MSE | MASTER OF SCIENCE IN ENGINEERING

Hes·so
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences and Arts Western Switzerland

Master of Science HES-SO in Engineering
Av. de Provence 6
CH-1007 Lausanne

Master of Science HES-SO in Engineering
Orientation: Data Science – Data Analytics (DS)

SAFE: Sentence-based temporal and sentiment Analysis of Financial news to Enhance stock market Analysis

Author:
Léonard Nikita Noth

Under the direction of:
Dr. Sandy Ingram

ISIS

Fribourg, HES-SO//Master, February 2024

Information about this report

Contact information:

Author:	Léonard Nikita Noth
	MSE Student
	HES-SO//Master
	Switzerland
Email:	me@leonardnoth.com

Declaration of honour:

I, undersigned, Léonard Nikita Noth, hereby declare that the work submitted is the result of a personal work. I certify that I have not resorted to plagiarism or other forms of fraud. All sources of information used, and the author quotes were clearly mentioned.

Place, date: **Fribourg, 09.02.2023**

Signature: 

Validation:

Accepted by the HES-SO//Master (Switzerland, Lausanne) on a proposal from:

Dr. Sandy Ingram, Thesis project advisor

Place, date: _____

Signature: _____

Acknowledgments

The completion of this thesis would not have been possible without the contributions, support and encouragement of several individuals. I would like to express my deepest gratitude to them.

Firstly, I would like to express my sincere appreciation to my supervisor, Sandy Ingram. Her guidance, expertise and support throughout this journey have been essential in making this project a reality. Her flexibility in scheduling to accommodate my work commitments alongside this thesis has allowed for a harmonious balance that has made this work possible.

I am also grateful to Samuel Fringeli for his assistance during the dockerisation phase of this project. His expertise in GPU configuration and Docker was crucial in overcoming the technical challenges that arose, ensuring a scalable architecture. His willingness to share his knowledge greatly enhanced this work.

Special thanks also go to ChatGPT, the AI developed by OpenAI. This tool has significantly improved the quality of my written English in the report, making the writing process more efficient and the final product much better.

Their collective contributions have significantly enriched this thesis, and I am grateful for their support, which went above and beyond their professional responsibilities.

Abstract

This thesis presents a comprehensive system designed to enhance financial analysis through advanced temporal and sentiment analysis. At its core, the project develops two key analysis toolkits and a dataset creation/augmentation command-line interface (CLI) to generate and enhance data specifically for financial contexts. The effort begins with identifying dataset requirements and exploring existing resources and augmentation techniques to prepare the ground for tool development.

The temporality toolkit focuses on understanding the timing and relevance of financial events, while the sentiment toolkit aims to measure the emotional tone of financial news. Both toolkits undergo development, testing and integration processes, resulting in a Decision Maker component that synthesizes the analysis results into actionable financial insights.

A simple user interface was added to the Analysis toolkit to ensure that the system is accessible and effective for end users, prioritizing ease of use and clear presentation of analysis results.

The report concludes with reflections on the project's limitations, potential for future development and a self-assessment of its achievements. Through this integrated approach, the system promises to provide nuanced insights into financial news and support informed decision-making in the dynamic financial sector.

Keywords: Financial analysis, temporality, sentiment analysis, data augmentation, machine learning, decision making.

Table of Contents

Table of Figures

Table of Tables

1. Introduction

1.1. General Context

The intersection of finance and technology, often referred to as fintech, has been a fertile ground for innovation and transformation, reshaping the way individuals and institutions interact with financial services. The exponential growth of digital data, driven by online transactions, social media and the digitisation of financial records, presents both challenges and opportunities. The ability to sift through, analyse and derive meaningful insights from this flood of data has become a cornerstone of competitive advantage. This is particularly true in the field of financial analysis, where understanding market dynamics, investor sentiment and temporal trends can significantly influence investment strategies and economic forecasts.

The emergence of advanced computational techniques, including machine learning (ML) and natural language processing (NLP), has brought a new era of data analysis. These technologies offer the potential to automate the extraction of insights from vast, unstructured data sets, including news articles, financial reports and social media feeds. However, realising this potential requires specialised tools that can understand the nuances of financial language, recognise the significance of temporal markers, and measure the emotional undertone of textual content.

This Master's thesis is situated at the intersection of these needs and opportunities. It aims to develop a comprehensive analytical toolkit that uses ML and NLP to provide in-depth entity-based sentiment and temporality analysis tailored to the financial sector. This project is not only about technological advancement, but also about bridging the gap between the quantitative data that has traditionally underpinned financial analysis and the qualitative insights that are increasingly recognised as critical to understanding market behaviour.

The project considers the complexity of financial texts, which are often loaded with industry-specific jargon, implicit temporal references and subtle expressions of sentiment. Addressing this complexity requires a multi-faceted approach that combines dataset creation and enrichment, text segmentation, sentiment and temporal classification, and a coherent decision framework. Each component of the project is designed to address specific challenges within this context, from identifying and classifying temporal references to detecting nuanced expressions of sentiment.

1.2. Project Scope And Objectives

The overall objective of this thesis is to design, implement and evaluate a suite of tools that enable advanced entity-based sentiment and temporality analysis in the financial domain. The project covers several key areas:

- **Dataset Creation/Augmentation CLI:** This component focuses on developing a command line interface (CLI) for efficiently creating and augmenting datasets. It is implemented to be generic but will be used specifically to create a temporality and a sentiment analysis dataset. It involves defining precise dataset requirements, mining existing datasets for potential use or adaptation, and applying innovative textual dataset augmentation techniques to improve dataset quality and diversity.
- **Text segmentation module:** Recognizing the importance of context and granularity in text analysis, this module aims to implement a custom text segmentation solution. By identifying meaningful text segments, the module improves the accuracy and relevance of subsequent analysis.
- **Temporal Classification Toolkit:** A specialized temporal classification toolkit will be developed to accurately identify, extract and classify temporal references within financial text. This enables more dynamic analysis of financial events and trends over time.
- **Sentimental Classification Toolkit:** This component involves the development of a sentiment analysis model capable of detecting and categorizing the emotional and sentimental nuances of financial text. The toolkit helps to understand market sentiment, investor attitudes and the potential impact on financial markets.
- **Decision maker:** The integration of sentiment and temporal analysis tools into a comprehensive decision-making framework is a key objective. This algorithm synthesizes insights from both analyses toolkits to support informed decision-making in financial contexts.
- **User Interface (UI):** Although not a primary objective, the development of a user-friendly interface is recognized as an important secondary objective. The UI aims to make the toolkit accessible and usable, allowing users to easily enter data, initiate analyses and visualize results. The emphasis is on ensuring that the sophisticated capabilities of the toolkit are matched by an intuitive and simple user experience, thereby extending the utility of the tool beyond academic and professional circles to a broader audience interested in financial analysis.

In summary, this project aims to make a technological advance in the field of financial analysis, bridging the gap between quantitative data and qualitative insights through the innovative application of ML and NLP techniques. The comprehensive approach, ranging from dataset augmentation to user interface design, reflects a holistic view of the challenges and opportunities of using digital data for financial analysis.

2. Dataset Creation/Augmentation CLI

In our journey to develop an advanced financial news analysis system, obtaining the right datasets is one of the required elements. The specific nature of our system requires datasets that are structured for 'sentence-wise' or 'meaningful group of words-wise' analysis. This is crucial as we aim to perform in-depth sentiment and temporality analysis at this granular level, particularly in the financial news domain. The datasets will be essential for training, testing and evaluating our system. This section delves into the intricacies of creating these datasets, ensuring that they contain samples that not only fit our analytical 'shape' (aka “sentence wise” or “meaningful group of words”), but also provide information that is conducive to sentiment and temporality classification.

We will begin our research by identifying the specific information that our dataset needs to encapsulate or emphasise. This will give us a preliminary idea of its structure, although the intricacies of structuring will inherently be influenced by our intention to analyse on a 'sentence-by-sentence' basis. In order to identify potential datasets, our primary focus will be on those related to temporality and sentiment analysis in financial news. However, recognising the limitations that can arise from a narrow scope, we will also broaden our search to include datasets from other areas. Such a strategy may uncover a dataset in another domain that closely matches our requirements and can be easily adapted for our purpose. Based on the two previous points, our aim is to explore methods of creating or augmenting textual datasets, ensuring that we take steps to mitigate any potential risks arising from data absence.

The final step in this section involves building a Command Line Interface (CLI) tool. This will assist in the dataset creation and augmentation processes. We will lay out the specifications for this tool, exposing the main points of the development of the CLI, and finally test and document the testing.

2.1. Defining Dataset Requirements

Having the appropriate datasets is key in our journey to develop an advanced financial news analysis system. In this section, we will address the challenge of creating/augmenting datasets that are tailored to our specific needs, particularly for temporality and sentiment analysis.

We will start by defining exactly what our dataset should contain and outline. We will then look at existing datasets related to temporality and sentiment analysis in the financial sector, and afterwards extend it to other research fields. Based on the previous two points, we will look at existing methods for creating and enriching textual datasets ensuring that we take steps to mitigate any potential risks arising from data absence.

The final step in this section is to build a command line interface (CLI) tool. This will support the dataset creation and augmentation processes. We will set out the specifications for this tool, outline the main points of the CLI development, and finally test and document the tests.

2.1.1. Temporality Analysis Dataset

Temporal analysis in finance goes beyond the chronological ordering of events. It delves into the intricate interplay of past actions, present circumstances, and future expectations, weaving them into a comprehensive narrative that shapes market dynamics. The temporal context of news articles, financial reports and investor communications can provide invaluable insights, revealing patterns and trends that might otherwise go unnoticed.

In order to carry out such an analysis, we need a dataset that is tailored to our needs and domain, to ensure that we capture domain-specific nuances. This section explores the requirements for such a dataset, highlighting the importance of capturing the multiple temporal dimensions that drive financial narratives and decisions.

Importance of Temporality in Financial Analysis

Temporal context is not just a backdrop but a driving force in the field of financial news and their analysis. The timing of events, whether rooted in the past, unfolding in the present or looming on the horizon, acts as a compass guiding market dynamics and investor sentiment.

For instance, in the paper named “STaCK: Sentence Ordering with Temporal Commonsense Knowledge” [1] the significance of understanding events in their chronological order. This can be likened to assembling a story from individual chapters, where each event builds upon the previous one, creating a cohesive narrative. Historical occurrences, like major market downturns or significant regulatory changes, play a foundational role in shaping today's financial landscape. They act as reference points, offering valuable insights and guiding principles for both investors and analysts navigating current market scenarios. Past events, such as historical market crashes or landmark regulatory decisions, often set the stage for current market conditions and serve as lessons or benchmarks for investors and analysts alike.

The concept of real-time localisation of events, as outlined in the paper named “Adaptive Proposal Generation Network for Temporal Sentence Localization in Videos” [2], brings the focus on the immediacy and urgency of the financial world. In this digital age, where information travels at the speed of light, real-time updates or announcements about companies, stocks or market conditions can ripple through global markets in minutes. These rapid shifts in market sentiment underscore the importance of being attuned to the present and reacting quickly to emerging trends and news.

On the other hand, the art of forecasting, of gazing into the crystal ball to anticipate future events, is equally crucial. In the paper named “A Sequential Model for Classifying Temporal Relations between Intra-Sentence Events” [3], Choubey and Huang's exploration of understanding temporal relationships between intra-sentence events offers a glimpse into this aspect. Predicting events and their relationship within a sentence, whether it is a company's quarterly performance or potential geopolitical shifts, can shape market strategies in the near future and give investors a competitive edge.

In conclusion, the temporal dimension of financial news is an indispensable aspect of market analysis, as it can provide valuable insights on the financial world and how it will react to a given news. The interplay of past events, present dynamics and future expectations within the same news could be confusing as it can create a dynamic narrative that is key to understand to perform an informed decision making. By understanding and analysing this temporal context, investors and analysts can navigate the complexities of the financial landscape with greater clarity and foresight. As the papers reviewed show, whether it's piecing together historical events, reacting to real-time updates, or forecasting future trends, temporality remains at the heart of financial analysis, underscoring its central role in shaping investment strategies and market outcomes.

What Temporal Classes are Important to Detect?

The impact of a financial news is closely linked to its temporal context. Given the importance of time in shaping financial narratives, it's crucial to identify and categorise events according to their temporal dimensions. The selection of temporal classes was guided by the following elements:

- **Relevance to financial dynamics:** Financial markets react to a variety of events, from historical benchmarks to real-time updates and future projections. Each temporal dimension plays a distinct role in influencing market sentiment and guiding investment decisions.
- **Insights from academic research:** The referenced papers especially provided valuable perspectives on temporality in different contexts. From these studies, we have deducted classes that resonate strongly with the financial domain.
- **Complexity of financial narratives:** Financial news often interweaves past events, present scenarios, and future projections to create a comprehensive narrative. Our different temporal classes will help to deconstruct these narratives for clearer understanding.

Based on these considerations, the following temporal classes have emerged as essential:

- Distant Past
- Near Past
- Present
- Near future
- Distant future

A deeper exploration of these temporal classes and their implications will be elucidated in the subsequent section.

Dataset Specification

Building upon a synthesis of insights from the referenced papers and a nuanced understanding of financial market dynamics, we've identified distinct temporal classes that are interesting to classify. These classes serve as a lens to view and categorize the multifaceted temporal aspects inherent in financial news. Presented in … below are the classes we will have, and a description offering both a general perspective and a tailored interpretation within the financial context.

Table 1: Temporal classes description

TEMPORAL CLASS	DESCRIPTION
DISTANT PAST	Historical events from a significant period of time ago, such as major market downturns from years past or landmark regulatory decisions, which provide broader context and fundamental reference points.
RECENT PAST	Events in the recent past, such as last quarter's results, recent policy changes or market movements in the last few days, which have a direct impact on current market dynamics.
PRESENT	Current events, including real-time updates, price movements and immediate market reactions.
NEAR FUTURE	Events that are expected to happen soon, such as upcoming quarterly results or upcoming corporate actions.

DISTANT FUTURE	Long-term predictions and expectations, relating to potential market trends or technological innovations.

2.1.2. Sentiment Analysis Dataset

We consider that sentiment analysis, particularly in the financial sector, goes beyond simply classifying sentiment as positive, negative, or neutral. It delves deeper into the emotional undertones of news articles, reports, and investor communications, extracting nuanced emotions that can significantly influence investment decisions.

In addition to capturing these emotional undertones which will be specified in this section, a well-constructed sentiment analysis dataset also reflects the domain-specific jargon and nuances of the financial world. This subsection explores the requirements for such a dataset, emphasising the importance of capturing the multifaceted emotions that influence trading and investment decisions.

Importance of Emotion and Sentiment in Financial Analysis

The complex dynamics of financial markets are heavily influenced by the rapid dissemination of news and information. As the paper on Evaluation of Sentiment Analysis in Finance: From Lexicons to Transformers [4] highlights that the ability to identify the sentiment of financial news articles quickly and accurately is paramount. This ability becomes even more critical in volatile markets, where news can trigger rapid and significant price movements. The paper addresses the challenges posed by the domain-specific language of finance and the nuanced expression of sentiment. Through a comprehensive evaluation of various sentiment analysis methods, it highlights the superior efficiency of modern NLP techniques such as transformers-based text classification for sentimental analysis. These advanced tools, capable of capturing the complex emotions embedded in financial text, if done properly could provide invaluable insights into market sentiment, enabling traders and investors to make more informed decisions.

Delving deeper into the realm of financial markets, emotions and sentiments emerge as powerful undercurrents influencing investment behaviour. The paper Investment Sentiment in Finance Market [5] offers a granular exploration of this complex relationship. While quantitative, data-driven analysis remains a cornerstone of investment strategies, the emotional pulse of the market often plays a decisive role in shaping short-term trends. Investment decisions made at the intersection of data and emotion are influenced by a myriad of factors. This paper systematically examines various indicators of investment sentiment, shedding light on the delicate balance between data-driven insights and emotional intuition. It highlights the importance of recognising and understanding these emotional undercurrents, as they can have a significant impact on investment outcomes.

Asset price bubbles, a phenomenon that has long fascinated financial analysts, provide a vivid illustration of the collective emotional psyche of the market. The study Behavioral Framework of Asset Price Bubbles: Theoretical and Empirical Analyses [6] provides a comprehensive analysis of this phenomenon. By creating a model that integrates the behaviour of fundamental traders and extrapolative investors, the study reveals the profound impact of news on market sentiment. Positive news about fundamentals can lead to overwhelming market optimism, driving asset prices to unsustainable levels. Conversely, negative news can plunge the market into pessimism, leading to sharp downturns. The study's emphasis on the tangible impact of

collective sentiment on market dynamics underscores the need for sophisticated sentiment analysis tools capable of capturing this market sentiment in real time.

While exchange rates are primarily driven by macroeconomic indicators, they are also influenced by investor sentiment. The paper Investor sentiments and exchange rate volatility [7] explores this complex interplay. By constructing a sentiment index that captures the emotional pulse of the market, the study embarks on an empirical exploration of the influence of investor sentiment on exchange rate volatility. The findings reveal the multifaceted nature of financial markets, where rational indicators and emotional sentiments constantly interact to shape market outcomes. The paper highlights the importance of understanding these emotional undercurrents, as they provide invaluable insights into potential market movements and investor behaviour.

When we dive deep into the world of financial news, it becomes clear that there's a lot more to the story than just the cold, hard numbers. Emotions play a huge role in shaping how markets move and how investors react. From the papers we've examined, it's clear that understanding the sentiment behind financial news is crucial. Think about it: when markets are unpredictable (and let's face it, when aren't they?), reading the emotional undertones can be a game changer.

These studies also remind us that our feelings, whether they're optimism, fear or anything in between, can seriously influence our investment decisions. It's not just about what's happening on the surface; it's about the underlying emotions that drive those decisions. So, the big takeaway? Tapping into the emotional pulse of the financial world is key. By understanding the feelings driving the headlines, traders and investors can get a fuller picture, making it easier to navigate the ever-changing financial landscape.

What Emotions are Important to Detect?

The financial market, often likened to a living organism, pulsates with a myriad of emotions. Identifying and understanding these emotions is not just an academic exercise; it's a practical necessity for traders, investors and financial analysts. Based on the papers we analysed in the previous section, here is a list of emotions that were mentioned:

- **Fear and Greed** [5]**:** These primal emotions, highlighted in Investment Sentiment in the Financial Market, often act as the twin forces driving market movements. Fear can lead to rapid selloffs, while greed can fuel buying frenzies. Achieving a balance between these emotions is critical to sustainable investment strategies.
- **Optimism and pessimism** [5] [6]**:** These sentiments, as discussed in the paper Behavioural Framework of Asset Price Bubbles, can shape market trends. Excessive optimism can lead to asset price bubbles, while pervasive pessimism can trigger market downturns.
- **Overconfidence** [5] [6] [7]**:** A sentiment that can lead to excessive risk-taking. Overconfident investors who believe in their superior knowledge or intuition may miss critical market signals, leading to potential losses.
- **Uncertainty and doubt** [5] [7]**:** These emotions, often triggered by ambiguous or conflicting news, can lead to market volatility. Investors struggling with uncertainty may delay investment decisions or seek safer investments.
- **Trust and distrust** [5] [7]**:** In a market where transactions are based on confidence, trust plays a key role. Conversely, distrust, whether in financial institutions, market regulation or economic policy, can deter investment and stifle market growth.

In addition, a comprehensive study entitled "Enhancing Financial Market Analysis and Prediction with Emotion Corpora and News Co-Occurrence Network" [8] delves deeper into

this aspect, extending the categorisation of emotions from just the basic three emotions to 24 different emotions. Inspired by Plutchik's research, the study used eight core emotions: joy, sadness, trust, disgust, fear, anger, surprise and anticipation, which form the basis of Plutchik's famous Wheel of Emotion. This wheel is a well-accepted scientific model that categorises human emotions in a structured and hierarchical way.

The inclusion of the Plutchik Wheel of Emotion as a foundational model for emotion determination demonstrates the study's methodical and research-based approach. These primary emotions were further combined, resulting in twenty-four more granular emotions that provide a multidimensional perspective on emotional responses. This approach to emotion identification and categorisation is something we aim to emulate in our project. By grounding our emotion selection in established scientific models, we ensure that our choices are not arbitrary, but are rooted in well-researched psychological frameworks. However, it's worth noting that despite the study's [8] robust approach, the dataset and code used are not publicly available. This foundation will be pivotal as we further explore and decide upon the specific emotions that are most pertinent to our objectives.

