph to determine the grade level.

import matplotlib.pyplot as plt

# Example financial summary

financial\_summary = """

The company's revenue increased by 15% in the last quarter, reaching a total of $5 million.

This growth was driven by a 10% increase in sales and a 5% increase in service revenue.

Operating expenses remained stable at $3 million, leading to an operating profit of $2 million.

Net profit after tax amounted to $1.5 million, representing a 20% increase from the previous quarter.

The company's cash flow from operations was strong, allowing for significant reinvestment in growth initiatives.

"""

def count\_sentences(text):

sentences = text.split('.')

return len(sentences)

def count\_syllables(word):

vowels = 'aeiouy'

count = 0

prev\_char\_was\_vowel = False

for char in word:

if char.lower() in vowels:

if not prev\_char\_was\_vowel:

count += 1

prev\_char\_was\_vowel = True

else:

prev\_char\_was\_vowel = False

return count

def analyze\_text(text):

words = text.split()

total\_words = len(words)

total\_sentences = count\_sentences(text)

total\_syllables = sum(count\_syllables(word) for word in words)

return total\_words, total\_sentences, total\_syllables

def fry\_readability\_index(total\_words, total\_sentences, total\_syllables):

words\_per\_sentence = total\_words / total\_sentences

syllables\_per\_word = total\_syllables / total\_words

fry\_index = 0.39 \* (words\_per\_sentence + 11.8 \* syllables\_per\_word) - 15.59

return fry\_index

# Analyze the financial summary

total\_words, total\_sentences, total\_syllables = analyze\_text(financial\_summary)

fry\_index = fry\_readability\_index(total\_words, total\_sentences, total\_syllables)

print(f"Fry Readability Index: {fry\_index}")

# Plot the Fry Readability Graph

plt.figure(figsize=(8, 6))

plt.plot(total\_sentences, total\_syllables, 'bo')

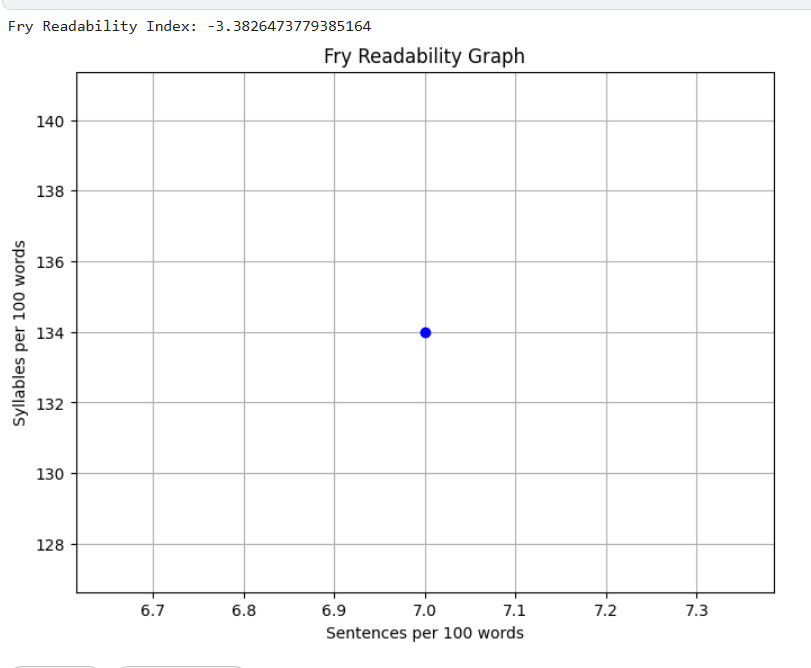
plt.xlabel('Sentences per 100 words')

plt.ylabel('Syllables per 100 words')

plt.title('Fry Readability Graph')