Now that we've agreed on the interest of this wheel, let's try to understand it better. Plutchik's Wheel of Emotions is a categorical model based on eight primary labels: Joy, Trust, Fear, Surprise, Sadness, Disgust, Anger and Anticipation. This model represents emotions in a flower-shaped diagram that has become a classic reference in the field of emotion research. The wheel uses the arrangement of the 'petals' (representing individual emotions) to highlight the similarities or contrasts between emotions as we can see in Figure 1**.** Emotions that are similar in nature are placed in the same 'hemisphere' of the wheel, and their spatial arrangement is crucial as it signifies their relationships and potential combinations. This unique visual representation allows an intuitive understanding of the relationships between different emotions, their intensities**,** and their combinations, making it a valuable tool for understanding the complex landscape of human emotions.

Figure 1: Plutchik Wheel of Emotions

Dyads in Plutchik's model refer to combinations of emotions. Depending on their spatial proximity on the wheel, similar emotions can be combined into primary, secondary and tertiary dyads. Primary dyads are formed by combining two adjacent emotions on the wheel, such as the combination of 'joy' and 'trust', which could represent 'love'. Secondary dyads are formed by emotions that are two petals apart, and tertiary dyads are formed by emotions that are three petals apart. The exact emotions included in each dyad may vary depending on the specific emotions combined and their relative positions on the wheel as we can see in the Figure 2**.**

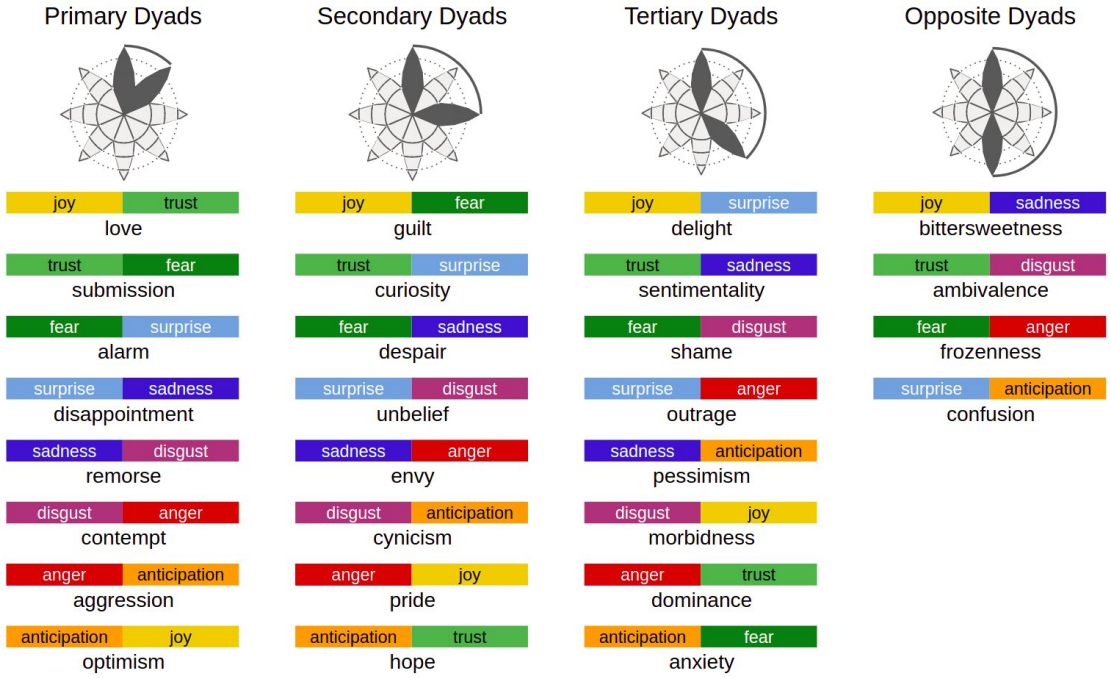

Figure 2: Plutchik's Emotion Wheel Dyads

Dataset Specification

Building on the foundations of previous studies and the Plutchik's Wheel of Emotions, we are ready to define the classes for our dataset. The wheel, with its eight primary emotions, provides a comprehensive framework that captures the breadth and depth of human emotional experience and allowing granularity by combining emotions. These eight pillars: joy, trust, fear, surprise, sadness, disgust, anger and anticipation, will serve as the primary classes for our dataset.

In the initial stages of our classification, we've opted for the eight primary emotion classes. This choice was made with the intention of minimising the number of classes, thereby mitigating the potential performance degradation that often accompanies an increase in the number of classes. Furthermore, these basic emotions allow for subsequent combinations to construct emotion dyads, providing flexibility without complexity. This streamlined approach makes our dataset not only robust - given the distinct nature of these basic emotions - but also more manageable, as a reduced set of classes facilitates adaptability, comprehension and overall management.

You can find here under Table 2 listing the classes and an explanation of what they mean.

Table 2: Plutchik Emotions Description

Emotion Name	Emotion Description
Joy	Often experienced as a feeling of happiness or elation, joy is the emotion that comes from achieving a goal, finding something pleasurable, or experiencing love and connection. It's the opposite of sadness and can manifest as contentment, satisfaction or even euphoria in its most intense form.
Trust	This emotion is characterised by a feeling of safety and security in the presence of another. Trust is built up over time and is fundamental to healthy relationships. It's the opposite of disgust and can be thought of as acceptance, belief or confidence in someone or something.
Fear	A primal emotion, fear arises in response to perceived threat or danger. It's a survival mechanism that triggers the body's "fight or flight" response. Fear can range from mild discomfort or apprehension to intense terror or dread. It is the opposite of anger on Plutchik's wheel.
Surprise	This is a brief emotional state caused by an unexpected event. Surprise can be pleasant or unpleasant and is often accompanied by a physical startle response. It's a neutral emotion and can lead to joy or fear, depending on the context. Its opposite is anticipation.
Sadness	This emotion is characterised by feelings of loss, helplessness or disappointment. Sadness arises when we experience setbacks or losses. It's a natural response to challenging situations and is the opposite of joy. In its intense form, it can lead to feelings of despair or hopelessness.
Disgust	This emotion arises from a sense of disgust or revulsion. It can be a reaction to something that is offensive, morally wrong or physically repulsive. Disgust acts as a protective mechanism, keeping us away from potentially harmful situations or substances. It's the opposite of trust.
Anger	A strong feeling of displeasure or hostility, anger arises when we feel wronged or face obstacles. It can motivate us to take corrective action but can also lead to aggression if left unchecked. On Plutchik's wheel, anger is opposite fear.
Anticipation	This emotion is characterised by excitement or anxiety about a future event. It's a forward-looking emotion, often involving preparation or planning for what's to come. Anticipation can be positive (e.g. looking forward to a celebration) or negative (e.g. dreading an upcoming task). It's the opposite of surprise.

Although the dyads of the Plutchik Wheel offer a comprehensive range of emotions, we have chosen not to include all of them directly in our model. Our primary classification focuses only on the eight basic emotions. Subsequently, when two emotions are close in terms of detected probability, we interpret this to mean that the text is likely to represent a combination of both basic emotions, thus signifying a dyadic emotion. This methodology ensures that we capture the depth of the emotional spectrum without overcomplicating our initial classification. Details of the specific dyads we've considered for this study are detailed in Table 3.

Table 3: Dyad choice and description

Dyad name	Related Primary emotions	Dyad description
Primary Dyads		
Love	Joy + Trust	A positive sentiment reflecting confidence and satisfaction in financial decisions or market performance.
Alarm	Fear + Surprise	A sudden realization of potential financial risks or unexpected market downturns.
Disappointment	Surprise + Sadness	The feeling when financial expectations aren't met, often due to unforeseen market changes.
Optimism	Anticipation + joy	Expectation of positive financial outcomes, driven by favourable market indicators.
Secondary Dyads		
Guilt	Joy + Fear	Regret over missed financial opportunities despite overall positive market conditions.
Curiosity	Trust + Surprise	Interest in exploring new financial ventures or strategies after unexpected market data.
Despair	Fear + Sadness	A sense of hopelessness in financial situations, often during prolonged market slumps.
Unbelief	Surprise + Disgust	Scepticism towards financial data or market anomalies that seem too outlandish.
Envy	Sadness + Anger	Discontent seeing others' financial success, especially if feeling left behind in market trends.

Cynicism	Disgust + Anticipation	Doubtful anticipation of financial outcomes, often due to past market disappointments.
Pride	Anger + Joy	Satisfaction in financial achievements, especially after overcoming market adversities.
Hope	Anticipation + Trust	Positive outlook on future financial endeavours, trusting in market stability.
Tertiary Dyads		
Delight	Joy + Surprise	Elation from unexpected positive financial outcomes or market boons.
Shame	Fear + Disgust	Regret over financial missteps, especially if they were avoidable.
Outrage	Surprise + Anger	Strong displeasure at unforeseen financial injustices or market manipulations.
Pessimism	Sadness + Anticipation	Expectation of negative financial trends, often based on current market downturns.
Anxiety	Anticipation + Fear	Worry over potential financial risks or upcoming market uncertainties.
Opposite Dyads		
Ambivalence	Trust + Disgust	Mixed feelings about financial decisions, torn between trust in data and aversion to certain market aspects.
Confusion	Surprise + Anticipation	Uncertainty about future financial directions due to unexpected market data.

2.2. Exploration of Existing Datasets

Before we dive into creating our own temporality and sentiment analysis datasets, it's important to understand the landscape of what's already out there. By exploring existing datasets, we can assess current standards, identify gaps. In this section, we'll take a deep dive into the datasets currently available for temporality and sentiment analysis, laying the groundwork for our subsequent data augmentation and creation efforts.

2.2.1. Temporality analysis dataset

Within the field of temporality analysis, there seems to be a noticeable lack of comprehensive research. Despite the intriguing nature of the field, our research revealed a limited number of papers and datasets dedicated to this specific task. The only two significant contributions we were able to uncover are detailed in the following sections.

This observation highlights that research in temporality analysis is still in its early stages, with a notable absence of well-constructed and tested datasets. Given this context, our research aims to pioneer in this field, making the task of building a reliable dataset both challenging and ground-breaking

TempoWordNet for Sentence Time Tagging

The field of temporal analysis has seen some advances with the introduction of TempoWordNet [9], a time-tagged version of WordNet. This dataset has been meticulously constructed using a method known as distant supervision, which uses pre-defined rules or heuristics to automatically generate labelled training data. The primary goal of TempoWordNet is to capture the intricate temporal nuances inherent in linguistic expressions, trying to fill the gap between natural language processing, historical linguistics, and the study of language development.

TempoWordNet stands out for its approach of enriching WordNet datasets with temporal dimensions. Each dataset entry is tagged with one of four temporal dimensions: atemporal, past, present, or future. This enrichment process was initiated with a manual selection of seed entries, followed by iterative classification processes to ensure a comprehensive representation of temporal concepts.

While the paper focuses primarily on the methodology behind TempoWordNet, it also highlights the potential applications of such a dataset. The ability to determine the time period in which a text was written can provide invaluable insights for NLP systems, especially those dealing with time-sensitive / time-evolutive input.

True to its name, the TempoWordNet dataset consists mainly of words, as can be seen in the Figure 3. While the insights and results derived from this word-centric approach are undeniably insightful, the dataset in its current form doesn't quite fit our specific needs.

Past			
Top 10		Bottom 10	
Word (Sense)	Cat.	Word (Sense)	Cat.
by (1)	adv.	iguanodon (1)	n.
recently (1)	adv.	ground-shaker (1)	n.
in the first place (1)	adv.	diplodocus (1)	n.
remember (3)	v.	saurischian (1)	n.
old (1)	n.	argentinosaur (1)	n.
old (1)	adj.	ornitischian (1)	n.
old (2)	adj.	titanosaur (1)	n.
old (6)	adj.	mellowing (1)	n.
erstwhile (1)	adj.	appearance (4)	n.
honest to god (1)	adj.	psychosexuality (1)	n.
Present			
Top 10		Bottom 10	
Word (Sense)	Cat.	Word (Sense)	Cat.
immediately (1)	adv.	overstay (1)	v.
now (3)	adv.	visit (7)	v.
presently (2)	adv.	run (30)	v.
present (3)	n.	drag on (1)	v.
immediate (3)	adj.	wear (6)	v.
instant (2)	adj.	crawl (3)	v.
attendant (1)	adj.	bond (3)	v.
ever-present (1)	adj.	ramp (5)	v.
here (1)	adj.	stand back (2)	v.
omnipresent (1)	adj.	line up (3)	v.
Future			
Top 10		Bottom 10	
Word (Sense)	Cat.	Word (Sense)	Cat.
prophecy (1)	n.	example (4)	n.
prefiguration (2)	n.	referral (1)	n.
prognosis (1)	n.	palmist (1)	n.
prophecy (2)	n.	sibyl (1)	n.
meteorology (1)	n.	prophetess (1)	n.
fortunetelling (1)	n.	augur (1)	n.
extropy (1)	n.	sibyl (2)	n.
horoscope (1)	n.	onomancy (1)	n.
guess (2)	n.	arithmancy (1)	n.
credit rating (1)	n.	lithomancy (1)	n.

Figure 3: Extract of the TempoWordNet Dataset

It does, however, provide a valuable foundation from which to draw. Inspired by the words and their associated temporal categories in TempoWordNet, we plan to create our own sentences tailored to the financial domain. These sentences will include all the temporalities we have described in the Temporality Analysis Dataset section, thus ensuring a domain-specific and comprehensive representation of temporality for our classifier.

Temporal classification for historical Romanian texts

The study of temporality in linguistic expressions has been enriched by the research presented in the paper Temporal classification for historical Romanian texts [10]. This research delves deep into the intersection of natural language processing, historical linguistics and the study of language evolution, with the aim of identifying the period in which a text was written.

The main objective of the work is to develop a system capable of recognising the temporal nuances of historical Romanian texts. By understanding and classifying these texts according to the time of their creation, the research offers insights into the diachronic[1] dynamics of the Romanian language. Such a system promises to transfer modern linguistic resources and tools to historical domains, thereby bridging the gap between contemporary and historical linguistic studies.

Although the specific details of the construction of the dataset were not explicitly stated in the first sections of the paper, the dataset would include historical Romanian texts spanning centuries. The principle of capturing subtle changes in linguistic features over time remains consistent, whether it's classifying by centuries or by shorter timeframes, like weeks. This offers valuable insights into the evolution of the language and allows a classification based on the periods they were written. While the timespan of centuries presents a different context than

[1] **Diachronic**: Pertaining to the study of language change over time, often involving the comparison of different historical stages of a language or languages.

weeks, the foundational concept of temporal classification remains the same, making the distinction less critical for our purposes.

The paper highlights the solvability of supervised temporal text classification across genres and authors with a high degree of accuracy.

In conclusion, regarding the dataset from the Romanian temporal classification study, it's important to note that we will not be incorporating it directly into our work. As the dataset is not publicly available and its domain is very different from our focus, it doesn't fit seamlessly with our objectives. However, the good results achieved on such a nuanced task, where subtle distinctions exist between text classes, strengthen our confidence in the feasibility of our temporal classification toolkit.

2.2.2. Sentiment analysis dataset

In the vast field of sentiment analysis datasets, a trend has emerged: many available datasets are limited to mere word collections for each sentiment class, or they present a paltry number of examples without substantial context. Although such simplistic datasets might prove beneficial in certain situations, they lack the necessary robustness and comprehensiveness for our project's objectives. Moreover, the lack of accompanying research papers or extensive validation of these datasets raises questions about their dependability and quality.

Nevertheless, in this context, two datasets stand out. Not only do they contain substantial amounts of information, but they are also supported by research papers that demonstrate their quality and suitability. In the subsequent sections, we will examine these two datasets closely, analysing their compositions, advantages, and ways in which we can incorporate and utilize them in our Dataset Creation/Augmentation CLI.

DENS: A Dataset for Multi-class Emotion Analysis

In the field of sentiment analysis, the Dataset for Emotions of Narrative Sequences (DENS) is an innovative dataset tailored for multi-class emotion analysis of long narratives written in English based on the Plutchik's wheel of emotions. This dataset was introduced in the paper titled DENS: A Dataset for Multi-class Emotion Analysis [11] and brings together narratives from two different sources: the classics texts accessible on Project Gutenberg and the more actual and “realistic” narratives from Wattpad. The thorough annotation process utilised Amazon Mechanical Turk to ensure a comprehensive portrayal of emotions in the narratives.

The development of DENS was motivated by the need to capture the diverse emotions reflected in human language, particularly in narratives. Prior advancements in natural language processing have typically focused on shorter texts such as product reviews and tweets, but have sometimes struggled with length limitations, intent ambiguity, and a lack of emotional depth. DENS concentrates on intricate narratives that are infused with intense emotions, striving to bridge this interdisciplinary divide.

Delving into the structure of DENS, the dataset consists of passages extracted from works of fiction, encompassing both classical literature and contemporary tales written in English. These extracts are self-contained units, incorporating multiple sentences and exploring a diverse range of topics as we can see in Figure 4. Each narrative snippet is diligently annotated according to one of the nine emotional classes. To guarantee the dependability of the annotations, each sample includes an indicator that emphasises agreement between annotators.

Text	Label
I found this was a little too close upon him, but I made it up in what follows. He stood stock-still for a while and said nothing, and I went on thus: "You cannot," says I, 'without the highest injustice, believe that I yielded upon all these persuasions without a love not to be questioned, not to be shaken again by anything that could happen afterward. If you have such dishonourable thoughts of me, I must ask you what foundation in any of my behaviour have I given for such a suggestion?"	Angry
She stretched hers eagerly and gratefully towards him. What had happened? Through all the numbness of her blood, there sprang a strange new warmth from his strong palm, and a pulse, which she had almost forgotten as a dream of the past, began to beat through her frame. She turned around all a-tremble, and saw his face in the glow of the coming day.	Anticipation
Ah! That moving procession that has left me by the road-side! Its fantastic colors are more brilliant and beautiful than the sun on the undulating waters. What matter if souls and bodies are failing beneath the feet of the ever-pressing multitude! It moves with the majestic rhythm of the spheres. Its discordant clashes sweep upward in one harmonious tone that blends with the music of other worlds–to complete God's orchestra.	Joy

Figure 4: Extract of the DENS Dataset

DENS's emotional classification is inspired by a basic version of Plutchik's wheel of emotions considering the 8 basic emotions. The empirical assessment of DENS, as stated in the paper, highlights the effectiveness of the fine-tuned pre-trained BERT model. When this model was utilised with the dataset, it achieved a good average micro-F1 score of 60.4%. These findings highlight the potential of the dataset as a new approach in emotion analysis, emphasising the need to move beyond conventional sentence-based methods for the best results.

Based on the results of this study, two important conclusions were drawn to guide our data set construction. First, the remarkable success of the fine-tuned BERT model cannot be ignored. Its potential is significant, and we look forward to further exploration as we tackle the sentiment analysis toolkit. Secondly, the concept of implementing multi-sentence narratives is intriguing. However, it is not suitable for the requirements of our project as we are focusing on sentence-based analysis. This approach is consistent with the entity detection already developed and the temporality and sentiment classification that will be developed in the future. Nevertheless, the multi-sentence narratives have provided ideas for our CLI dataset generation. We are likely to adopt some of these concepts, such as splitting our dataset entries into multiple examples and incorporating financial aspects into our dataset construction.

CARER: Contextualized Affect Representations for Emotion Recognition

The dataset associated with the CARER paper [12] is an important addition to sentiment analysis field, specifically crafted to capture the intricate linguistic nuances that exist within emotional expressions. The dataset was created using a methodology known as distant supervision, utilising noisy labels to collate emotion data. The main aim of CARER is to precisely picture and comprehend emotions presented in text, considering the diverse and subtle ways in which emotions are expressed, influenced by individual experiences, knowledge, and beliefs.

The dataset incorporates context-dependent, pattern-oriented emotional characteristics extracted from sentences, each linked to a particular emotional label. These characteristics are subsequently enhanced with word embeddings, maintaining the semantic connections between patterns. This representation proves particularly advantageous in sentiment analysis, where traditional methods encounter difficulties in capturing the dynamic linguistic forms that are common in opinionated content.

CARER takes a different approach from traditional sentiment analysis techniques, which tend to limit their analysis to the sentence level. Instead, CARER dives deeper to capture the complete range of emotional expressions in text. The dataset comprises labelled sentences,

while the paper's emotion recognition methodology may have used advanced techniques like network-based analysis.

The CARER dataset underwent comprehensive evaluation through various emotion detection models, including online classifiers and deep learning models as shown on this page[2]. This highlighted the dataset's effectiveness and potential in emotion recognition. In Figure 5, you can find an example of some entries of the dataset that is used in the CARER paper.

i didnt feel humiliated	0 (sadness)
i can go from feeling so hopeless to so damned hopeful just from being around someone who cares and is awake	0 (sadness)
im grabbing a minute to post i feel greedy wrong	3 (anger)
i am ever feeling nostalgic about the fireplace i will know that it is still on the property	2 (love)
i am feeling grouchy	3 (anger)
ive been feeling a little burdened lately wasnt sure why that was	0 (sadness)
ive been taking or milligrams or times recommended amount and ive fallen asleep a lot faster but i also feel like so funny	5 (surprise)

Figure 5: Extract of the CARER Dataset

In summary, this dataset provides an encouraging path for researchers and practitioners within the field of sentiment analysis who aim to capture and represent emotions more effectively.

Based on the findings of this study, two significant conclusions came up to guide our dataset creation. Firstly, the dataset's excellent performance across a variety of models, including BERT, is consistent with our previous findings and underscores our desire to further explore BERT for sentiment classification purposes. The paper's methodology represents each dataset sample as a single sentence or a group of words in opposition to multi-sentence narrative samples, which aligns with our intended approach. This congruence allows us to adapt examples from their dataset by introducing financial nuances. This means that we plan to enrich some samples from their dataset with financial references, which could include domain-related terms, stocks, tickers, and other related jargon. These modifications will ensure that the data match with the financial domain, offering contextually relevant samples for sentiment analysis. The intention behind these adjustments is to craft a foundation for the information or prompts that will be utilized to create our bespoke dataset, making it more tailored to financial sentiment analysis. While their dataset could potentially be used directly for emotion classification, introducing these financial nuances ensures that the emotions are contextualized within the realm of financial news, enhancing the relevance and accuracy of the analysis.

[2] https://paperswithcode.com/sota/text-classification-on-emotion

2.2.3. Conclusion

In our exploration of existing datasets in temporality and sentiment analysis, we've explored through a rich variety of research. Our journey has highlighted the current standards, introduced some methodologies, and highlighted the potential gaps that exist in these fields.

From our in-depth study of temporality analysis datasets, we identified a general lack of comprehensive research specifically tailored to the financial field. However, the datasets we encountered, such as TempoWordNet and the study of historical Romanian texts, provided valuable insights. They underlined the importance of capturing temporal nuances, whether it's over centuries or more precise time periods. Although the scope of our project requires a more granular temporal classification, the basic principles we observed remain consistent.

When we turned our attention to sentiment analysis datasets, we found a much more diverse landscape. While many datasets were relatively simple, lacking depth and validation, there were some exceptions, such as DENS and CARER. These datasets, supported by their respective research, demonstrated the subtleties of sentiment representation in text. Both datasets resonated with our idea of a sentiment analysis toolkit, highlighting the importance of sentence-level sentiment representation to ensure greater granularity in sentiment analysis of a full text. We were particularly inspired by these datasets to adopt a BERT-based approach to classification, given its consistent success across multiple studies, but this will be discussed in more detail later in this report.

In terms of actionable insights, our dataset construction will be heavily influenced by the methods and structures we observe. We aim to use a sentence-based approach for both temporal and sentiment analysis. For sentiment analysis, we will enrich some examples from existing datasets with financial nuance to form the basis of the prompt we will use to create our bespoke dataset. This will involve adapting examples from existing datasets, integrating financial terms, stocks, tickers and other domain-specific jargon to ensure relevance. For the temporal analysis, we will take less inspiration from the existing dataset, but will focus on adding financial terms and creating some relevant examples to use in the final dataset creation. Such modifications will help us to create a dataset that's not only tailored for sentiment or temporality analysis, but also imbued with a financial context.

2.3. Exploration of Existing Textual Dataset Augmentation Techniques

Exploring advanced augmentation techniques has become essential to improve the quality and diversity of textual datasets. This section provides a comprehensive review of existing methods and strategies that use state-of-the-art language models and innovative approaches to dataset creation and augmentation.

Through our research, we have identified three main papers that stand out for their contributions to the field. These papers not only provide insights into current best practices, but also highlight the challenges and potential solutions in the field of dataset augmentation.

By analysing the methodologies, results and implications of these studies, we aim to gain valuable knowledge that will inform and shape our Dataset Creation/Augmentation CLI. The following subsections provide a detailed analysis of each paper, highlighting their core techniques, findings and relevance to our project objectives.

2.3.1. WANLI: Worker and AI Collaboration for Natural Language Inference Dataset Creation

This paper uses GPT-3 [13], an older version of a state-of-the-art language model (GPT-4) developed by OpenAI, to build the dataset. GPT-3's strength lies in its ability to generate open-ended text that is often indistinguishable from human-written content. The choice of GPT-3 is motivated by its ability to replicate a pattern given only a few examples in context. This generative capability, combined with its extensive training data, allows it to produce diverse and contextually relevant examples that can be used to augment existing datasets.

The dataset creation process in WANLI is a collaboration between AI and human annotators. Starting with an existing dataset, MultiNLI for Natural Language Inference (NLI), the approach uses dataset cartography[3] to automatically identify examples that demonstrate challenging reasoning patterns. Once these patterns are identified, GPT-3 is instructed to compose new examples that follow similar patterns. These machine-generated examples undergo an automatic filtering process inspired by the data maps. After filtering, human annotators are tasked with assigning a gold label to these examples and have the option to revise them if necessary. This combination ensures that the examples generated are both diverse (thanks to GPT-3) and of high quality (due to human evaluation).

The paper emphasises the empirical strengths of the resulting dataset, WANLI, over existing NLI datasets. However, the specific metrics used to evaluate "dataset pertinence" are not detailed in this paper. The paper does mention the use of out-of-domain test sets, such as HANS and Adversarial NLI, to evaluate the performance of models trained on WANLI.

The resulting dataset, WANLI, contains 107,885 NLI examples. Remarkably, training a model on WANLI led to improved performance on eight out-of-domain test sets, including an 11% improvement on HANS and a 9% improvement on Adversarial NLI. This was achieved despite WANLI being significantly smaller than MultiNLI. Furthermore, WANLI proved to be more effective than MultiNLI even when the latter was augmented with other NLI datasets.

A major challenge in crowdsourcing large NLP datasets is the lack of linguistic diversity, as human authors often rely on repetitive patterns. This leads to models that may perform well on

[3] https://aclanthology.org/2020.emnlp-main.746.pdf

in-domain test sets but are not performing nicely on out-of-domain or adversarial examples. The WANLI approach aims to address this by combining the strengths of both machine-generated content and human evaluation.

2.3.2. DReCa: A General Task Augmentation Strategy for Few-Shot Natural Language Inference

This paper presents DReCa [14] that uses a BERT model, a transformer-based architecture, to embed textual examples. BERT, developed by Google, has been pre-trained on large amounts of text and is known to generate meaningful embeddings for textual data. These embeddings serve as the basis for the subsequent steps in the DReCa approach. The paper emphasises the use of a fine-tuned BERT model, which means that the model has been further trained on specific datasets related to the task at hand, enhancing its ability to generate contextually relevant embeddings.

The core methodology of DReCa is to decompose existing datasets into latent reasoning categories. These categories can be different syntactic constructs or semantic categories such as quantifiers and negation. The decomposition process starts by embedding the textual examples using the fine-tuned BERT model. Once the embeddings are generated, k-means clustering is applied to these embeddings to refine the original tasks. This clustering process aims to identify and group similar reasoning patterns within the data. The identified clusters are then used as the basis for constructing additional few-shot classification tasks, effectively augmenting the original task distribution using GPT-3.

The paper adapts the classic sinusoidal regression problem[4] to evaluate the effectiveness of meta-learning techniques in an NLP setting. While the specific metrics used to evaluate the pertinence of the dataset are not detailed in it, the paper mentions accuracy as a primary metric, especially when assessing the improvement of DReCa augmented meta-learners over baseline models in natural language inference tasks.

The results of the study demonstrate the effectiveness of the DReCa approach. In the adapted sinusoidal regression problem, standard meta-learning methods struggled to adapt. However, a model that meta-learned about the underlying inference types showed significant improvement. In the context of natural language inference (NLI), meta-learners augmented with DReCa outperformed baselines by 1.5-4 accuracy points on four separate NLI few-shot problems. This improvement was achieved without the need for domain-specific engineering or the addition of unlabelled data.

A significant challenge in meta-learning for NLP is the lack of a well-defined set of tasks that correspond to reusable skills. This often leads to the practice of treating entire data sets as tasks, which can lead to overfitting. DReCa addresses this challenge by decomposing datasets into latent reasoning categories, thereby increasing the quality and quantity of tasks available for meta-learning. The paper also highlights the inherent challenges posed by the heterogeneity of NLP datasets and the limited number of supervised datasets available for any given NLP problem.

[4] https://www.statisticshowto.com/sinusoidal-regression/

2.3.3. Textual Data Augmentation for Efficient Active Learning on Tiny Datasets

In this paper [15], they use GPT-2, a generative language model developed by OpenAI, for data augmentation. GPT-2, even though it is an older model, it is known for its ability to generate coherent and contextually relevant text, making it an ideal choice for generating artificial text examples if you are looking for an opensource model. Nevertheless, its performances are not as good as more recent models like GPT-4, but they weren't available at the time of the study. The study transforms the data generation task into an optimisation problem, aiming to maximise the usefulness of the generated output. The optimisation strategy employed is Monte Carlo Tree Search (MCTS), a heuristic search algorithm well known for its applications in decision making, particularly in the domain of games.

The core methodology of the paper is to automatically generate artificial text examples that can enrich an existing dataset. This is achieved by passing the output of GPT-2 through a search procedure, specifically using MCTS. A new set of examples is first generated, then manually labelled and added to the original training set. The classifier is then retrained on this extended training set. This process is repeated in active learning cycles using only a subset of the generated data. The selection of examples is based on their informative value, using entropy as a measure.

The paper adopts an active learning[5] approach where the selection of examples is based on their informativeness. In this context, entropy is used as a measure of informativeness. Entropy measures the uncertainty or surprise of an example, with higher entropy examples being considered more informative. These examples are assumed to add more information to an existing classifier, thereby improving its performance.

Using MCTS for data augmentation increased performance by 26% on the TREC-6 Questions dataset and 10% on the Stanford Sentiment Treebank SST-2 dataset. When compared to a Non-Guided Data Generation (NGDG) process that doesn't optimise for a reward function, the MCTS approach achieved performance improvements of 3% and 5% on TREC-6 and SST-2, respectively.

A major challenge in NLP is the complexity of language, which makes data augmentation a difficult task. Traditional active learning often assumes the availability of large pools of unlabelled data. However, in scenarios where data is limited, the onus shifts to the data collection process. The paper's approach addresses this by generating artificial text examples, reducing the need for manual data creation or collection from real-world interactions. The paper emphasises the real-world scenario where available data may be insufficient to run a typical active learning algorithm, and presents a method to minimise human intervention in data generation.

[5] https://teaching.cornell.edu/teaching-resources/active-collaborative-learning/active-learning#:~:text=Active%20learning%20methods%20ask%20students,words%20through%20writing%20and%20discussion.

2.3.4. Conclusion

The review of existing textual data augmentation techniques highlights the central role of advanced NLP models, particularly the GPT series, in the field of data creation and augmentation. The studies reviewed here have convincingly demonstrated the effectiveness of models such as GPT-3 in generating diverse, contextually relevant and high-quality examples that can significantly enhance existing datasets. In particular, the WANLI approach, which synergises the generative capabilities of GPT-3 with human evaluative strengths, exemplifies the potential of such collaborative efforts to produce datasets that are both rich and of superior quality.

However, a key takeaway from these studies is the indispensability of high-quality examples to provide context to those LLM. While models such as GPT-3 and GPT-2 have demonstrated remarkable generative capabilities, their performance is often dependent on the quality and relevance of the prompts provided (this includes the "instruction" and the "context"). This underlines the importance of careful prompt engineering to elicit the desired outputs from these models. In addition, the studies also highlight the occasional need for manual annotation or verification, emphasising that while automation can significantly speed up the dataset creation process, human oversight remains crucial to ensure the integrity and quality of the data.

Existing techniques, such as dataset cartography and Monte Carlo tree search, offer promising ways of optimising the data generation process without imposing undue constraints. These methods not only increase the diversity of examples generated, but also ensure their relevance and usefulness for the intended tasks.

In light of these findings, to develop our dataset generation/augmentation CLI, we will leverage the generative capabilities of GPT-4, the latest in the GPT series, to ensure state-of-the-art performance. In the development process, we will integrate the lessons learned from these studies, emphasising the importance of quality prompts and the occasional need for manual intervention. By combining the strengths of advanced NLP models with the lessons learnt from existing techniques, we aim to create a tool that is both powerful and user-friendly, ensuring optimal dataset augmentation with minimal constraints.

2.4. CLI Procedure Specification

In this section we delve into the details of our Command Line Interface (CLI), designed for creating and extending datasets. We will outline the architecture that supports the CLI, detail its operational workflow, and provide specifications that guide its functionality.

2.4.1. Architecture

The architecture of our CLI is designed to be scalable and modular. It is packaged in a Docker container, ensuring a consistent environment that is both isolated and reproducible. This containerized approach allows for easy deployment and compatibility across different platforms and systems.

Our CLI is part of the whole ecosystem we will build during this project. This ecosystem will use docker-compose to run all the elements. This docker-compose setup is global across our project, and as the number of bricks grows, we will add more containers and volumes to this composition. This approach ensures that as we develop additional tools or functionality during this project, they can be seamlessly integrated into our existing infrastructure without disrupting the current workflow.

At its core is the Docker Compose setup, which acts as an orchestration hub for various Docker containers. The CLI for Dataset Augmentation container is where our dataset creation and augmentation processes will run. This container communicates with an external API to leverage the capabilities of the GPT-4 model. The volume bound to this container ensures that datasets, whether new or augmented, are stored, and can be easily accessed or modified.

2.4.2. Workflow and Sequence

The primary function of the CLI is to facilitate the creation and extension of datasets, and its operation begins with user input. The user is required to provide a configuration file in TOML format (i.e. create it, place it on the volume and then provide the path to the CLI). This configuration file is essential to meet the user's needs, as it contains all the necessary parameters and directives for the CLI to function optimally. It specifies paths to essential files such as the new dataset file, any existing dataset file, human-written examples and a context description. It also provides a detailed description of the examples or task types that the GPT-4 model is expected to generate. The configuration file also specifies the number of samples required. This structured approach ensures that the user has granular control over the dataset generation process and can tailor it to their specific needs.

At launch and upon user request, the CLI loads and reads the provided configuration file. This step is fundamental as it sets the stage for subsequent operations and ensures that all user specifications are captured and ready for processing. Once the configuration file has been parsed, the CLI creates a prompt tailored to the GPT-4 model to obtain the best results. This prompt will be constructed based on the information extracted from the configuration file, ensuring that the speech model receives clear and concise instructions.

Once the prompt is ready, the CLI interacts with the GPT-4 model via an API. This interaction is iterative, with the CLI making as many calls as necessary to generate the desired number of samples. Each call results in the generation or augmentation of the dataset, with the GPT-4 model generating contextually relevant and coherent samples based on the prompt provided.

The final step in the workflow is for the CLI to store the generated data. Depending on the user's specifications in the configuration file, the CLI will either create a new dataset file or append the generated samples to an existing dataset. This flexibility ensures that users can either start afresh with a new dataset or add augmented data to their existing datasets.

Following this breakdown, in Figure 6 is a more visual representation of the workflow, captured in a diagram that illustrates the entire process.

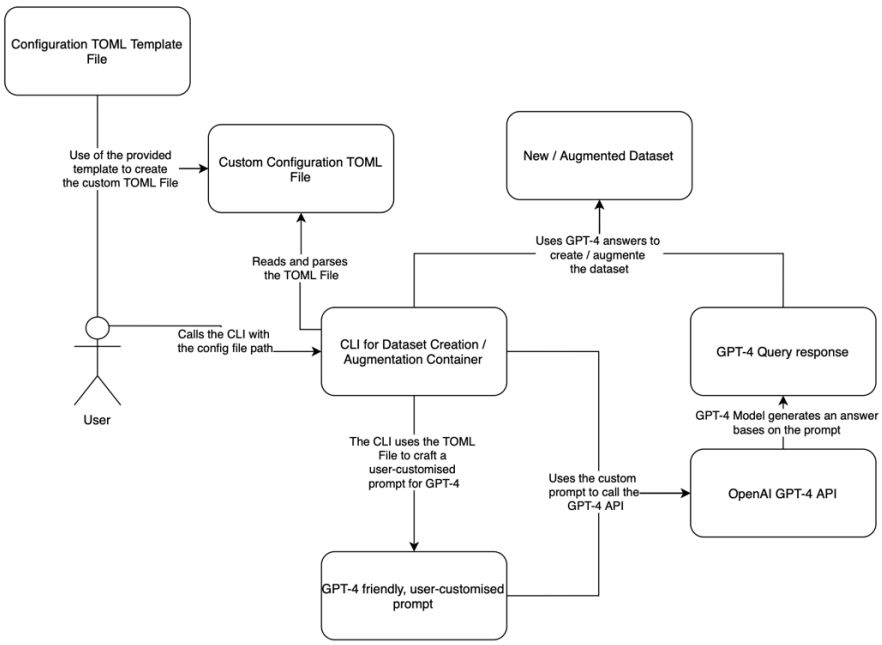

Figure 6: CLI Workflow and Procedure

2.4.3. Configuration TOML File

In the field of software configuration, TOML (Tom's Obvious, Minimal Language) stands out for its clarity and simplicity. For our CLI, the TOML file serves as the primary configuration mechanism, allowing users to specify various parameters and preferences for creating or extending datasets. Let's look at each section of the TOML file that will be needed.

Dataset Information

This section provides a general overview of the context of the dataset.

- **context_description**: A brief description that sets the stage for the type of content that will be generated, ensuring that the model has a clear understanding of the overarching theme or domain.
- **result_path**: A path for the CLI output. It can be an already existing file where the new content will be appended or a new file that will only contain the new content

Classes

As datasets often encompass multiple classes or categories. This section allows users to specify details for each class.

- **class_name**: The name of the class or category within the dataset.
- **class_description**: A detailed description of what each record or sample in this class should encapsulate.
- **number_of_samples**: The amount of content to be generated for this particular class.
- **path_to_written_examples**: A path to human-written examples for this class, which will be used to craft the prompt.

Generation Details

This section contains directives for the actual generation process.

- **language**: Specifies the desired language for the dataset.
- **temperature**: A parameter that controls the randomness of the model's output. Higher values make the output more random, while lower values make it more deterministic.

TOML Example

The example TOML file in Figure 7 instructs the CLI to generate financial news articles with a focus on stock market trends, divided into positive and negative sentiment. The model is guided by human-written examples and is set to generate content in English with a temperature of 0.4.

```toml
[dataset_information]
context_description = "Financial news articles focusing on stock market trends."
result_path = "/path/to/results.csv"

[[classes]]
class_name = "Positive"
class_description = "Articles that portray a positive sentiment towards stock market trends."
number_of_samples = 1000
path_to_written_examples = "/path/to/positive_examples.csv"

[[classes]]
class_name = "Negative"
class_description = "Articles that portray a negative sentiment towards stock market trends."
number_of_samples = 100
path_to_written_examples = "/path/to/negative_examples.csv"

[generation_details]
language = "English"
temperature = 0.4
```

Figure 7: TOML Configuration File example

To simplify the process of creating datasets and to ensure that users can easily configure their requirements, a TOML template is provided with the container. This template acts as a basic blueprint to guide users in specifying their dataset preferences without having to start from scratch. By following the structure of this template, users can easily tailor the parameters to their specific needs.

2.4.4. Conclusion

In this section, we have described in detail the architecture, workflow and configuration specifics of our Command Line Interface (CLI), which is tailored for dataset creation and augmentation. The modular and scalable architecture, encapsulated in a Docker container, not only ensures reproducibility and compatibility, but also paves the way for future integrations and extensions. The user-centric design, which relies on a TOML configuration file, provides users with granular control over the dataset generation process, allowing for a tailored experience that meets their unique requirements. By leveraging the power of the GPT-4 model and incorporating human-written examples, our CLI promises to deliver high quality, contextually relevant datasets. The provision of a TOML template underscores our focus on ease of use, ensuring a seamless and intuitive dataset creation journey. As we move forward, this CLI will serve as a foundational tool in our arsenal, facilitating the creation of robust and diverse datasets that meet a wide range of research and application needs, but specifically sentiment and temporal analysis datasets for our project.

2.5. CLI Implementation

The command-line interface (CLI) implementation is a fundamental part of our project, enabling smooth interaction with GPT-4 and facilitating dataset generation. The CLI is designed to be modular and scalable, ensuring that each component serves a specific purpose, from configuration handling to communication with the OpenAI API. This section delves into the details of the CLI, explaining its different components, their interdependencies, and their collective role in creating a functional CLI.