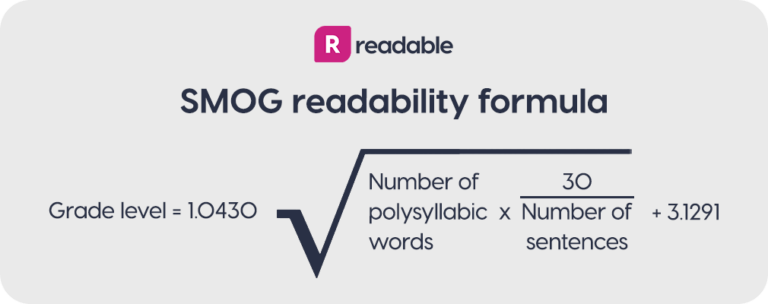
plt.grid(True)

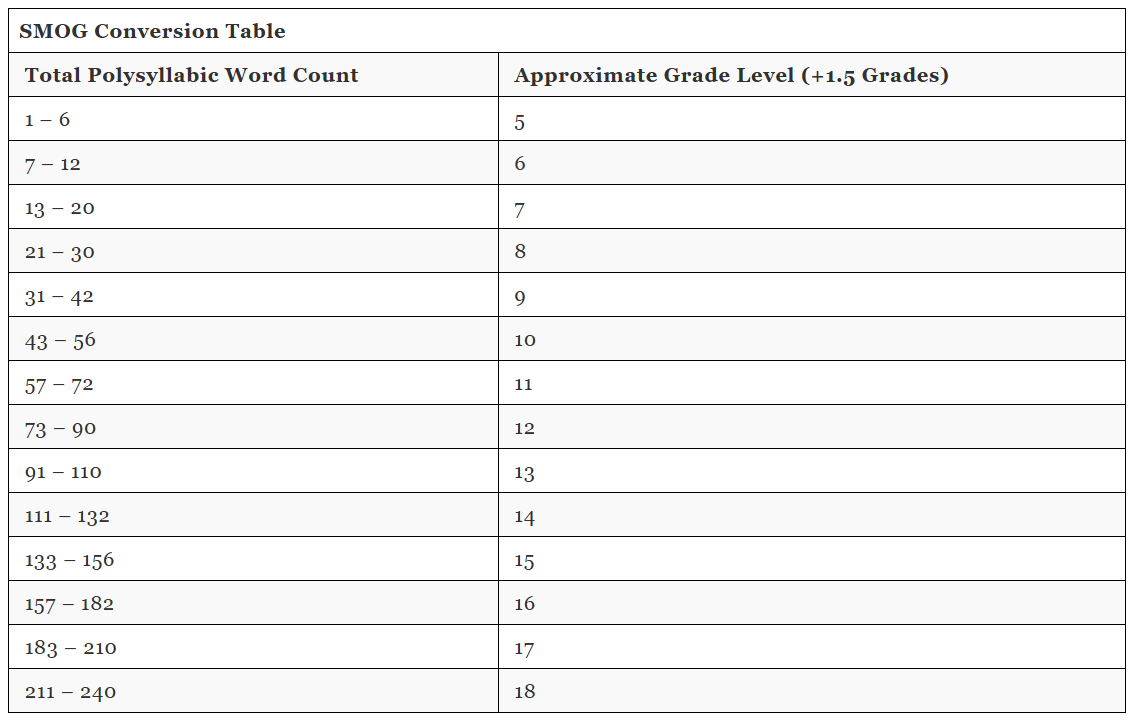
plt.show()



### 4. McLaughlin’s SMOG Formula

The Simple Measure of Gobbledygook (SMOG) estimates the years of education required to understand a text. It is calculated using the number of polysyllabic words (words with three or more syllables). The formula is:





import re

import math

# Example financial summary

financial\_summary = """

The company's revenue increased by 15% in the last quarter, reaching a total of $5 million.

This growth was driven by a 10% increase in sales and a 5% increase in service revenue.

Operating expenses remained stable at $3 million, leading to an operating profit of $2 million.

Net profit after tax amounted to $1.5 million, representing a 20% increase from the previous quarter.