2.5.1. ConfigHandler

The ConfigHandler class acts as a gateway to our configuration management, specifically designed to read and parse the TOML configuration files. These configuration files contain critical metadata about the dataset we wish to create and ensure that the CLI behaves in a manner consistent with our requirements.

The full code for this implementation can be found in in the Appendix section A1. config_handler.py.

The main responsibilities of the ConfigHandler class include:

- **Loading the TOML configuration:** Using the toml library, the configuration file provided during initialisation is loaded and parsed, making the data easily accessible for subsequent processes.
- **Extract dataset information:** This involves retrieving general information about the dataset, such as context descriptions and the path where the dataset results will be stored.
- **Gather Class Details:** The configuration file can specify multiple classes, each with its attributes such as class name, description, number of samples required and paths to any pre-written examples. The ConfigHandler processes this information and organises it for easy retrieval.

Here under in Figure 8: TOML configuration file loading is a visual representation of the mapping between the TOML configuration file and the ConfigHandler class.

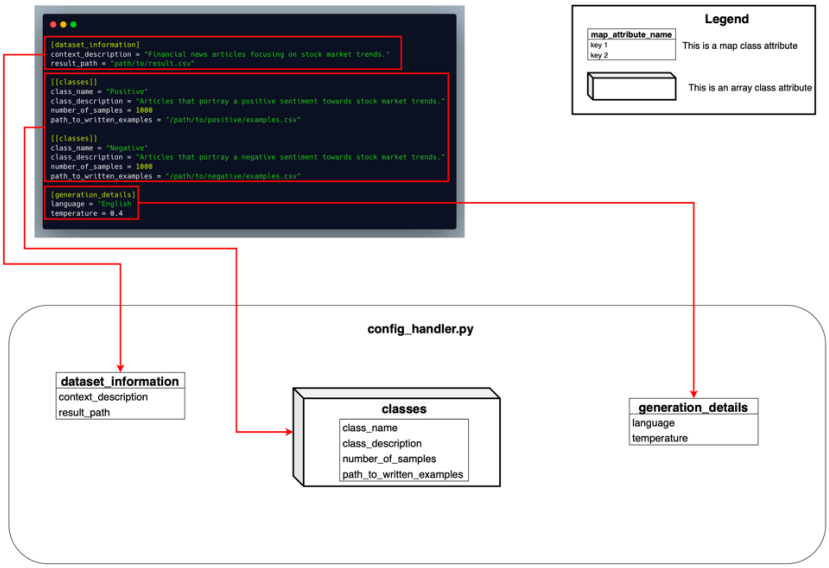

Figure 8: TOML configuration file loading

2.5.2. Dataset

Ensuring the orderly arrangement, durable preservation, and cost-effective creation of our dataset is critical to the success of our CLI implementation. With these goals in mind, we have implemented the Dataset class. This class has been carefully designed to monitor and store our data as we interact with GPT-4 during the dataset creation phase.

The full code for this implementation can be found in the Appendix section A2. dataset.py.

The main responsibilities of the Dataset class include:

- **Initialization with Existing Data:** One of the main aspects of our dataset management is the avoidance of redundancy. Upon instantiation, the `Dataset` class checks the specified result path to determine if a dataset file already exists. Should such a file exist, the existing data is integrated into the current class instance, ensuring that we have a continually evolving dataset without the risk of duplicate entries, lack or overflow of data.
- **Class Management:** Our dataset, by design, can span multiple classes, each holding information. The Dataset class is built with functionality to easily add new classes, as well as mechanisms to retrieve the number of samples already curated within a given class.
- **Data Addition and Export:** To offer an easy dataset grows, the Dataset class offers methods for the addition of entries to any specified class. Post data accumulation, the class provides an export function, turning our data to a CSV file. This not only ensures the persistence of our data but also offers it in a universally recognized format.

2.5.3. DiscussionHandler

Central to the process of creating the dataset is our ability to communicate effectively with the GPT-4 model. This capability is encapsulated in our DiscussionHandler class, which acts as the hub for managing the flow of dialogue with GPT-4. Through this class, we generate data samples tailored to our specific requirements, as outlined in our configuration.

The full code for this implementation can be found in the Appendix section A3. discussion_handler.py.

The main responsibilities of the DiscussionHandler class include:

- **Configuration initialisation:** At the start, the DiscussionHandler class is initialised with our specified configurations, extracted from the provided TOML file by the ConfigHandler class. This ensures that our interactions with GPT-4 are based on our predefined dataset criteria.
- **Facilitating OpenAI API interactions:** Within its scope, the DiscussionHandler class integrates the functionality of the OpenAIApiCaller class (detailed in 2.5.4). This integration facilitates communication with GPT-4, allowing us to forward messages and receive model responses.
- **Dataset Supervision:** This class holds an instance of the Dataset class. This ensures the accurate and efficient storage of samples generated during our interactions with GPT-4.
- **GPT-4 response parsing:** Responses from GPT-4 are processed using the OpenAiChatParser class (detailed in 2.5.5), which parses the model's response to extract the dataset samples.

- **Interactive session management:** The DiscussionHandler class is responsible for the entire cycle of dialogue with GPT-4. This includes crafting discussions that are contextually rich yet concise, ensuring budgetary efficiency. It ensures that the desired number of samples is achieved, decides when to end the session, and is adept at dealing with unforeseen interruptions or crashes.

Here under in Figure 9: DiscussionHandler flow is a visual representation of the flow of the DiscussionHandler and how it interacts with the other components of our CLI.

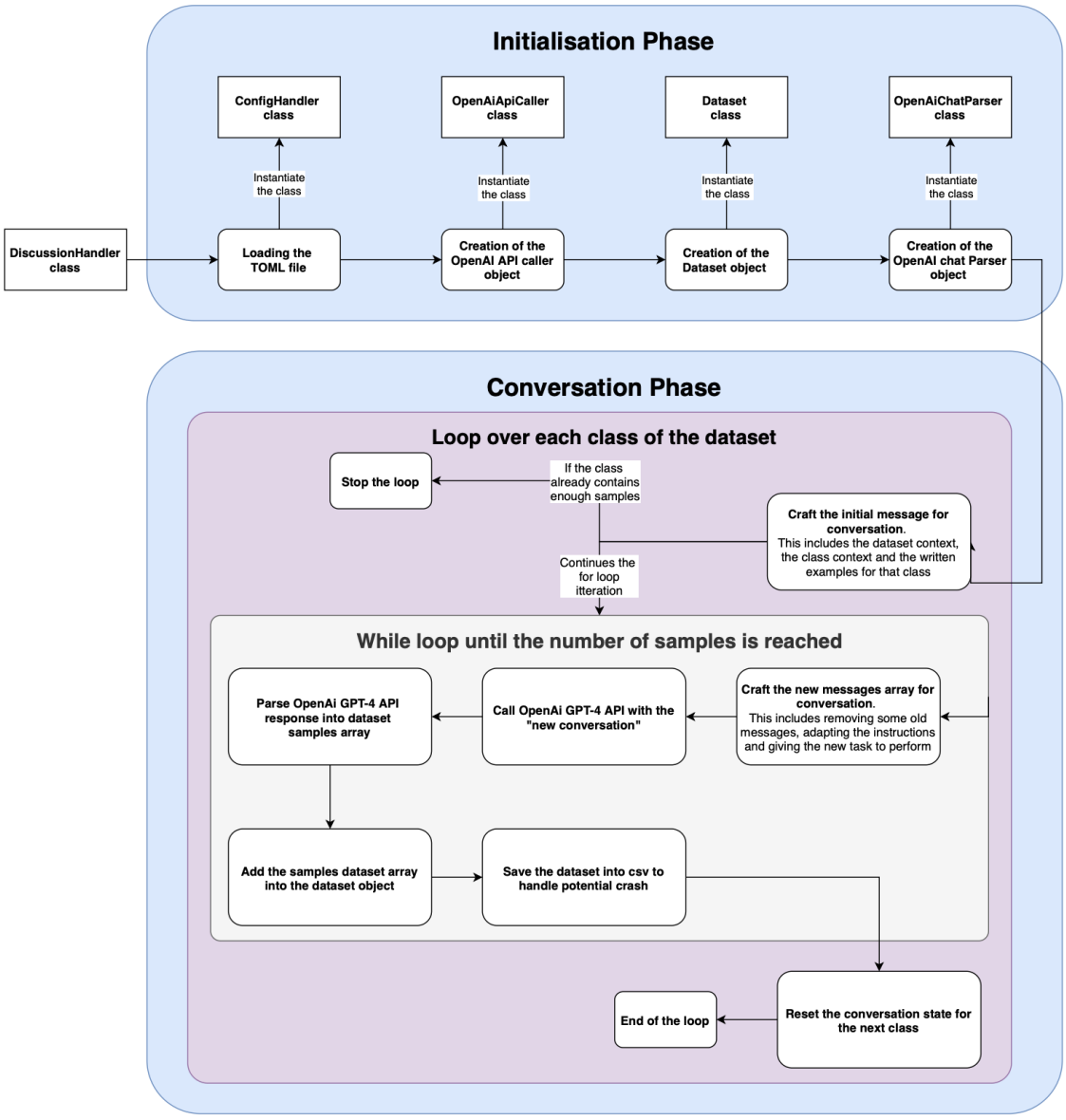

Figure 9: DiscussionHandler flow

2.5.4. OpenAiApiCaller

The data generation process relies on our interaction with GPT-4 to be as seamless and efficient as possible. To manage this essential aspect, the OpenAiApiCaller class has been designed to encapsulate the logic associated with interfacing with the OpenAI API. This ensures that our dataset creation process remains decoupled from the specifics of the API, allowing for greater scalability and flexibility.

The full code for this implementation can be found in the Appendix section A4. openai_api_caller.py.

The main responsibilities of the OpenAiApiCaller class include:

- **API initialisation:** Upon instantiation, the class initialises the OpenAI API, making it ready for subsequent interactions. This is achieved by setting the API key and specifying the model, in this case GPT-4.
- **Message Management:** One of the main tasks of this class is to manage the structure and flow of messages between the user and the assistant. To maintain the context of a conversation, the class keeps track of the message history, ensuring that each API call contains relevant previous interactions.
- **API Interaction:** The class contains a method for calling the chat API. This method handles various potential exceptions that may occur during the interaction, such as timeouts, rate limits, and other API-related errors. To handle these exceptions, the class is equipped with built-in retry mechanisms, including context-aware wait times.
- **Message Reset:** After each complete conversation cycle, it's important to clear the history in order to start a new interaction. The OpenApiCaller class provides this reset functionality.

2.5.5. OpenAiChatParser

The responses from the GPT-4 model, although delivered as structured strings, require skilful processing to transform them into usable data samples for our dataset. Given that GPT-4, as a language model, naturally produces string outputs, there's a need to process these outputs to extract our desired array of dataset samples. To deal with this complexity, we've created the OpenAiChatParser class. This class is carefully designed to parse, extract and organise the data, ensuring that our interactions with the model produce structured and usable results.

The full code for this implementation can be found in the Appendix section A5. openai_chat_parser.pyA4. openai_api_caller.py.

The main responsibilities of the OpenAiChatParserclass include:

- **Text extraction:** The class can identify and extract individual items or entries from a given response. These items are typically delimited by a 'number dot space' pattern, making them distinct and identifiable. This pattern is what we've asked the model for in our prompt, and that's why we get a formatted response. The extract_items method is designed to perform this operation, using regular expressions to capture the desired pattern and then extract the items.
- **Response parsing:** One of the primary functions of this class is to parse the raw response from the GPT-4 model. The parse_response method uses the text extraction capability of the class to parse the response and return a structured set of individual items. This structured output becomes an integral part of our dataset.

By isolating the parsing logic in its class, we achieve a modular design, ensuring that any changes to the response format or additional processing requirements can be easily integrated without affecting other components of the CLI.

2.5.6. Bringing It All Together: The Main

While the classes above function as individual units, they are orchestrated in the `main.py' script. This script acts as the entry point to our CLI, tying together the functionality of each class and controlling the process of creating datasets.

The full code for this implementation can be found in the Appendix section

A6. main.pyA4. openai_api_caller.py.

The main functions of the `main.py' script include:

- **Argument parsing:** The script uses the argparse module to capture command line arguments, in particular the path to the TOML configuration file, which is essential for the dataset creation process.
- **Discussion Initialisation and Start:** Using the `DiscussionHandler` class, the script initiates the interactive sessions with GPT-4. This includes providing the model with the necessary context, guiding the generation of samples based on the defined classes, and populating the dataset by starting the "discussion" with GPT-4 API.
- **Dataset testing handling:** Using the DatasetTester class, depending on the input of the user, the main.py will start the testing process of the dataset to evaluate their quality.

Through this modular and systematic approach, our CLI provides a robust, flexible and scalable solution for dataset creation, opening the way for further enhancements and adaptability to different requirements.

2.6. CLI Testing

2.6.1. Testing Procedure

Introduction to the testing method

To ensure the robustness and semantic consistency of the datasets created by our CLI tool, an advanced testing procedure has been designed. This procedure leverages the newly developed DatasetTester class, which serves as a cornerstone for dataset validation. The complete code for this class can be found in the datatest_tester.py (refer to Appendix A7. datatest_tester.py).

Testing with Multiple NLP Models

The DatasetTester is designed to use three distinct NLP models to ensure various encodings, and make sure that the results have no bias linked to the choice of the model. For that we used the three most used models for sentence embeddings. There are detailed here-under:

- **Sentence Transformers:** Utilizes the underlying 'all-mpnet-base-v2' model for generating embeddings and calculating similarity scores across data samples.
- **BERT:** Employs a BERT model for producing context-aware embeddings contributing to the nuanced similarity calculations.
- **spaCy:** Leverages the underlying 'en_core_web_lg' model for its efficiency in generating vectors and measuring similarity.

Each of these models provides a unique lens through which the data can be analyzed, capturing a diverse range of semantic relationships within the dataset to measure it's robustness.

Detailed Testing Procedure

As mentioned above, the DatasetTester class is engineered to conduct a rigorous validation of datasets, with the ability to discern and quantify semantic distinctions among various classes of data. Here is a step-by-step breakdown of the sophisticated procedure that the class employs:

- **Initialization:** Upon instantiation, the class loads the dataset from the specified CSV file, readying itself for a sequence of in-depth analyses. It sets up a framework to meticulously record the outcomes of each test it performs.
- **Embedding Generation:** Using one of the three models specified above, the DatasetTester systematically encodes textual data into these embeddings, setting the stage for comparative analysis.
- **Cosine Similarity Computation:** For each class, the DatasetTester calculates cosine similarities within the embeddings, which helps in assessing the homogeneity of samples within the same class. By examining these intra-class relationships, it provides insights into the consistency of semantic encoding across each category.
- **Statistical T-Test Execution:** The nuanced aspect of this testing procedure lies in its statistical validation. It executes t-tests between all combinations of class embeddings. This is not just a comparison of means; it is a hypothesis testing to determine if there are statistically significant differences between the semantic representations of classes. The t-test is adept at revealing subtle yet important distinctions, providing a robust statistical basis for the semantic integrity of the dataset.
- **Results Aggregation and Visualization:** The outcomes of these tests are aggregated into a comprehensive DataFrame. This DataFrame tabulates the t-statistics and p-values, offering a clear numerical summary of the similarities and differences between classes. Subsequently, for clarity and accessibility, the DataFrame is exported as an

image, which encapsulates the entirety of the statistical analysis in a visually digestible format. This step ensures that the results are not only precise but also presentable and ready for review.

To launch the analysis with all three models, we rely on the main.py implementation that can be found in the Appendix section

A6. main.py. This main.py will take the dataset file given by the user and the output directory for the images and perform the tests with the three models using the DatasetTester class.

2.6.2. Testing Results

In this section, we will present the results of our dataset testing, which was conducted to evaluate the semantic coherence of the generated samples. The evaluation was performed using the procedure detailed in section 2.6.1.

Before we dive deeper into the dataset analysis, here is an explanation of what the T-Test results tables will be. Here is an explanation of the format and the significance of each column:

- **Class_A and Class_B:** These columns list pairs of classes that have been compared against each other. Each row represents a unique pairing, indicating that the model has computed how semantically similar or distinct these two classes are within the dataset.
- **T-Statistic:** The T-Statistic column reports the calculated statistic from the t-test for each class pair. This value measures the difference in means relative to the variation observed in the samples. A high absolute value indicates a significant difference between the two classes. Positive values suggest that the mean similarity within Class_A is greater than Class_B, and negative values suggest the opposite.
- **P-Value:** This column provides the p-value from the t-test, which informs us of the statistical significance of the observed differences. A p-value below 0.01, as seen throughout this table, indicates that the difference in semantic similarities between classes is statistically significant. This p-value threshold suggests a less than 1% probability that the observed differences are due to a random chance.

In our use case, our goal is to have distinct classes that are well-separated, then a high absolute T-Statistic and a low P-Value (below 0.01) are desirable, as they indicate clear differentiation between classes.

Temporal Analysis Dataset

Bert Embedding T-Test Analysis

	Class_A	Class_B	T-Statistic	P-Value
0	Distant Past	Recent Past	22.49	<0.01
1	Distant Past	Present	104.69	<0.01
2	Distant Past	Near Future	132.37	<0.01
3	Distant Past	Distant Future	100.40	<0.01
4	Recent Past	Present	125.91	<0.01
5	Recent Past	Near Future	152.14	<0.01
6	Recent Past	Distant Future	121.45	<0.01
7	Present	Near Future	35.81	<0.01
8	Present	Distant Future	2.45	0.01
9	Near Future	Distant Future	37.18	<0.01

Figure 10: Bert Embedding T-Test for Temporal Dataset

In the image presented in the Figure 10: Bert Embedding T-Test for Temporal Dataset, the T-test analysis describes a comprehensive comparison between the temporal classes within the dataset. It is noteworthy that all T-statistic values, with the exception of the pairing of 'Present' with 'Distant Future', are significantly below the 0.01 level, indicating robust statistical significance in the semantic distinctions between classes. The pairing of 'Present' and 'Distant Future' yields a P-value of 0.01, which, although higher than the others, is still within an acceptable threshold for rejecting the null hypothesis of no difference.

The magnitude of the T-statistics is compelling, with predominantly high absolute values, suggesting a substantial separation between the time classes being compared. Nevertheless, certain pairings, namely 'Distant Past' with 'Recent Past', 'Present' with 'Near Future', and 'Present' with 'Distant Future', register comparatively moderate T-statistics. This may indicate a closer semantic relationship or a less pronounced differentiation between these specific temporal categories in the embeddings of the not fined-tuned BERT model.

The analytical results suggest that the dataset, encoded via a non-fine-tuned BERT model, has a pronounced potential for temporal classification tasks. The strong semantic contrasts, as evidenced by the high T-statistic values in most of the class comparisons, underline the inherent ability of the model to effectively discriminate between different temporal contexts. This bodes well for future model fine-tuning, suggesting that with appropriate adjustments, the discriminative power of the model could be further enhanced for improved temporal classification.

Spacy Embedding T-Test Analysis

	Class_A	Class_B	T-Statistic	P-Value
0	Distant Past	Recent Past	46.71	<0.01
1	Distant Past	Present	93.20	<0.01
2	Distant Past	Near Future	108.96	<0.01
3	Distant Past	Distant Future	171.40	<0.01
4	Recent Past	Present	150.61	<0.01
5	Recent Past	Near Future	165.43	<0.01
6	Recent Past	Distant Future	235.55	<0.01
7	Present	Near Future	20.37	<0.01
8	Present	Distant Future	86.04	<0.01
9	Near Future	Distant Future	61.99	<0.01

Figure 11: Spacy Embedding T-Test for Temporal Dataset

The Figure 11: Spacy Embedding T-Test for Temporal Dataset provides a detailed account of the T-test results between different temporal classes within the evaluated dataset. Each T-statistic value significantly exceeds the critical alpha level of 0.01, confirming that the semantic differences between class pairings are statistically significant. These results suggest a clear separation between temporal contexts, a crucial factor for the effectiveness of temporal classification models.

An examination of the T-statistic column reveals predominantly high absolute values, indicating a strong semantic divergence between most of the class comparisons. An exception is the class pairing 'present' and 'near future' which, although exceeding the significance threshold, has a relatively modest T-Statistic of 20.37. This suggests a lower degree of differentiation between these particular temporal classes, suggesting that the not fine-tuned spacy model has trouble differentiating.

These results provide preliminary evidence that the dataset is suitable for temporal classification tasks, even without fine-tuning the underlying Spacy encoding. The inherent structure of the dataset seems to naturally facilitate the discrimination of temporal categories, as evidenced by the significant T-statistic values. The data promises to provide a solid basis for subsequent fine-tuning aimed at refining these distinctions and optimising the model for temporal classification performance.

Sentence-Transformer Embedding T-Test Analysis

	Class_A	Class_B	T-Statistic	P-Value
0	Distant Past	Recent Past	309.14	<0.01
1	Distant Past	Present	328.12	<0.01
2	Distant Past	Near Future	181.85	<0.01
3	Distant Past	Distant Future	296.53	<0.01
4	Recent Past	Present	14.82	<0.01
5	Recent Past	Near Future	135.06	<0.01
6	Recent Past	Distant Future	21.08	<0.01
7	Present	Near Future	152.23	<0.01
8	Present	Distant Future	36.64	<0.01
9	Near Future	Distant Future	117.39	<0.01

Figure 12: Sentence-Transformer Embedding T-Test for Temporal Dataset

In the Figure 12: Sentence-Transformer Embedding T-Test for Temporal Dataset, the T-test results underline the distinctiveness of temporal class pairings within our dataset. The P-values are consistently below the stringent 0.01 threshold, rejecting the null hypothesis in all class comparisons. This strong statistical evidence supports the premise that each temporal category has unique semantic properties that can be distinguished by our current model.

The T-statistic values provide additional insights; while most pairings have high absolute T-statistics, indicating pronounced semantic separations, a few pairings - 'Recent Past' with 'Present', 'Recent Past' with 'Distant Future', and 'Present' with 'Distant Future' - have relatively lower values. Although these scores still indicate significant differences, their relative magnitude is reduced compared to other class pairings. This may indicate a more nuanced or subtle semantic distinction between these temporal frames that isn't completely captured by the not fine-tuned sentence transformer.

This analytical perspective suggests that the non-fine-tuned sentence transformer encoding effectively captures temporal semantic differences to a degree that is statistically meaningful. Even in the absence of fine-tuning, the sentence transformer model shows considerable capacity for distinguishing between temporal contexts, and thus holds promising potential for improved performance in temporal classification tasks with further model refinement.

Conclusion

The full T-test analyses carried out on various temporal class pairings, as documented in Figure 10, Figure 11 and Figure 12, using the BERT, Spacy and Sentence-Transformer embeddings respectively, have yielded statistically significant results, supporting the suitability of the current dataset for temporal classification tasks. Despite the lack of fine-tuning, the embeddings demonstrate a robust ability to capture and discriminate semantic distinctions across temporal contexts, an essential feature for our upcoming temporal analysis models.

Sentimental Analysis Dataset

Bert Embedding T-Test Analysis

Figure 13: Bert Embedding T-Test for Sentimental Dataset

	Class_A	Class_B	T-Statistic	P-Value
0	Joy	Trust	102.44	<0.01
1	Joy	Fear	353.31	<0.01
2	Joy	Surprise	104.82	<0.01
3	Joy	Sadness	115.47	<0.01
4	Joy	Disgust	407.80	<0.01
5	Joy	Anger	83.27	<0.01
6	Joy	Anticipation	166.77	<0.01
7	Trust	Fear	295.00	<0.01
8	Trust	Surprise	18.91	<0.01
9	Trust	Sadness	21.36	<0.01
10	Trust	Disgust	352.17	<0.01
11	Trust	Anger	7.19	<0.01
12	Trust	Anticipation	85.24	<0.01
13	Fear	Surprise	250.31	<0.01
14	Fear	Sadness	267.43	<0.01
15	Fear	Disgust	37.38	<0.01
16	Fear	Anger	275.49	<0.01
17	Fear	Anticipation	202.22	<0.01
18	Surprise	Sadness	0.10	0.92
19	Surprise	Disgust	298.95	<0.01
20	Surprise	Anger	23.04	<0.01
21	Surprise	Anticipation	57.15	<0.01
22	Sadness	Disgust	320.78	<0.01
23	Sadness	Anger	25.32	<0.01
24	Sadness	Anticipation	62.29	<0.01
25	Disgust	Anger	326.02	<0.01
26	Disgust	Anticipation	250.05	<0.01
27	Anger	Anticipation	81.94	<0.01

In the analysis results shown in Figure 13: Bert Embedding T-Test for Sentimental Dataset, the T-test evaluation of the Bert Embeddings for Sentiment Classification provides a consistent comparison between different sentiment classes within the dataset. It is noticeable that almost all T-statistic values fall below the 0.01 threshold, indicating a significant statistical differentiation between most of the sentiment classes. The exception is the comparison between 'Surprise' and 'Sadness', where the P-value is significantly higher at 0.92, suggesting a potentially weaker sentiment discrimination provided by the non-fine-tuned BERT model for these particular emotions.

The T-statistics are largely robust, confirming the presence of distinct semantic separations across the sentiment classes. However, lower T-statistics for pairings such as 'Trust' with 'Surprise', 'Sadness' and 'Anger', as well as 'Fear' with 'Disgust' and 'Surprise' with 'Sadness' and 'Anger', suggest more nuanced differences that the current state of the model may not capture as effectively.

Overall, the results reflect positively on the suitability of the dataset for sentiment classification tasks, suggesting that the embedded sentiments are sufficiently distinct to be categorised with a high degree of accuracy. The high T-statistics for most pairings highlight the innate ability of the BERT model to discriminate between complex emotional states. This performance is indicative of the potential improvements achievable through model fine-tuning, which could unlock even more nuanced sentiment discrimination capabilities in the dataset.

Spacy Embedding T-Test Analysis

Figure 14: Spacy Embedding T-Test T-Test for Sentimental Dataset

	Class_A	Class_B	T-Statistic	P-Value
0	Joy	Trust	108.38	<0.01
1	Joy	Fear	144.77	<0.01
2	Joy	Surprise	62.24	<0.01
3	Joy	Sadness	141.50	<0.01
4	Joy	Disgust	131.16	<0.01
5	Joy	Anger	3.86	<0.01
6	Joy	Anticipation	135.28	<0.01
7	Trust	Fear	28.42	<0.01
8	Trust	Surprise	49.81	<0.01
9	Trust	Sadness	28.66	<0.01
10	Trust	Disgust	12.52	<0.01
11	Trust	Anger	104.95	<0.01
12	Trust	Anticipation	19.36	<0.01
13	Fear	Surprise	83.29	<0.01
14	Fear	Sadness	1.18	0.24
15	Fear	Disgust	17.94	<0.01
16	Fear	Anger	141.29	<0.01
17	Fear	Anticipation	9.72	<0.01
18	Surprise	Sadness	81.72	<0.01
19	Surprise	Disgust	67.93	<0.01
20	Surprise	Anger	58.51	<0.01
21	Surprise	Anticipation	73.66	<0.01
22	Sadness	Disgust	18.48	<0.01
23	Sadness	Anger	138.09	<0.01
24	Sadness	Anticipation	10.56	<0.01
25	Disgust	Anger	127.55	<0.01
26	Disgust	Anticipation	7.91	<0.01
27	Anger	Anticipation	131.77	<0.01

The T-test analysis Figure 14: Spacy Embedding T-Test T-Test for Sentimental Dataset in displays the Spacy Embeddings for Sentiment Classification reveals a meticulous differentiation between the different emotional states encoded in the dataset.

In particular, the P-values uniformly underscore significant statistical separations of 0.01, with the sole exception of the combination of 'Fear' and 'Sadness', where the P-value exceeds the conventional significance level and rests at 0.24.

The distribution of T-statistics, mostly between 50 and 150, suggests that while there is a noticeable separation between most pairs of sentiments, the distinctions are not as pronounced as they might be with a fine-tuned model. Of note are the relatively low T-statistics observed for pairings such as 'Joy' with 'Anger', 'Trust' with 'Fear', 'Sadness' and 'Disgust', as well as 'Fear' with 'Sadness', 'Disgust' and 'Anticipation', and similarly muted figures for 'Sadness' with 'Anticipation' and 'Disgust' with 'Anticipation'. These suggest subtle nuances in the emotional content that the current state of the non-fine-tuned Spacy model may not fully capture.

Despite these subtleties, the results tend to support the suitability of the dataset for sentiment classification. The evidence of statistically significant differences for the majority of class comparisons indicates a basic effectiveness of the Spacy model in distinguishing between sentiments. This is a promising starting point, suggesting that subsequent model refinement and fine-tuning could significantly improve the detection of subtler emotional nuances, thereby increasing the accuracy of sentiment classification.

Sentence-Transformer Embedding T-Test Analysis

Figure 15: Sentence-Transformer Embedding T-Test for Sentimental Dataset

	Class_A	Class_B	T-Statistic	P-Value
0	Joy	Trust	113.24	<0.01
1	Joy	Fear	330.59	<0.01
2	Joy	Surprise	458.07	<0.01
3	Joy	Sadness	259.93	<0.01
4	Joy	Disgust	394.89	<0.01
5	Joy	Anger	101.20	<0.01
6	Joy	Anticipation	339.27	<0.01
7	Trust	Fear	216.69	<0.01
8	Trust	Surprise	326.82	<0.01
9	Trust	Sadness	139.77	<0.01
10	Trust	Disgust	267.63	<0.01
11	Trust	Anger	207.66	<0.01
12	Trust	Anticipation	208.96	<0.01
13	Fear	Surprise	85.03	<0.01
14	Fear	Sadness	86.55	<0.01
15	Fear	Disgust	31.42	<0.01
16	Fear	Anger	415.53	<0.01
17	Fear	Anticipation	31.67	<0.01
18	Surprise	Sadness	185.43	<0.01
19	Surprise	Disgust	57.75	<0.01
20	Surprise	Anger	548.66	<0.01
21	Surprise	Anticipation	130.27	<0.01
22	Sadness	Disgust	127.15	<0.01
23	Sadness	Anger	352.27	<0.01
24	Sadness	Anticipation	62.89	<0.01
25	Disgust	Anger	485.79	<0.01
26	Disgust	Anticipation	69.69	<0.01
27	Anger	Anticipation	434.70	<0.01

The statistical overview of the sentence-transformer embeddings for sentiment classification shown in the Figure 15: Sentence-Transformer Embedding T-Test for Sentimental Dataset reflects a high degree of statistical significance across the board, as evidenced by the uniformly low P-values, which are all below the 0.01 threshold. This suggests a reliable level of discrimination between the pairs of sentiments when classified with the non-fine-tuned sentence-transformer model.

Focusing on the T-statistics, we observe robust values for almost all comparisons, suggesting a strong separation between most emotional states. A few exceptions to this pattern are the pairs 'fear' with 'disgust' and 'anticipation', as well as 'surprise' with 'disgust', where the T-statistics are relatively low. These lower values may indicate a more subtle differentiation between these emotions within the dataset as the one captured by the model in its current form.

Despite these exceptions, the results appear to be favourable for sentiment classification tasks. The high T-statistics prevalent in the data reinforce the model's ability to significantly discriminate between most sentiments. The presence of more subtle distinctions in some sentiment pairings, which are not as well captured by the model, may be due to the subtleties of linguistic expression, which require a fine-tuned model to detect. Nevertheless, the overall strong performance of the dataset suggests that it is well suited to sentiment classification, even with a model that has not been fine-tuned, and promises further improvements with additional model optimisation.

Conclusion

The full T-test analyses carried out on various sentimental class pairings, as documented in Figure 13, Figure 14 and Figure 15, using the BERT, Spacy and Sentence-Transformer embeddings respectively, have yielded statistically significant results, supporting the suitability of the current dataset for sentimental classification tasks. Despite the lack of fine-tuning, the embeddings demonstrate a robust ability to capture and discriminate semantic distinctions across sentimental contexts, an essential feature for our upcoming sentimental analysis models.

3. Text segmentation module

3.1. Introduction to Text Segmentation

Text segmentation, a central aspect of Natural Language Processing (NLP), plays a vital role in breaking down long financial news articles into usable chunks for various analysis tasks, including, in our case, Named Entity Recognition (NER), sentiment analysis and temporal analysis. Without segmentation, analysing these articles would be challenging due to their potential complexity, which includes multiple sentiments, temporalities, entities, and other complicated elements. It's a sophisticated task that requires an understanding of the structure, context and nuances of the language being used. In the field of NLP, text segmentation is fundamental to several higher-level tasks such as text summarisation, topic modelling, sentiment analysis, and information retrieval, allowing these tasks to be performed with greater accuracy and efficiency. Jurafsky and Martin highlight the importance of this process in the broader context of understanding and processing human language [16].

The importance of text segmentation lies in its ability to make unstructured text understandable and manageable for both algorithms and humans. In the context of machine learning and AI, text segmentation allows text to be represented in a structured way, making it easier to process and analyse large amounts of data [17]. For humans, well-segmented text improves readability and comprehension, especially in situations where rapid scanning or review is required.

The complexity of text segmentation comes from the intricacies of human language, which is inherently ambiguous and context-dependent, with different syntactic and semantic rules. A robust text segmentation system must recognise these linguistic nuances, accurately identify the end of one thought and the beginning of another, distinguish between segment types (such as headings, paragraphs, sentences), and handle different writing styles and formats. In this paper [18], they discuss the challenges of normalising non-standard words in text, which is a critical aspect of effective text segmentation.

The relevance of text segmentation extends to different types of text, such as news articles, academic papers, legal documents and social media posts, each of which requires a specific segmentation approach due to their unique structural and stylistic features. Lee, Przybocki, & Narayanan [19] addressed the task of segmenting broadcast news transcripts into a sequence of stories, demonstrating the application of text segmentation in the media.

The development of digital communication has brought new challenges to text segmentation. With the rise of online content, there's an increasing use of informal language, emojis, hashtags, and other internet-specific linguistic phenomena. Baldwin, T., Cook, P., Lui, M., MacKinlay, A., and Wang, L [20] explored how the linguistic noise in social media text differs across different sources, highlighting the need for text segmentation tools to continuously evolve to effectively handle such diverse and dynamic content.

In summary, text segmentation is an essential step in the NLP pipeline, setting the stage for all subsequent analysis and interpretation. Its ability to break text down into logical and coherent units not only facilitates a range of NLP applications, but also has a significant impact on the quality of the insights that can be derived from text data. As we continue to generate and rely on massive amounts of textual information, the role of sophisticated and accurate text segmentation becomes increasingly important, making it a critical area of focus for NLP and AI.

3.2. Our need for custom Text Segmentation

In the pursuit of granular time and sentiment analysis in addition to NER, the need for advanced text segmentation is essential. This need comes from the need to break down complex sentences into distinct sections that could encapsulate different temporal contexts and emotional sentiments. Standard sentence segmentation, which typically divides text based solely on syntactic markers such as punctuation, proves inadequate for our purposes. This is because a single sentence can often contain multiple temporal references, entities, or different sentiments, making simple segmentation insufficient for deep analysis.

Moreover, this refined segmentation is critical for effective data analysis in our scenario where the accuracy of temporal and sentiment interpretation can significantly impact the user feedback. In our case, aka financial news analysis, understanding the specific timing of events or shifts in market sentiment is essential for accurate forecasting and strategy formulation.

That's why our custom text segmentation module is not just a technical enhancement; it's a crucial tool that enables more nuanced and precise analysis of text data. By accurately segmenting text into granular sections based on temporal and emotional content, we can extract richer insights and make more informed decisions based on the complex and multifaceted nature of human language.

3.3. Methodology and Implementation

To perform our Text Segmentation, we create the article_processor.py class. The full code for this implementation can be found in the Appendix section A8. article_processor.py.

To perform text segmentation, the first step is to load the Spacy NLP model, which is specifically optimised for processing English text. This is a deliberate choice, as the model's sophisticated understanding of English linguistics is crucial for accurately interpreting the structure and subtleties of the language. Once the model is loaded and initialised, the process moves on to sentence tokenisation. This uses Spacy's advanced tokenisation capabilities to meticulously segment the text into individual sentences. This segmentation is based not only on punctuation, but also on linguistic structure, ensuring that each sentence is a complete and coherent unit of thought.

Once the text has been segmented into sentences, the next step is to identify the sections within those sentences. This is done by breaking down the sentences based on conjunctions and punctuation. These linguistic elements are often key indicators of shifts in ideas, topics or sentiments within a sentence. By identifying and using these markers, the process can effectively break down a sentence into smaller segments, each of which encapsulates a unique aspect of the overall meaning or intent of the sentence.

An integral part of our segmentation strategy is a detailed analysis of the verbs within the segments. Verbs are central to conveying actions and states, and their presence significantly affects the context and meaning of a segment. Our methodology pays particular attention to this, ensuring that each segment is analysed in terms of its verb content to accurately reflect the intended message and context.

Through this detailed approach, our text segmentation methodology efficiently processes complex texts, breaking them down into nuanced and contextually relevant segments. This segmentation is tailored to our specific analytical needs, such as detailed temporal and sentiment analysis, ensuring that the segmented text is optimally structured for subsequent processing and interpretation. Below, in Figure 16: Text Segmentation Process is a visual

representation of the segmentation process and which parts were provided by Spacy and which parts were developed during this project.

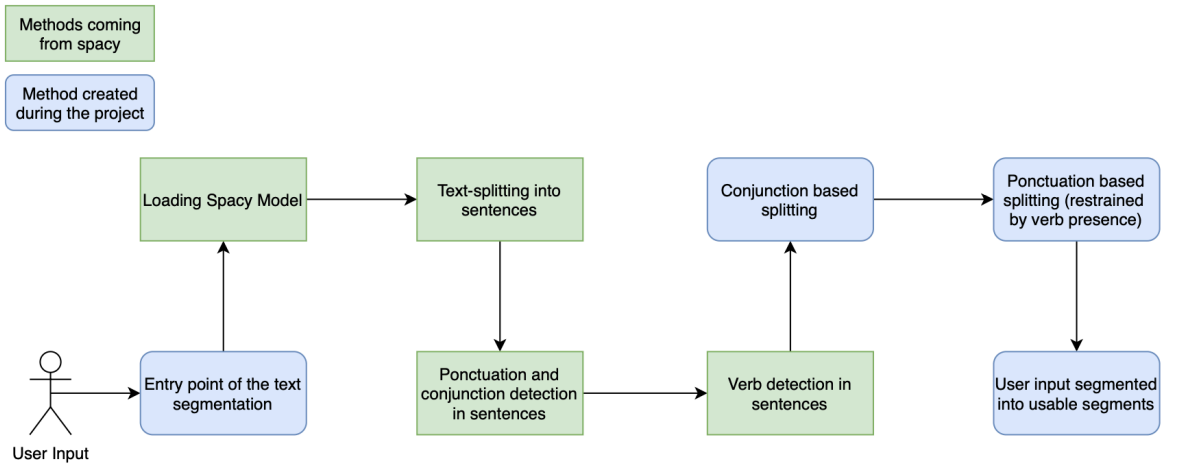

Figure 16: Text Segmentation Process

3.4. Testing the module

3.4.1. Testing Procedure

The testing of the text segmentation module was carefully designed to simulate real-world conditions and assess the performance of the module in different linguistic scenarios. Central to this testing was the creation and use of a specialised dataset, built through a manual process.

Dataset Creation

In order to test the text segmentation module, a dataset of 50 sentences was constructed. These sentences were manually selected from a wide range of news articles from a variety of sources, including Yahoo Finance, Bloomberg and other reputable financial news sources. During the dataset creation, the sentences were chosen for their complexity, length and the wide range of topics they covered, from technology to healthcare, ensuring that the dataset reflected a realistic and challenging range of linguistic scenarios. Adjustments were also made to the sentences to increase the richness of the dataset, such as changing the names of stocks, or the adjectives used to describe them. This editing aimed to ensure that the dataset was not biased towards any topic, such as technology, and thus provided a multi-facetted evaluation dataset for this module.

Each selected sentence was then manually segmented into its constituent parts. This manual segmentation process was essential as it provided a clear, precise reference for evaluating the accuracy of the module's segmentation capabilities. Through this process, the dataset serves as a representative baseline against which the performance of the text segmentation module is measured, ensuring a thorough and nuanced assessment of its capabilities across a wide range of linguistic scenarios.

Implementation of the Testing Framework

A custom Python testing script was developed to evaluate the module. This script was designed to process each sentence from the dataset, apply the text segmentation module, and then compare the output with the manually segmented reference. The main objective was to measure the accuracy of the module's performance, defined as the proportion of sentences where the module's segmentation accurately matched the manual segmentation. The framework was also

equipped to log and categorise any errors in the segmentation, allowing a detailed analysis of the types of errors and their frequency. To perform this testing, we created the article_segmentation_tester.py class. The full code for this implementation can be found in the Appendix section A9. article_segmentation_tester.py

Automated and Manual Testing

The testing process included both automated and manual components. The automated tests provided a broad assessment of the module's performance across the dataset, providing insight into general trends and overall accuracy. In parallel, a manual review was carried out to investigate the module's output, focusing in particular on instances where there were differences between the module's segmentation and the manual reference. This dual approach of automated and manual testing allowed for a comprehensive evaluation of the module, identifying not only its strengths but also areas for refinement. In particular, the manual review provided valuable insight into the context and nuances of the module's performance, revealing subtleties that purely automated methods might miss.