The company's cash flow from operations was strong, allowing for significant reinvestment in growth initiatives.

"""

def count\_polysyllabic\_words(text):

words = re.findall(r'\b\w+\b', text)

polysyllabic\_words = [word for word in words if syllable\_count(word) >= 3]

return len(polysyllabic\_words)

def syllable\_count(word):

word = word.lower()

vowels = "aeiouy"

count = 0

if word[0] in vowels:

count += 1

for index in range(1, len(word)):

if word[index] in vowels and word[index - 1] not in vowels:

count += 1

if word.endswith("e"):

count -= 1

if count == 0:

count += 1

return count

def smog\_readability\_score(text):

total\_sentences = len(re.findall(r'\.', text))

polysyllabic\_words = count\_polysyllabic\_words(text)

if total\_sentences == 0:

return None # Avoid division by zero

smog\_score = 1.0430 \* math.sqrt(polysyllabic\_words \* (30 / total\_sentences)) + 3.1291

return smog\_score

# Calculate McLaughlin’s SMOG Formula for the financial summary

smog\_score = smog\_readability\_score(financial\_summary)

print(f"McLaughlin’s SMOG Score: {smog\_score}")

# Interpret the SMOG score

def interpret\_smog\_score(score):

if score is None:

return "Unable to calculate due to insufficient data."

elif score <= 12:

return "Understood by high school graduates."

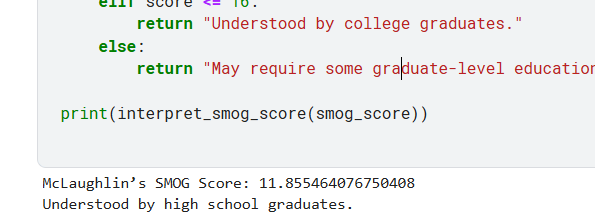
elif score <= 16:

return "Understood by college graduates."

else:

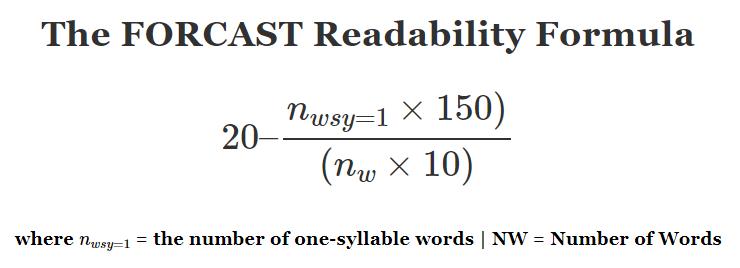
return "May require some graduate-level education to understand."

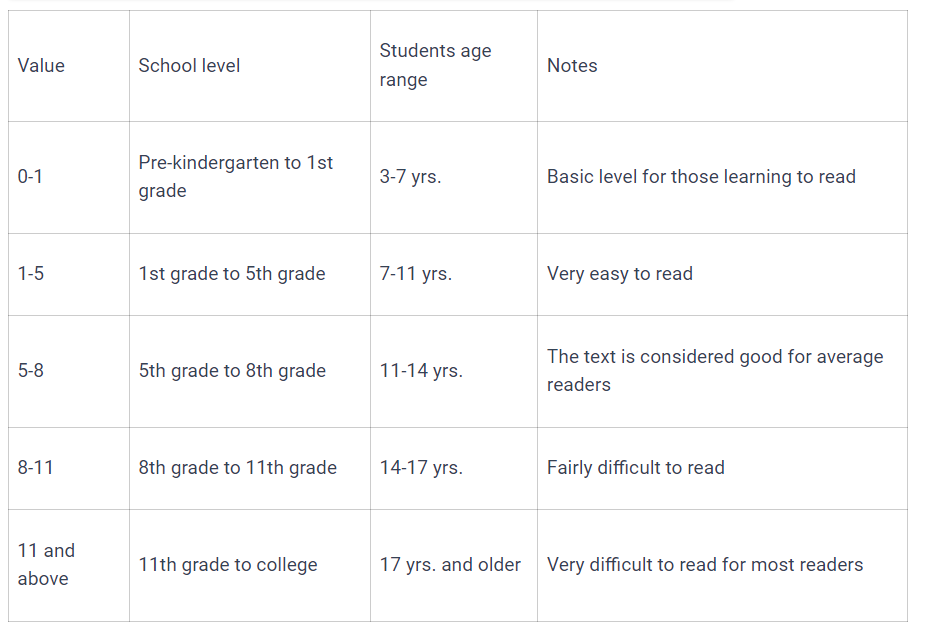
print(interpret\_smog\_score(smog\_score))