Through this structured and multi-faceted testing process, the module was rigorously evaluated, ensuring its robustness and reliability in real-world applications. The knowledge gained from this process has been used to further develop and optimise the module, improving its accuracy and adaptability to different text segmentation tasks.

3.4.2. Testing Results

In the rigorous pursuit of achieving the best Text Segmentation module, achieving good testing results is an important part of our development process. This section describes the results of a series of tests designed to push the limits of the module's capabilities, revealing both its strengths and areas for improvement.

Quantitative Results

Testing on a carefully curated dataset of 50 different samples yielded an encouraging 96.0% accuracy rate. This figure is not only a testament to the robustness of the module, but also an indicator of its ability to skilfully process a wide range of sentence structures. This high level of accuracy underlines the effectiveness of the underlying algorithms and their implementation. The quantitative analysis, focusing primarily on the rate of agreement between the module's output and expertly crafted reference segments, provides a solid basis for the module's reliability in practical applications.

Qualitative Analysis and Case Studies

Beyond the numbers, qualitative analysis delves into the nuances of the module's performance. It examines the context and complexity of sentences where the module did not match the expected segmentation. Two cases highlighted specific challenges with the module's current segmentation logic:

- **Case 1: "Oil giants pivot to renewable energy thanks to growing concerns about climate change"**

  The module's handling of sentences, particularly when it interprets "thanks to" as a single unit rather than segmenting at that phrase, highlights a potential gap in its ability to recognise less conventional segmentation patterns. This observation suggests that improving the module's linguistic models could be beneficial, particularly in recognising and accurately interpreting phrases such as 'thanks to' which, despite some

misconceptions, are valid in English (although not very formal) and act as central conjunctions in sentence structure.

- **Case 2: "The textile industry faces challenges related to ethical sourcing concerns".**
  Again, the module's failure to segment at the conjunction "related to" highlights the potential need for refinement in the processing of sentences with subtle conjunction-based segmentation points.

Despite the specific instances discussed above where the module deviated from expected behaviour, it is important to note that such deviations (around 5% of the cases) do not necessarily compromise the usefulness of the module in practical scenarios. For example, sentences structured as 'Automotive stocks dip as electric vehicle competition intensifies globally' present a distinct analytical challenge when treated as a monolithic block. However, the module's ability to segment it into "Automotive stocks dip" and "Electric vehicle competition intensifies globally" is consistent with the desired functionality, demonstrating its practical value in real-world applications.

These results, both quantitative and qualitative, provide a comprehensive picture of the module's capabilities. They provide a clear direction for potential targeted enhancements and confirm the module's readiness for use in scenarios aligned with our use cases. The next steps include addressing the identified limitations and further refining the module's segmentation logic to improve both its accuracy and its adaptability to complex sentence structures. Through this iterative process of testing and refinement, we aim to increase the performance of the module to meet our needs of text segmentation in financial news natural language processing.

3.5. Conclusion

The development and testing of our text segmentation module has demonstrated its effectiveness and potential. By achieving a high accuracy rate of 96.0% in our tests, the module has shown its ability to handle a wide variety of sentences, making it a valuable addition to our NLP toolkit. While there are areas for improvement, as highlighted by the instances where segmentation did not match expectations, these are not significantly detrimental to our current use case. Furthermore, the design of the module allows for future enhancements, ensuring its adaptability and relevance as language use and analysis needs evolve.

Another feature of our module is its accessibility and integration capability. Using Flask, the module is exposed as a service via simple HTTP requests, making it easily connectable to other services of our projects. This flexibility facilitates seamless integration into broader NLP and data analysis pipelines, increasing its utility and applicability.

In conclusion, our text segmentation module is a robust and adaptable tool in the field of NLP. Its current performance is satisfying, and its architecture allows for continuous improvement and integration, promising to remain a significant contributor to our NLP efforts.

4. Temporality Classification Toolkit

In the field of financial news analysis, the ability to accurately classify and understand temporal aspects within data is very important. This section introduces our Temporality Classification Toolkit, an integrated suite of tools and methodologies designed to analyse, interpret, and classify financial news sentences based on their temporality. The toolkit is a combination of advanced machine learning models and manual data extraction techniques, all working in synergy to detect and understand temporal information from textual data. The following subsections detail the development, implementation, and evaluation of this toolkit, showcasing its capabilities in handling the complexities of temporal data in various contexts.

4.1. Training of a Temporal Classification Model

One of the foundations of our Temporality Classification Toolkit lies in the development and training of a robust deep-learning temporal classification model. This section delves into the process of training this specialized model, designed to discern and classify temporal references in sentences.

4.1.1. Introduction to Temporal Classification Model Training

In natural language processing, temporal classification plays a key role in understanding and interpreting temporal textual data. For this purpose, we chose the BERT (Bidirectional Encoder Representations from Transformers) model, which is specifically designed for sequence classification tasks. BERT's ability to process text in a bidirectional way, understanding the context of each word in relation to the whole sentence, makes it particularly suitable for handling temporal nuances in text. In addition, its ability to be easily retrained and widely used is another reason why we chose this model. Our training process is methodically structured, starting from setting up configurations, through data loading, preprocessing, model management and finally execution of the training pipeline. Each step is carefully designed to ensure an efficient and effective training process, ultimately leading to a robust and reliable temporal classification model.

4.1.2. Config

The config.py file is the essential element that sets the stage for the model training environment. It contains essential settings and designated paths that are integral to the training workflow. The complete code can be found in the Appendix section A10. temporal_analysis/training/config.py

The file specifies the location of the temporal analysis dataset through DATASET_PATH, and it establishes the use of a pre-trained BERT framework by providing paths to the BERT tokeniser and model through TOKENIZER_PATH and MODEL_PATH.

It also specifies directories for storing outputs, logs and the final trained model with OUTPUT_DIR, LOGGING_DIR and FINAL_MODEL_DIR, which are essential for monitoring training progress and archiving the resulting model.

The device configuration is automatically optimised for performance based on the capabilities of the hardware, be it the Mac's MPS, GPU or CPU, as indicated by the IS_COMPUTER_MAC_ARM flag and the DEVICE setting.

In terms of training configuration, the file sets the number of training epochs to 5, which defines the number of times the entire dataset is run through the model. The batch sizes for training and evaluation are set to 16 and 32 respectively to balance computational load and training efficiency. The WARMUP_STEPS parameter is set to 100 to control the gradual increase in the learning rate, which is critical for model convergence. A WEIGHT_DECAY of 0.01 is set to prevent overfitting by applying regularisation.

The logging and evaluation strategies are methodically configured to generate logs and evaluate the model every 50 steps using LOGGING_STRATEGY, LOGGING_STEPS, EVALUATION_STRATEGY and EVAL_STEPS.

In addition, the SAVE_STRATEGY ensures that model checkpoints are saved at regular intervals, while the FP16 setting, set to False, indicates the use of 32-bit precision instead of 16-bit for floating point operations.

Finally, LOAD_BEST_MODEL_AT_END ensures that the best performing model is retained at the end of training.

4.1.3. CustomDataLoader

The custom_dataloader.py file is designed to facilitate the preparation and loading of data, a task critical to any machine learning workflow, especially in the natural language processing domain where text data must be carefully handled and structured to ensure it matches the model requirements. The CustomDataLoader class, derived from the Dataset class in PyTorch's utilities, provides an approach to managing the tokenised text and associated label data. The complete code can be found in the Appendix section A12. temporal_analysis/training/data_processing.py.

On instantiation, the class takes two parameters: encodings and labels. The encodings parameter is expected to be a dictionary where each key corresponds to a different aspect of the tokenised text, such as input IDs, attention masks or token type IDs, which are standard components of input data for models such as BERT. These encodings are pre-processed text data converted into a form that BERT can understand, where each encoding is a sequence of integers representing the text in a tokenised form. The labels parameter contains the ground truth data, the classifications that the model is supposed to predict, encoded as integers for processing efficiency.

The __getitem__ method is implemented to retrieve a single data point from the dataset. When called with an index, idx, it constructs a dictionary consisting of the sliced components of the encodings corresponding to the given index, along with the corresponding label. This dictionary is transformed by PyTorch into tensors, which are the standard format for data in PyTorch models, allowing gradient calculations and backpropagation.

Finally, the __len__ method provides a simple but essential functionality: it returns the size of the dataset defined by the length of the label list. This method is often used by PyTorch's data loaders to determine the number of batches needed for an epoch, thus facilitating the model's training loop.

4.1.4. DataProcessor

The data_processing.py file encapsulates the process of data preparation, an essential step to training a temporal classification model. This file introduces the DataProcessor class, which oversees the transformation of raw data into a structured format suitable for machine learning algorithms. The complete code can be found in the Appendix section A12. temporal_analysis/training/data_processing.py

The class begins its role on initialisation by accepting a file path to the dataset, defaulting to the presumed location of the temporal analysis CSV file. It then proceeds to instantiate several attributes to None, placeholders for the DataFrame (df), label-to-ID mappings (id_to_label and label_to_id), and the training, validation and test DataFrames (train_df, val_df, test_df).

The read_dataset method has the function of ingesting the dataset from the CSV file into a Pandas DataFrame, thus translating the data into a tabular format that can be manipulated and analysed.

After ingesting the data, the convert_labels_to_ids method handles the conversion of textual class labels to numerical IDs, a common requirement for classification tasks where numerical inputs are a necessity for machine learning models such as for the BERT Model. This method not only performs the conversion, but also stores the mappings in class attributes, ensuring a bi-directional mapping between the class names and their corresponding numerical representations.

The method add_ids_to_dataframe enriches the DataFrame with a new column, Labels, containing the numerical class IDs, establishing a direct correspondence between the data samples and their categorical targets.

Data splitting is managed by the split_dataframe method, which splits the dataset into training, validation and test subsets based on specified proportions. This splitting is crucial for evaluating model performance and preventing overfitting, ensuring that the model can generalise to new, unseen data.

The class also provides a utility method, convert_df_to_list, which converts the DataFrame into a list of samples and labels, a format often required for further processing or input into custom data loaders.

Finally, the DataProcessor class includes a static almost_equal method to ensure that the proportions for the data splits add up to 1.0 within a specified tolerance, reflecting a detail-oriented approach to data integrity and consistency.

4.1.5. ModelManager

The model_management.py file introduces the ModelManager class, which embodies the entire lifecycle management of the temporal classification model. This class is designed to streamline the process from model loading to training, evaluation and prediction. The complete code can be found in the Appendix section A13. temporal_analysis/training/model_management.py

At the heart of the ModelManager is its constructor, which initialises the class with label-to-ID mappings, a reflection of the classes in the dataset, and the computing device setting, catering for CPU, GPU or MPS (for Mac). These mappings are essential to BERT's understanding of the output classes during training and prediction as well as efficient training and prediction.

The encode_text method is used for converting raw text into a format that BERT can process, applying the tokeniser with padding and truncation to maintain uniform sequence lengths.

The create_dataloader method encodes text data and wraps it into a CustomDataLoader object that we introduced earlier, facilitating batch delivery of data to the model during training and evaluation sessions.

The load_pretrained_model method is important as it loads the right pre-trained BERT model from the specified paths. This process involves setting the number of labels based on the unique classes in the dataset and linking the internal ID-to-label mappings to ensure that the model outputs match the classes of our specific task.

Training arguments are formulated within the create_training_args method, encapsulating details such as output directories, batch sizes, learning rates and logging intervals. These arguments are fundamental to setting up the trainer - the engine that drives the training process in the Hugging Face Transformers library.

Computing metrics is an important aspect of evaluating the performance of a model. The static method compute_metrics provides a custom metrics computation function that calculates accuracy, precision, recall and F1 score, which are standard metrics for classification tasks.

The train_model method invokes the training process, using the previously set up trainer and training arguments to fine tune the model on the provided data.

Prediction functionality is encapsulated in the predict method, which uses the trained model to infer class labels for new text input. This method demonstrates the practical application of the model by transforming raw text into a tokenised form, performing inference, and then translating numerical predictions back into understandable class labels.

To complete the model's training procedure, the save_final_model method provides the ability to persist the fine-tuned model and tokeniser to a specified directory, ensuring that the trained assets can be reused in our temporality analysis pipeline.

Finally, the load_fine_tuned_model method provides a convenient way to load a trained model from a path and wrap it in a hugging face pipeline to facilitate subsequent predictions, completing the model lifecycle from inception to deployment.

4.1.6. Main

The main.py file is the conductor of the training process, unifying all the previous steps into a cohesive process that takes the model from raw data to trained readiness. It serves as the executable script that will invoke the training of the temporal classification model, integrating the functionality of the DataProcessor and ModelManager classes alongside the configurations set in Config. The complete code can be found in the Appendix section A14. temporal_analysis/training/main.py

The process starts with data processing, where an instance of DataProcessor is created using the dataset path specified in Config. This instance then dutifully reads the dataset, converts the class labels into numerical IDs, adds these IDs to the DataFrame and splits the data into training, validation and test sets. Each of these steps is critical, transforming raw data into a structured format that is ready for machine learning.

Moving on to model management, main.py creates an instance of ModelManager, passing the necessary label mappings and device configuration. The manager then loads the pre-trained BERT model and tokeniser, setting the stage for model training.

In preparation for training, the script extracts text and labels from the data splits and uses the ModelManager to create appropriate dataloaders. These dataloaders are essential for batching the data, a process that optimises memory usage and computational efficiency during model training.

Within the training block, main.py calls the ModelManager to configure the training arguments, drawing on settings such as output directories, batch sizes, warm-up steps and logging configurations - each parameter fine-tuned to the needs of the training session.

The ModelManager then creates a trainer, the component from the Hugging Face library responsible for orchestrating the model training. With the training arguments and data loaders in place, model training begins, using the custom metric calculation to iteratively measure model performance.

After training, the script ensures that the trained model is preserved by calling the ModelManager's save_final_model method, saving the model and tokeniser in the specified directory for future use or deployment.

Although the example prediction code is commented out, it provides an insight into how the trained model could be used to make predictions on new text input, demonstrating the practical application of the model beyond the confines of the training environment.

In essence, main.py serves as a pragmatic entry point that brings to life the various components of the model training ecosystem. It encapsulates the entire training pipeline, from data processing to model fine-tuning and storage, underscoring the harmonious integration of modular code into machine learning project structures.

4.2. Model Testing

4.2.1. Testing Procedure

The testing procedure for the temporal classification model was designed to maximise the use of the available dataset to ensure a robust evaluation of the model's capabilities. In contrast to traditional approaches where a portion of the dataset is dedicated to testing, our strategy did not involve the creation of a separate test set. Instead, the entire dataset, constructed using our Command Line Interface (CLI), was used during the training process. This approach allowed us to use the full range of data for training purposes, while still ensuring a robust evaluation of the model's performance as it is done during the training phase.

During the training, the dataset was divided into two main parts: one for training and one for validation. The training part was used to adjust the weights and biases of the model, while the validation part was used to evaluate the performance of the model during the training phase. This setup helped to identify and correct any overfitting problems, thereby improving the model's ability to generalise to new, unseen data. The decision to use all available data for training and validation was driven by our goal to maximise accuracy, taking into account the importance of using as much data as possible to capture the complex patterns inherent in temporal data.

4.2.2. Testing Results

Upon completion of the training process, the temporal classification model achieved high performance metrics, demonstrating its ability to accurately classify temporal data. The effectiveness of the model is reflected in its high scores on several key evaluation metrics.

The model achieved an accuracy of 99.6%, indicating that it correctly predicted the class labels for almost all of the samples in the evaluation dataset. This level of accuracy is particularly significant in the context of temporal classification, which often deals with complex patterns within the data.

Complementing the accuracy, the model's precision was recorded at 99.64%, illustrating that of the instances classified as positive, almost all were correct. The recall rate was 99.55%, showing that the model successfully identified the vast majority of positive cases in the dataset.

The F1 score, a mean of precision and recall, was a remarkable 99.59%. This score is very relevant because it balances precision and recall, providing a single metric that conveys the overall quality of the model's predictive power, particularly in scenarios where an equal trade-off between false positives and false negatives is desired.

The evaluation loss, which measures the prediction error of the model, was low at 0.0342. A low loss value indicates the model's ability to generalise well from the training data to unseen data, reducing the likelihood of overfitting.

These metrics were obtained efficiently, with the evaluation phase processing approximately 25.42 samples per second. The execution time for the evaluation phase was approximately 9.84 seconds, demonstrating the model's fast performance in practical applications.

In summary, the temporal classification model demonstrated a high degree of accuracy and reliability in its predictions. The low loss value combined with the high F1 score underlines the balanced and robust performance of the model. This strong performance provides a solid

foundation for using the model in real-world applications where accurate and timely predictions are critical.

4.3. Techniques for Manual Extraction of Temporal Data

To perform temporal data analysis, automated deep learning models, while highly efficient, sometimes require the support of manual extraction techniques to fully catch the subtleties of temporal expressions in text. This section delves into the manual methods employed for extracting and interpreting temporal data, methods that are crucial in instances where automated models might not fully capture the contextual nuances of time.

4.3.1. Utilizing SUTime for Temporal Tagging

The main.py script uses SUTime as a sophisticated temporal tagging tool. SUTime is notable for its ability to interpret natural language phrases indicating time, such as "next month" or "two days ago", and convert them into structured temporal data. It works with a high degree of accuracy on a wide variety of expressions, including those that denote ranges or specific points in time. Through configuration, SUTime is set up to mark these ranges and include them in its output, providing detailed insight into the temporal context of events described in the text. The complete code can be found in the Appendix section A15. temporal_analysis/main.py

4.3.2. Date Parsing and Normalization

The manual extraction process includes advanced parsing techniques for a variety of date formats found in text. The script uses regular expressions to identify standard date formats, ISO week dates such as "2023-W48", decade expressions such as "203X", and year-only entries such as "2023". It uses the Python datetime library to convert these different formats into a single datetime object. For example, it translates the decade expressions by assuming a midpoint (e.g. "203X" becomes "2035"), and it interprets ISO week dates to the corresponding calendar date, providing a consistent representation of time across the dataset.

4.3.3. Temporal Distance Calculation and Classification

Temporal distance calculation

The calculate_difference function in the main.py script serves as a sophisticated mechanism for calculating the temporal distance between a given date and the current date. This function deftly handles a variety of date formats, including standard dates, ISO week dates (e.g. '2023-W48'), decade expressions (e.g. '203X') and year-only inputs (e.g. '2023'). It uses regular expressions to interpret these different formats and Python's datetime and dateutil libraries to convert them to a standard datetime format.

For example, the script interprets decade expressions by converting them to a mid-decade year (e.g. '203X' to '2035'), and ISO week dates are converted to the corresponding date of that week. This conversion process ensures that all dates, regardless of their original format, are normalised into a consistent structure for subsequent time distance calculations.

Once the input date has been normalised, the script calculates the difference from the current date, breaking down the temporal distance into years, months and days. This calculation provides a granular view of how far in the past or future an event is relative to the present.

Date Difference Classification Map

The script takes its temporal analysis further by classifying the calculated date differences into different temporal categories. Those categories are coming from our analysis described in the section 2.1.1, and the timeframe chosen are coming from common sense but can easily be changed in the script without the need to retrain any model. This classification is performed by the sut_classify_temporality function, which uses a structured map to categorise dates based on their calculated temporal distance from the present. The classification map includes the following categories:

- **Distant past**: Classified for dates where the calculated years are less than a pre-defined negative threshold (e.g. more than one year in the past).
- **Recent Past**: Classified for dates that fall within the last year to the present, typically defined as dates within the range of -1 to 0 years and -12 to -1 months from the current date.
- **Present**: Categorised for dates within a narrow range of the current date, typically defined as within +/- 31 days.
- **Near Future**: For dates that are predicted to occur in the near future, typically within the next 12 months.
- **Distant Future**: Dates beyond the near future threshold, typically more than a year in the future.

Each date's classification is determined by comparing its calculated difference in years, months and days to these pre-defined thresholds in the map. The function's decision logic systematically evaluates the temporal distance against each category's criteria, assigning the most appropriate classification based on the date's relative position to the present.

4.4. Temporal Analysis Algorithm Implementation

In an effort to develop a comprehensive algorithm for temporal analysis, the algorithm synthesises two methodologies: the rule-based SUTime and the machine learning-driven BERT model. This synthesis aims to exploit the strengths of both approaches to ensure a robust system capable of interpreting a wide range of temporal expressions. The complete code can be found in the Appendix section A15. temporal_analysis/main.py.

4.4.1. Integration of SUTime and BERT for temporal analysis

The script uses SUTime for its ability to identify and normalise explicit temporal phrases. However, recognising that SUTime may not capture all temporal nuances, especially those embedded in complex sentence structures or subtle linguistic cues, the script also uses our BERT model fine-tuned for temporal analysis. This dual approach ensures that both explicit and implicit temporal references are captured and accurately classified.

4.4.2. Sentence processing with SUTime and BERT

Each sentence is passed through SUTime, which searches for date-related information. The output of SUTime is parsed to identify the types of dates mentioned and their respective values. This information is crucial for constructing a timeline of events or understanding the sequence of historical data.

At the same time, the sentence is analysed by the BERT model, which has been trained to interpret the temporality of text data. By adapting the model to focus on temporality, it can infer the implied temporal context (implied by expressions for example) that may not be explicitly stated.

4.4.3. Decision making between SUTime and BERT

After collecting temporal data from both SUTime and the BERT model, the script uses a decision-making process to determine which source takes precedence. If SUTime successfully identifies a date, its classification is prioritised due to its rule-based accuracy for explicit dates. However, in cases where SUTime returns an 'Unknown' classification, indicating the absence of recognisable temporal phrases, the script defers to the inference of the BERT model.

This process leverages SUTime's precision for explicit expressions and BERT's nuanced understanding of language, combining their outputs to produce a more accurate and comprehensive temporal analysis. The decision logic systematically evaluates the strengths of each method, ensuring that the most reliable temporal classification is selected for each sentence. Below, in Figure 17: Temporal decision-making process is a visual representation of the decision-making process.

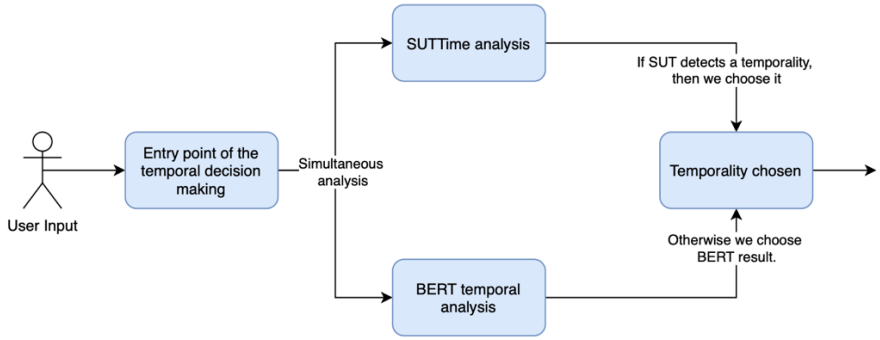

Figure 17: Temporal decision-making process

4.5. An Example of Temporality Analysis in Financial News

To showcase our Temporality Classification Toolkit, we present an analysis of a fake financial news article. Prior to temporal classification, the article was subjected to Named Entity Recognition (NER) and sentence splitting, extracting key financial entities and structuring the content into analysable segments. Below is the article under consideration: "In the beginning of 2023, Tesla experienced a dramatic surge in its stock price following the unveiling of their cutting-edge battery technology, while a few decades ago Tesla's stock was worth nothing. During the previous quarter, Amazon's stock witnessed a substantial rise after a notably successful Prime Day event, which stands in stark contrast to the previous year's Amazon Prime day event that yielded less impressive results, leaving the stock relatively unaffected. Looking ahead to next week, anticipation is building around Apple's announcement of their newest iPhone model, a move that's widely anticipated to bolster their stock value, and make the COE richer, in opposition to MSFT that released the new Surface a few month ago and it didn't have a positive effect on the stock. In contrast, Pepsi may experience a minor setback due to unforeseen delays in rolling out new products. For the end of the year 2025, expectations are high for Microsoft to see considerable gains in its cloud computing revenue, a sentiment echoed for Google, who are also poised for similar growth in this sector in the next 4 months."

The temporal classifications from the analysis are summarized in the following table:

Extracted Text	Temporal Classification
In the beginning of 2023, Tesla experienced a dramatic surge in its stock price following the unveiling of their cutting edge battery technology,	**Expected:** Recent Past **Predicted:** Recent Past
while a few decades ago Tesla's stock was worth nothing.	**Expected:** Distant Past **Predicted:** Distant Past
During the previous quarter, Amazon's stock witnessed a substantial rise after a notably successful Prime Day event,	**Expected:** Recent Past **Predicted:** Recent Past
which stands in stark contrast to the previous year's Amazon Prime day event that yielded less impressive results, leaving the stock relatively unaffected.	**Expected:** Recent Past **Predicted:** Recent Past
Looking ahead to next week, anticipation is building around Apple's announcement of their newest iPhone model, a move that's widely anticipated to bolster their stock value, and make the COE richer,	**Expected:** Present **Predicted:** Present
in opposition to MSFT that released the new Surface a few month ago and it didn't have a positive effect on the stock.	**Expected:** Recent Past **Predicted:** Recent Past

In contrast, Pepsi may experience a minor setback due to unforeseen delays in rolling out new products.	**Expected:** Near Future **Predicted:** Near Future
For the end of the year 2025,expectations are high for Microsoft to see considerable gains in its cloud computing revenue,	**Expected:** Distant Future **Predicted:** Distant Future
a sentiment echoed for Google, who are also poised for similar growth in this sector in the next 4 months.	**Expected:** Near Future **Predicted:** Near Future

Table 4: Temporal Classification Tests

This analysis illustrates the toolkit's capability to accurately classify the temporality of sentences, providing a temporal context that is critical for financial news analysis. Each sentence's classification into categories like "Recent Past," "Distant Past," "Present," "Near Future," and "Distant Future" allows analysts to gain a comprehensive understanding of temporal dynamics and their potential impact on financial markets.

4.6. Conclusion

The Temporality Classification Toolkit marks a crucial step in our financial news analysis toolkit as done. It successfully combines the explicit date-tagging capabilities of SUTime with the nuanced contextual understanding of the BERT model to provide a comprehensive temporal classification solution. This integration ensures a high degree of accuracy and depth in the interpretation of temporal information.

Reflecting the synergy of machine learning and “manual” techniques, the toolkit demonstrates its potential for our application in the financial news analysis sector.

5. Sentimental Classification Toolkit

In financial news analysis, accurately understanding and classifying sentiments within data is crucial. This section introduces our Sentimental Classification Toolkit, an integrated suite of tools and methodologies designed to analyse, interpret, and classify financial news sentences based on their temporality. This section will elaborate on its development, implementation, and evaluation, illustrating its proficiency in managing the complexities of sentiment data in various financial contexts.

5.1. Training of the Sentimental Analysis Model

One of the foundations of our Sentimental Classification Toolkit lies in the development and training of a robust deep-learning sentimental classification model. This section delves into the process of training this specialized model, designed to discern and classify temporal references in sentences.

5.1.1. Introduction to Temporal Classification Model Training

In natural language processing, sentimental classification plays a key role in understanding and interpreting sentimental textual data. For this purpose, we chose the BERT (Bidirectional Encoder Representations from Transformers) model, which is specifically designed for sequence classification tasks. BERT's ability to process text in a bidirectional way, understanding the context of each word in relation to the whole sentence, makes it particularly suitable for handling sentimental nuances in text. In addition, its ability to be easily retrained and widely used is another reason why we chose this model. As we choose the same model as for the temporal classification model (see section 4.1), we mostly re-used the training process and files that we already developed for the previous model.

5.1.2. Major differences between the two-model training

In building both the sentimental and temporal classification models, we used a modular architecture that allowed for significant reuse of code and resources. Central to this architecture were shared components such as custom_dataloader.py, data_processing.py, main.py, model_management.py and model_downloader.py. These components, originally developed for the Temporal classification model, were directly applied to the Sentimental classification model, facilitating a reliable and efficient development process. Reusing these proven components ensured that we maintained consistency in performance and accuracy across both models.

Despite the common foundation, there was a need for customisation, particularly in the config.py file, to meet the unique requirements of each model. Within this configuration file, specific attributes such as DATASET_PATH and FINAL_MODEL_DIR were modified to point to the appropriate resources for the Sentimental classification model. These changes were critical to ensure that the model not only used the correct dataset for training and evaluation, but also stored the final trained model in the designated directory. Apart from these bespoke changes, the rest of the configuration, including paths for the tokeniser, model checkpoints, output directories and training parameters, was kept consistent with the configuration of the Temporal model, highlighting the flexible yet consistent nature of our modular design.

This approach not only streamlined the development process by minimising redundant work, but also ensured that both models benefited from the same robust, well-tested infrastructure, resulting in improved maintainability and scalability of our machine learning pipeline within the Docker-Compose environment. Below, in Figure 18: Code reuse is a visual representation of the code reuse process.

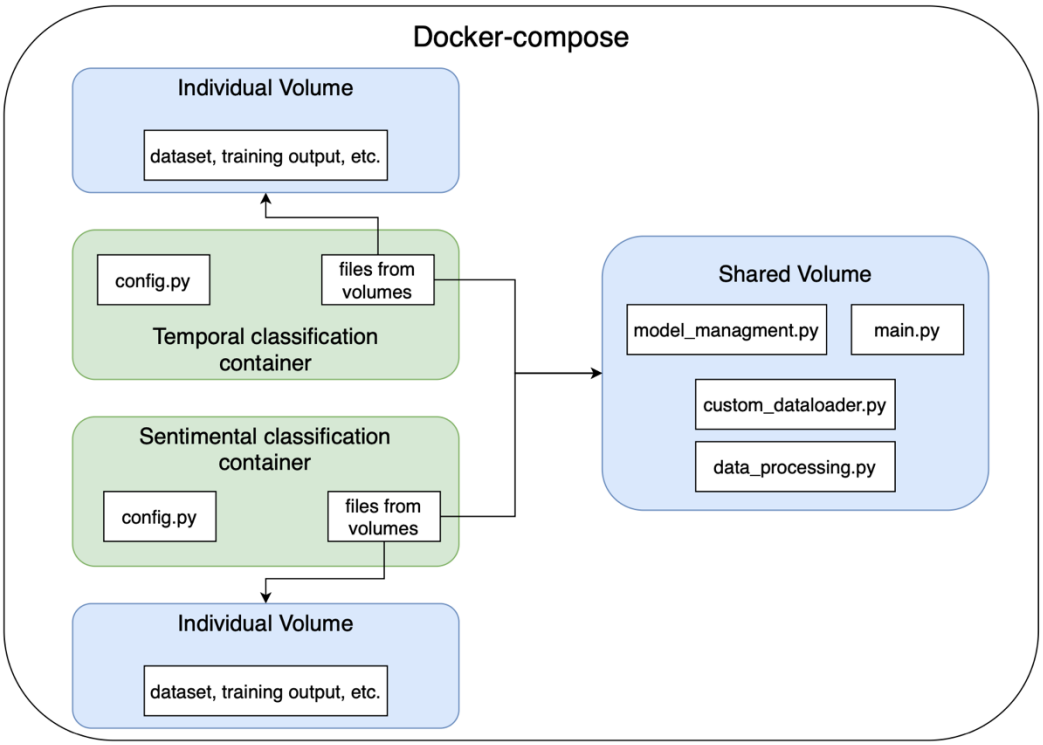

Figure 18: Code reuse

5.2. Model Testing

5.2.1. Testing Procedure

The testing procedure for the sentimental classification model was designed to maximise the use of the available dataset to ensure a robust evaluation of the model's capabilities. In contrast to traditional approaches where a portion of the dataset is dedicated to testing, our strategy did not involve the creation of a separate test set. Instead, the entire dataset, constructed using our Command Line Interface (CLI), was used during the training process. This approach allowed us to use the full range of data for training purposes, while still ensuring a robust evaluation of the model's performance as it is done during the training phase.

During the training, the dataset was divided into two main parts: one for training and one for validation. The training part was used to adjust the weights and biases of the model, while the validation part was used to evaluate the performance of the model during the training phase. This setup helped to identify and correct any overfitting problems, thereby improving the model's ability to generalise to new, unseen data. The decision to use all available data for training and validation was driven by our goal to maximise accuracy, taking into account the importance of using as much data as possible to capture the complex patterns inherent in temporal data.

5.2.2. Testing Results

After the training process, the Sentimental Classification Model showed good performance metrics, underlining its ability to classify temporal data. The model demonstrated its effectiveness with very good scores in several crucial evaluation metrics, as reflected in the updated values.

The model achieved an accuracy of 98.75%, demonstrating its exceptional ability to accurately predict class labels for almost all samples in the training dataset. This high accuracy is particularly significant in the context of temporal classification, which often involves complex patterns within the data.

In terms of precision, the model achieved a remarkable score of 98.75%, suggesting that the vast majority of instances it classified as positive were in fact correct. Complementing this, the recall rate was 98.80%, indicating that the model successfully identified almost all positive cases in the dataset.

The F1 score, a critical metric that balances precision and recall, was a remarkable 98.77%. This score is essential as it provides a comprehensive metric that encapsulates the overall effectiveness of the model's predictive power, especially in scenarios where an equal balance between false positives and false negatives is crucial.

The scoring loss, a metric of the model's prediction error, was significantly low at 0.0526. Such a low loss value indicates the model's strong ability to generalise well from training data to unseen data, suggesting a minimal risk of overfitting.

Efficiency was also a highlight, with the model processing approximately 21.25 samples per second during the evaluation phase. The total time taken for this phase was approximately 18.83 seconds, demonstrating the fast and efficient performance of the model in practical application scenarios.

In summary, these revised metrics for the Temporal Classification Model underscore its high degree of accuracy, precision and reliability in predictions. The combination of a low loss value and a high F1 score highlights the balanced and robust nature of the model's performance, making it a formidable tool in real-world applications where accurate and timely predictions are essential.

5.3. An Example of Sentimental Analysis in Financial News

In this section, we demonstrate the application of our sentiment analysis toolkit to financial news. The aim here is not only to illustrate the capabilities of the toolkit, but also to provide a real-world context where its practical utility is most important. To achieve this, we have created a table showing sentences extracted from various financial news articles, each paired with its corresponding sentiment classification. This table, divided into two columns - one for the sentence and the other for its sentiment classification - serves as a tangible representation of our toolkit's performance in real-world scenarios, going beyond the mere theoretical results produced by our fine-tuning script.

The inclusion of such a table is important for a number of reasons. Firstly, it allows us to observe the effectiveness of the toolkit in identifying sentiments in a specialised field such as finance, where the language used often has unique nuances and jargon. Secondly, this approach allows us to assess the versatility and accuracy of the toolkit in interpreting different linguistic styles and expressions found in different news sources. By presenting these results, we aim to provide a comprehensive overview of the Toolkit's practical effectiveness, offering insights into its real-world applicability in the dynamic and complex field of financial news analysis.

Extracted Text	Temporal Classification
Shareholders rejoice as Apple declares unprecedented annual growth.	**Expected:** Joy **Predicted:** Joy
The consistent performance of Tesla vehicles has fortified investor belief in the company.	**Expected:** Trust **Predicted:** Trust
Investors retreat from France bounds worried about a potential national debt crisis.	**Expected:** Fear **Predicted:** Fear
Unexpectedly, MistralAi pushes the boundaries of artificial intelligence with an industry-transforming algorithm.	**Expected:** Surprise **Predicted:** Surprise
The launch of a massive product recall deeply affected the share prices of BMW.	**Expected:** Sadness **Predicted:** Sadness
General Mills, under fire for falsely advertising genetically modified produce as organic.	**Expected:** Disgust **Predicted:** Disgust
Hidden fees and costs by Interactive Brokers incite a feeling of betrayal among investors.	**Expected:** Anger **Predicted:** Anger

First Solar, Inc is expected to announce a ground-breaking sustainable energy solution, causing a surge in market excitement.	**Expected:** Anticipation **Predicted:** Anticipation
In a shocking move, Impossible Foods announces the successful growth of lab-grown meat, sending its valuation soaring.	**Expected:** Surprise **Predicted:** Surprise
US Bonds constant yields infuse a sense of security amongst investors.	**Expected:** Trust **Predicted:** Trust
Unethical labour practices at Amazon triggers public anger.	**Expected:** Anger **Predicted:** Anger
Caterpillar receives backlash for negligent construction practices.	**Expected:** Disgust **Predicted:** Disgust
Peugeot is enforcing extensive recalls following brake malfunctions in their updated cars.	**Expected:** Fear **Predicted:** Fear
An atmosphere of jubilation wraps the financial markets as Samsung announces their new device: the samsung ring.	**Expected:** Joy **Predicted:** Joy
Market awaits an exciting disruption as Teladoc Health gears up for an industry-changing mental health application reveal.	**Expected:** Anticipation **Predicted:** Anticipation
Due to a major spill at their primary drilling site, Noble Corp PLC anticipates a revenue setback.	**Expected:** Sadness **Predicted:** Sadness

Table 5: Sentimental Classification Tests

The results in Table 5 demonstrate the remarkable accuracy of our sentiment analysis toolkit in the financial news sector, successfully aligning predicted sentiment with expected sentiment across different scenarios. This underscores the toolkit's effectiveness in providing a nuanced understanding of market dynamics, making it an invaluable asset for financial analysis.

5.4. Conclusion

In conclusion, our Sentimental Analysis Toolkit, the toolkit comes out as a very performant and useful tool in our financial news analysis pipeline. Leveraging the BERT model's advanced understanding of context, it provides a nuanced approach to sentimental classification. The toolkit's integration of BERT ensures not only precision, but also comprehensive analysis of temporal references within financial news narratives. This demonstrates the power of machine learning to improve the accuracy and depth of sentiment analysis in financial news and demonstrates its significant potential for applications in this sector.

6. Decision Maker

6.1. Overview

6.1.1. Introduction to the Decision Maker's Role

The Decision Maker module is a key component of our system, with the critical role of synthesising information from our different analytical services into coherent, actionable insights. At its core, this module is designed to integrate outputs from sentence splitting, Named Entity Recognition (NER), sentiment analysis and temporal analysis. The primary objective is to ensure that these separated analysis tools converge effectively, enabling a nuanced understanding of the underlying narrative within the financial news being analysed.

In the domain of data science, natural language processing especially in the field of financial news analysis, such a module is not a luxury, but a necessity. The ability to sift through numerous specialized tools to process unstructured data, identify key entities, understand the sentiment and temporal context, and then integrate these aspects into a single decision-making framework is crucial.

6.1.2. Objectives of the Decision Maker in our System

The primary objectives/tasks of the Decision Maker module can be summarised into several key points:

- **Integration of our analytical services:** The module is designed to seamlessly integrate and chain the outputs of different services. This integration is made to ensure that each analytical component contributes effectively to the overall understanding of the financial news article.
- **Synthesis of sentiment and temporal analysis:** A key function is the synthesis of sentiment and temporal data. This involves understanding how sentiments expressed in the text correlate with different time and how this affects the overall interpretation.
- **Entity-oriented decision making:** Given its integration with the NER service, the module places a strong emphasis on entity-centred analysis. This focus allows for a detailed understanding of how different entities are perceived and discussed in different temporal contexts.
- **Adaptability and scalability**: The module is designed to be adaptable to different types of text data and scalable to handle varying amounts of data, making it a versatile tool in the data analysis arsenal.
- **Generate actionable insights:** The ultimate goal is to generate actionable insights. This involves not only analysing the data, but also interpreting it in a way that is useful to the end user, whether it is for making financial decisions, understanding public sentiment, or any other application.

In the following sections, we will delve deeper into the intricacies of Decision Maker's functionality and explore how each component contributes to these objectives and the overall decision-making process.

6.2. Integration of Services

In this section we will explore the intricate details of how our services are integrated into the Decision Maker module. For a complete understanding, the code and technical specifics of this module can be found here A16. decision_maker/main.py.

6.2.1. Services Integration

The integration of all our analytical services is a base in the construction of our Decision Maker module, each bringing a unique dimension to the financial text analysis process for a more nuanced and comprehensive understanding.

The process begins with the sentence splitting service, which breaks down text into smaller, more manageable segments. This is essential to ensure coherence in the subsequent NER, sentiment and temporal analysis, avoiding confusion in the detection of temporality and sentiment.

After sentence splitting, the NER service identifies and classifies entities such as names of people, organisations or locations. This helps contextualising the sentiment and temporal aspects for later analysis.

A critical step in this integration is the merging of non-entity segments and the updating of the NER indices, a process that ensures continuity and contextual integrity. For each sentence, sections without entity recognition are merged with adjacent sections. This merging is followed by an update of the entity indexes to maintain the accuracy of the entity's location within the text. This step is key to maintaining the flow of the article, ensuring that the absence of entities in certain sections doesn't disrupt the overall analysis.

Sentiment analysis then examines the emotional tone, providing insights into attitudes, opinions and emotions that are critical to understanding sentiment towards identified entities. The implementation involves sending requests to our dedicated sentiment analysis service, where each section of the text is analysed independently. This service, accessible via a specific URL, receives the text and returns a JSON response containing the sentiment analysis results.