### 5. FORECAST Formula

The FORECAST Formula is used for technical and scientific texts. It considers the number of single-syllable words in a 150-word passage. The formula is:





# Calculate FORECAST Formula for the financial summary

forecast\_score = forecast\_readability\_score(financial\_summary)

print(f"FORECAST Score: {forecast\_score}")

# Interpret the FORECAST score

def interpret\_forecast\_score(score):

if score <= 5:

return "Easily understood by average 5th-grade students or lower."

elif score <= 10:

return "Easily understood by average 10th-grade students or lower."

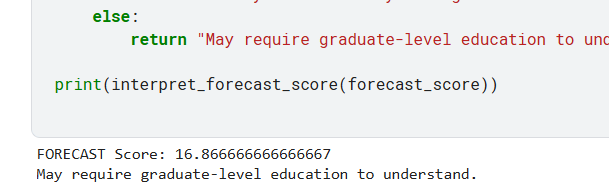
elif score <= 15:

return "Easily understood by college students."

else:

return "May require graduate-level education to understand."

print(interpret\_forecast\_score(forecast\_score))



### 6. Readability and Newspaper Readership

Readability scores are crucial for newspapers to match their content with the reading skills of their audience. Newspapers often aim for a readability level around the 8th-grade level to ensure that their content is accessible to a broad audience. High readability can improve comprehension and reader engagement.

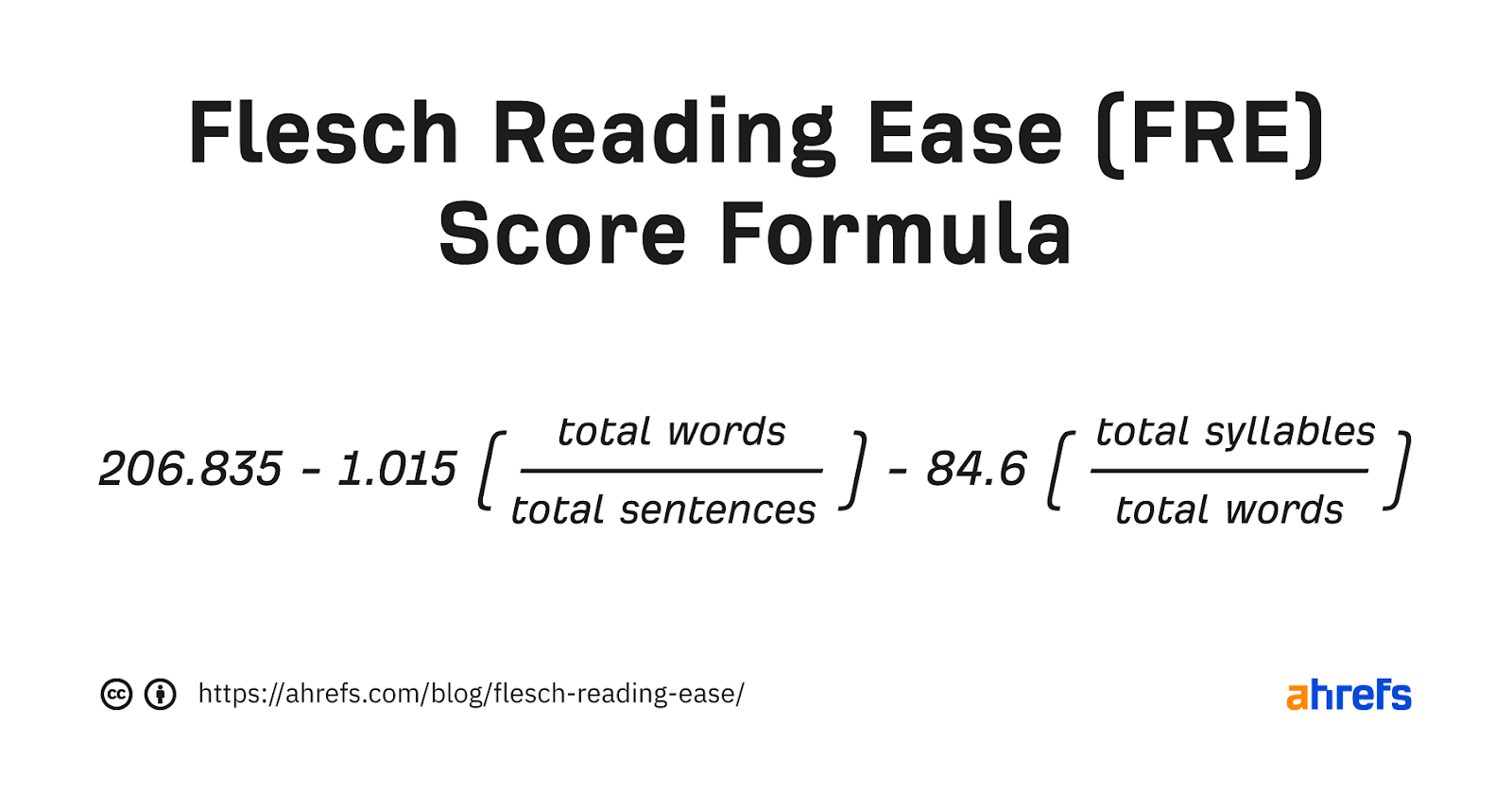
### 7. Flesch Scores

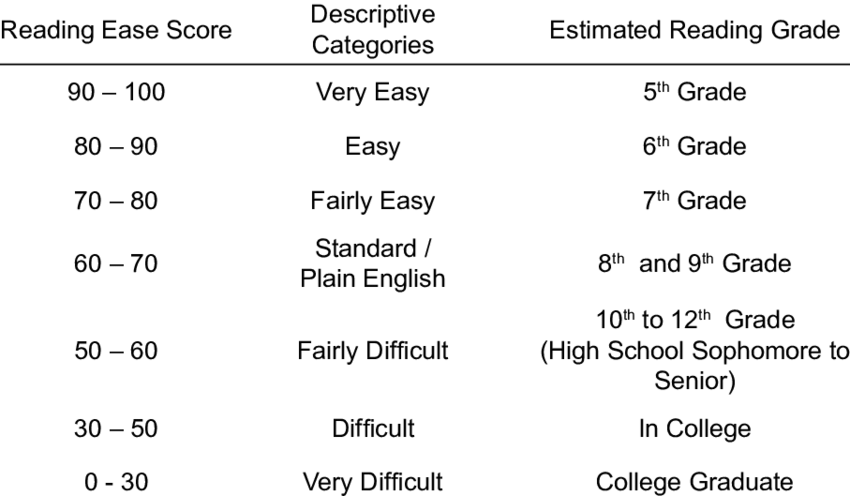
The Flesch Reading Ease and Flesch-Kincaid Grade Level are two related formulas:

* **Flesch Reading Ease**: This score ranges from 0 to 100, with higher scores indicating easier readability. It uses sentence length and syllable count.

Formula

* **Flesch-Kincaid Grade Level**: This score correlates with U.S. school grade levels.





def calculate\_flesch\_reading\_ease(text):

total\_words = count\_words(text)

total\_sentences = len(re.findall(r'[.!?]', text))

total\_syllables = sum(count\_syllables(word) for word in re.findall(r'\b\w+\b', text))

flesch\_reading\_ease = 206.835 - 1.015 \* (total\_words / total\_sentences) - 84.6 \* (total\_syllables / total\_words)

return flesch\_reading\_ease

def calculate\_flesch\_kincaid\_grade\_level(text):

total\_words = count\_words(text)

total\_sentences = len(re.findall(r'[.!?]', text))

total\_syllables = sum(count\_syllables(word) for word in re.findall(r'\b\w+\b', text))

flesch\_kincaid\_grade\_level = 0.39 \* (total\_words / total\_sentences) + 11.8 \* (total\_syllables / total\_words) - 15.59

return flesch\_kincaid\_grade\_level

# Calculate Flesch Reading Ease and Flesch-Kincaid Grade Level for the financial summary

flesch\_reading\_ease = calculate\_flesch\_reading\_ease(financial\_summary)

flesch\_kincaid\_grade\_level = calculate\_flesch\_kincaid\_grade\_level(financial\_summary)

print(f"Flesch Reading Ease: {flesch\_reading\_ease}")

print(f"Flesch-Kincaid Grade Level: {flesch\_kincaid\_grade\_level}")

# Interpret the Flesch scores

def interpret\_flesch\_reading\_ease(score):

if score >= 90:

return "Very easy to read. Easily understood by an average 11-year-old student."

elif score >= 80:

return "Easy to read. Conversational English for consumers."

elif score >= 70:

return "Fairly easy to read."

elif score >= 60:

return "Plain English. Easily understood by 13- to 15-year-old students."

elif score >= 50:

return "Fairly difficult to read."

elif score >= 30:

return "Difficult to read."

else:

return "Very difficult to read. Best understood by university graduates."

def interpret\_flesch\_kincaid\_grade\_level(level):

if level <= 5:

return "Easily understood by average 10-year-olds."

elif level <= 8:

return "Easily understood by average 13- to 15-year-olds."

elif level <= 10:

return "Easily understood by average 16- to 17-year-olds."

elif level <= 12:

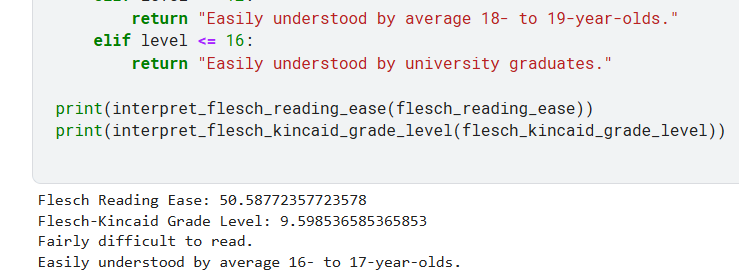
return "Easily understood by average 18- to 19-year-olds."

elif level <= 16:

return "Easily understood by university graduates."

print(interpret\_flesch\_reading\_ease(flesch\_reading\_ease))

print(interpret\_flesch\_kincaid\_grade\_level(flesch\_kincaid\_grade\_level))



### Application to Model Summarizer Output

Applying these readability formulas to the output of a model summarizer can help assess the clarity and accessibility of the summaries. This evaluation ensures that the summaries are suitable for the intended audience, improving user experience and comprehension. Here’s how each formula can be applied:

* **Dale–Chall**: Identify difficult words and calculate the score to gauge the grade level.
* **Gunning Fog**: Use sentence length and complex words to determine the education level needed.
* **Fry Graph**: Plot sentences and syllables per 100 words to find the grade level.
* **SMOG**: Count polysyllabic words and use the formula to assess readability.
* **FORECAST**: For technical texts, count single-syllable words in a sample.
* **Flesch Scores**: Calculate reading ease and grade level for overall readability.

These metrics help fine-tune the summarizer to produce outputs that are easy to read and understand, tailored to the target audience's reading level.