Temporal analysis, the final piece, adds a temporal dimension by categorising information into segments such as past, present or future, thereby deepening the overall analysis. The module sends a request to the temporal analysis service, where each section of text (coming as a result of the NER and sentence splitting process) is analysed to determine its temporal context. The response, in JSON format, details the temporal categorisation of the text.

This process of integrating sentence splitting, NER, sentiment and temporal analysis, complemented by the strategic combination of sections of text, forms the backbone of our Decision Maker module.

6.2.2. Flask and CORS Setup

The technical backbone of this integrated framework is the Flask web framework, complemented by the Cross-Origin Resource Sharing (CORS) setup. Flask was chosen for its efficiency and simplicity in setting up a web server capable of handling the complex flow of data between the different services.

In this setup, the Flask application acts as the central node, managing the requests and responses that flow between the user interface and the back-end analytics services. Its adept handling of HTTP requests makes it an ideal choice for this multifaceted system.

Equally important is the implementation of CORS, which is essential for secure and efficient cross-domain communication. This is particularly important given the modular nature of the system, where different services may be hosted on different servers or domains. CORS ensures that there's a seamless and secure exchange of data across these different origins.

6.3. Decision Making Algorithm

6.3.1. Counting Appearances of Sentiments and Temporalities

The first step in the Decision Maker's algorithmic process is to quantify the occurrence of different emotions and temporalities in relation to identified entities. This step is crucial in establishing a basic understanding of how different sentiments and temporalities are associated with each entity within the text.

This process starts with the analysis data obtained from the sentiment and temporal analysis services. Each piece of text, already segmented and analysed for sentiment and temporality, is further examined to count the frequency of each sentiment within specific temporal contexts for each recognised entity. For example, the algorithm determines how often sentiments such as joy, fear or anticipation are associated with an entity such as a company or a person in different temporal frames such as past, present or future.

The implementation of this process involves iterating through the analysis results, examining each block of data. The algorithm extracts temporality, sentiment and entity details from each entry. It then populates a structured dictionary, where keys represent entities and values are nested dictionaries containing the counts of each sentiment under each temporality category. This structured approach not only helps to organise the data efficiently, but also sets the stage for more sophisticated analysis in the next step.

6.3.2. Calculating Overall Temporality and Sentiment for Entities

After counting sentiments and temporalities, the Decision Maker module moves on to calculating the overall temporality and sentiment for each entity. This calculation is more complex than simply counting; it involves interpreting the collected data to derive the dominant sentiment for each temporality and for each entity.

The algorithm sorts the sentiments for each temporality based on their frequency, prioritising the most prevalent sentiments. This sorting is important in determining the dominant emotional tone associated with each entity in different temporal contexts. The module then uses decision logic to interpret these sorted sentiments. For example, if a particular sentiment significantly outweighs others in a given temporality, it is considered the dominant sentiment for that entity and time frame.

The module also takes into account the possibility of combined sentiments, especially when the difference in frequency between the top sentiments is not significant. In such cases, the algorithm checks whether a combination of these sentiments forms a recognised compound sentiment coming from the plutchik's wheel of emotions detailed in the section 2.1.2 (e.g.

combining 'Joy' and 'Trust' to form 'Love'). This nuanced approach ensures that the module captures the complexity and subtlety of emotional expressions in the text.

Through this algorithm, our Decision Maker module synthesises a comprehensive picture of each entity's sentiment and temporal profile. This synthesis is essential for applications where understanding the nuanced emotional and temporal landscape is key to making informed decisions, such as market analysis, public opinion tracking or strategic planning.

6.4. Conclusion

The Decision Maker module, as described in this section, is a central element of our financial news analysis system, exemplifying the fusion of our multiple analytical techniques such as sentiment and temporal analysis. Its ability to integrate these different services into a coherent decision making process underlines the sophistication and efficiency of modern data processing approaches. This module not only demonstrates the technical prowess of our system, but also reflects the broader capabilities and potential applications in areas such as market analysis, social media monitoring and public opinion research.

In summary, the Decision Maker is a robust and versatile tool, adaptable to different financial news and scalable for different data volumes.

7. User Interface

7.1. Overview and Purpose of the UI

The development of the user interface (UI) for this project is primarily driven by the need to present the backend's analytical capabilities in a direct and user-friendly manner. As an AI-focused thesis, the focus has always been on the sophistication and accuracy of temporal and sentiment analysis of financial text. The UI therefore acts as a bridge, making these complex processes accessible to users by presenting analysis results in a visual format that's easy to interpret. It's important to understand that the UI is a proof of concept, designed to be simple and straightforward without the extensive investment in user experience design elements typically seen in more UI-centric projects.

7.2. UI Design and Implementation

7.2.1. Design Philosophy

Given the AI focus of the project, the UI was designed as a minimalist interface, a decision that reflects our prioritisation of backend functionality over extensive UI design. This approach allowed us to focus on the essentials: creating an interface that facilitates easy interaction with the core features of our system. Users can navigate through our interface, enter financial news for analysis, and view the sentiment and timing associated with different financial instruments over different time frames. This simplicity ensures that users can take advantage of the tool's capabilities without a steep learning curve or the need for detailed instructions.

7.2.2. Visual Design

The visual design of the application is characterised by its modern, clean and structured layout. The interface uses a dark theme with a contrasting colour palette to not only provide visual comfort, but also to draw the user's attention to critical elements such as sentiment indicators and timeframes.

The entry point to the analysis, labelled 'SAFE', takes a minimalist approach, avoiding any unnecessary clutter as shown in **Error! Reference source not found.**. The large central text area is designed for clear visibility and easy user-input, with action buttons such as 'TEXT 1', 'TEXT 2' and 'TEXT 3', all highlighted in a subtle teal that stands out against the dark background. Those action buttons allow the user to directly enter some pre-written news articles to test the tool. The 'SUBMIT' button is prominently displayed, inviting action. The overall design adheres to the principles of simplicity and user-centricity, ensuring that the user's path to performing an analysis is straightforward and unambiguous.

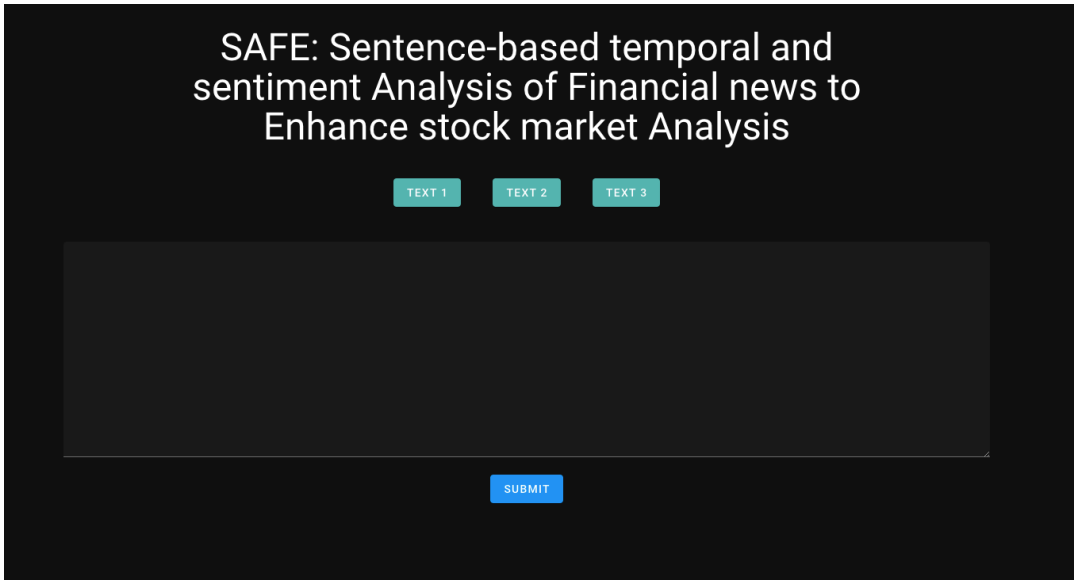

Figure 19: Home Page of the Website

In the 'Your Analysis' section, the presentation of data is neatly segmented into cards, with each card representing a different time category such as 'Recent Past', 'Near Future' and 'Distant Future' as shown in **Error! Reference source not found.**. This segmentation facilitates quick scanning and allows users to intuitively grasp the temporal progression of sentiment associated with stocks. Each sentiment within the cards is encapsulated in a pill-shaped container with a distinct colour, making it easy to identify at a glance, the level of confidence that our system has regarding that emotion.

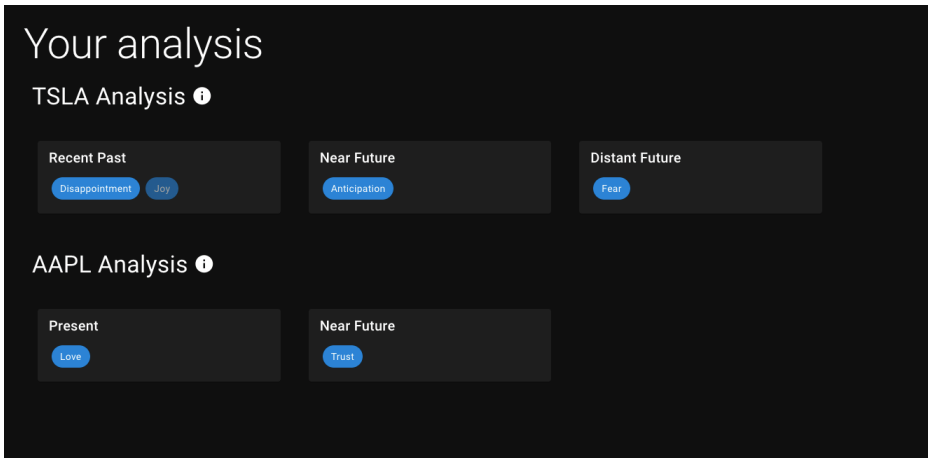

Figure 20: Analysis Page of the Website

7.3. UI Implementation

The implementation of the UI was an effort to bridge design and functionality, ensuring that the interactive visual elements created during the design process were brought to life through code.

7.3.1. Frontend Development Technologies

To develop the front-end, we used Vue.js, an advanced JavaScript framework that is highly customisable and provides an accessible, versatile and powerful foundation for building single page applications (SPAs). The source code shows the use of components, a fundamental feature of Vue.js that promotes reusability and maintainability. Specifically, components such as InputSection.vue (the complete code can be found A17. frontend/InputSection.vue) and AnalysisSection.vue (the complete code can be found A18. frontend/AnalysisSection.vue) modularise the UI by encapsulating the input fields and analysis display logic, respectively.

The Vue Router managed the routing of the application, allowing navigation between different views without refreshing the page, providing a seamless user experience. Pinia, a state management pattern and library, was used to manage the application's state, ensuring that data and UI states were managed centrally and reactively.

7.3.2. Integration with our Backend

Integration with the backend was facilitated by Vue's reactive data binding system, which ensures that the UI components are in sync with the backend data. AJAX calls managed by Vue's ecosystem were used to communicate with the backend APIs, sending user input for processing and retrieving analysis for display. The store directory, particularly store/main.js (the complete code can be found A19. frontend/store/main.js), indicates that Pinia played a critical role in managing the state of the application, presumably by processing data retrieved from the backend and updating the UI accordingly.

The backend integration was designed to be robust and efficient, with error handling and loading states considered to provide feedback to the user during data retrieval or processing. This integration is crucial as it links the user's actions in the front-end with the powerful data processing and analysis capabilities of the back-end, enabling real-time sentiment and temporal analysis of financial text.

8. Conclusion

As we reach the end of this thesis, it is imperative to reflect on the journey we have taken, the milestones we have reached, and the lessons we have learned along the way. This project embarked on an ambitious path to harness the power of machine learning and natural language processing for the nuanced task of sentiment and temporality analysis in the financial domain.

8.1. Summary of achievements

The project has successfully met all of its objectives and represents an advancement in bridging the gap between technology and finance in a public-accessible way.

A notable achievement is the development of a command line interface for dataset creation and extension, which has facilitated the efficient production of rich, domain-specific datasets tailored for temporality and sentiment analysis in our case. This tool fills a critical gap in the field by providing the high-quality data required for accurate analysis, but also a customisable, powerful generation tool that anyone can use to generate their own dataset.

Another contribution is the custom text segmentation solution, which has significantly improved the granularity and accuracy of our analysis. By identifying meaningful segments of text, this module has enabled more precise and contextual analysis, which is critical for dissecting the complex narratives found in financial text.

Another advance is the training and implementation of a temporal classification model, which enables the accurate identification and classification of temporal references within text. This capability enriches the analysis by providing a dynamic understanding of financial events and trends over time.

In addition, the development of a sentiment analysis model has achieved its goal of detecting and categorising emotional and sentimental nuances in financial texts. This tool plays a key role in understanding market sentiment and investor attitudes, providing deeper insights into potential market impacts.

The integration of sentiment and temporal analysis tools into a cohesive decision-making framework represents a leap towards the application of advanced analytical tools in real-world financial contexts. This integrated framework synthesises insights from both analyses to support informed decision making, demonstrating the practical utility of the tools developed.

Although considered a secondary objective, the creation of a user-friendly interface has successfully enhanced the accessibility and usability of the toolkit. This interface allows users to easily interact with the tool's features, input data, initiate analyses and visualise results, thereby extending the applicability and reach of the tool.

8.2. Limitations and perspectives

In its quest to advance the fields of financial analysis through machine learning and natural language processing, this project has made progress. However, as with any research, there are areas of limitation that offer opportunities for future improvement and research.

One of the more nuanced challenges faced by this project is the text segmentation module. Although it significantly improves the granularity and accuracy of our analysis, it is not without its imperfections. For example, in cases where a sentence contains one entity and multiple sentiments/temporalities separated by commas, the module may struggle to accurately associate sentiments and temporalities with the correct entity. This is because while the segmentation technique is effective in identifying distinct blocks of text, it does not always correctly interpret the relationship between these blocks and their associated entities, if not present in the same block. This limitation, although mild, suggests a potential area for improvement in developing a more sophisticated understanding of sentence structure and entity relationships.

The data generation process, while innovative in its use of real-world examples and the capabilities of GPT-4, faces its own set of challenges. The representativeness of the generated data, while substantial, is limited by the number of real-life examples provided and the extrapolation capabilities of GPT-4. To improve the accuracy of the models and their applicability to real-world datasets, future work could involve enriching the CLI tool with a larger set of examples or using fully manually annotated datasets. This would provide a more robust basis for training the sentiment and temporal classification models, ensuring a higher level of accuracy in real-world applications.

Another limitation arises from the Named Entity Recognition (NER) component of the project (developed during a previous project), which currently focuses only on stocks listed on NASDAQ. This scope restriction means that the tool's ability to associate sentiment and temporality with stocks is limited to those within the NASDAQ. Expanding the dataset to include a wider range of stocks, possibly using the CLI tool to generate the dataset, would significantly improve the coverage of the NER and the overall utility of the toolkit.

With regard to the user interface (UI), its simplicity and basic functionality, while sufficient, leave room for improvement. Enhancements such as allowing users to submit links for content retrieval and incorporating a user account system for saving analyses could improve user experience and engagement. These improvements would not only make the toolkit more accessible, but also more versatile, catering to a wider range of user needs and preferences.

Looking to the future, an exciting direction for further work would be the development of an automated system capable of collecting news, analysing it and presenting a real-time global market sentiment for each stock over different time periods. Such a system would represent a advance in financial analysis, providing users with dynamic insights into market sentiment and trends.

Despite these limitations, the achievements of this project should not be underestimated. Each identified limitation also represents an opportunity for future research and development, promising even greater advances in the field of financial analysis through the application of machine learning and natural language processing technologies. These limitations are stepping stones to refine and enhance the toolkit, ensuring its continued relevance and effectiveness in an ever-evolving financial landscape.

9. Bibliography

[1] N. M. R. M. S. P. Deepanway Ghosal, “STaCK: Sentence Ordering with Temporal Commonsense Knowledge,” 01 01 2021. [Online]. Available: https://aclanthology.org/2021.emnlp-main.683.pdf. [Accessed 29 09 2023].

[2] X. Q. J. D. a. P. Z. Daizong Liu, “Adaptive Proposal Generation Network for Temporal Sentence Localization in Videos,” 01 01 2021. [Online]. Available: https://aclanthology.org/2021.emnlp-main.732.pdf. [Accessed 29 09 2023].

[3] R. H. Prafulla Kumar Choubey, “A Sequential Model for Classifying Temporal Relations between Intra-Sentence Events,” 01 01 2021. [Online]. Available: https://aclanthology.org/D17-1190.pdf. [Accessed 29 09 2023].

[4] A. G. I. V. L. T. C. D. T. Kostadin Mishev, “Evaluation of Sentiment Analysis in Finance: From Lexicons to Transformers,” 16 07 2020. [Online]. Available: https://ieeexplore.ieee.org/ielx7/6287639/8948470/09142175.pdf. [Accessed 29 09 2023].

[5] L. S. C.-H. W. Han Chen, “Investment Sentiment in Finance Market,” 01 01 2021. [Online]. Available: https://www.atlantis-press.com/article/125966013.pdf. [Accessed 29 09 2023].

[6] C. H. H. Y. Cong Chen, “Behavioral Framework of Asset Price Bubbles: Theoretical and Empirical Analyses,” 09 12 2022. [Online]. Available: https://www.mdpi.com/2079-8954/10/6/251/pdf?version=1670921538. [Accessed 29 09 2023].

[7] M. U. Rehman, “Investor sentiments and exchange rate volatility,” 1 1 2013. [Online]. Available: https://ir.iba.edu.pk/cgi/viewcontent.cgi?article=1220&context=businessreview. [Accessed 29 09 2023].

[8] G. A. Shawn McCarthy, “Enhancing Financial Market Analysis and Prediction with Emotion Corpora and News Co-Occurrence Network,” 04 04 2023. [Online]. Available: https://www.mdpi.com/1911-8074/16/4/226/pdf?version=1680600504. [Accessed 29 09 2023].

[9] M. H. S. F. Y. M. Gaël Dias, “TempoWordNet for Sentence Time Tagging,” 23 10 2014. [Online]. Available: https://hal.science/hal-01074964/document. [Accessed 02 10 2023].

[10] A. M. C. L. P. D. O.-M. Sulea, “Temporal classification for historical Romanian texts,” 01 01 2013. [Online]. Available: https://aclanthology.org/W13-2714.pdf. [Accessed 02 10 2023].

[11] C. L. a. M. O. a. A. d. Andrade, “DENS: A Dataset for Multi-class Emotion Analysis,” 25 10 2019. [Online]. Available: https://arxiv.org/pdf/1910.11769.pdf. [Accessed 01 10 2023].

[12] H.-C. T. L. Y.-H. H. J. W. Y.-S. C. Elvis Saravia, “CARER: Contextualized Affect Representations for Emotion Recognition,” 01 12 2022. [Online]. Available: https://aclanthology.org/D18-1404.pdf. [Accessed 02 10 2023].

[13] S. S. N. A. S. Y. C. Alisa Liu, “WANLI: Worker and AI Collaboration for Natural Language Inference Dataset Creation,” 16 01 2022. [Online]. Available: https://aclanthology.org/2022.findings-emnlp.508.pdf. [Accessed 06 10 2023].

[14] T. B. H. C. D. M. Shikhar Murty, “A General Task Augmentation Strategy for Few-Shot Natural Language Inference,” 01 06 2021. [Online]. Available: https://aclanthology.org/2021.naacl-main.88.pdf. [Accessed 06 10 2023].

[15] S. S. R. S. Husam Quteineh, “Textual Data Augmentation for Efficient Active Learning on Tiny Datasets,” 01 11 2020. [Online]. Available: https://aclanthology.org/2020.emnlp-main.600.pdf. [Accessed 06 10 2023].

[16] D. &. M. J. H. Jurafsky, “Speech and Language Processing,” 2023. [Online]. Available: https://web.stanford.edu/~jurafsky/slp3/. [Accessed 30 10 2023].

[17] C. R. P. &. S. H. Manning, “Introduction to Information Retrieval,” 2009. [Online]. Available: https://nlp.stanford.edu/IR-book/. [Accessed 30 10 2023].

[18] R. B. A. C. S. K. S. O. M. &. R. C. proat, “Normalization of non-standard words,” 2001. [Online]. Available: https://doi.org/10.1006/csla.2001.0169. [Accessed 30 10 2023].

[19] K. P. M. &. N. S. Lee, “Text Segmentation And Topic Tracking On Broadcast News Via A Hidden Markov Model Approach,” 2003. [Online]. Available: https://www.researchgate.net/publication/2457995_Text_Segmentation_And_Topic_Tracking_On_Broadcast_News_Via_A_Hidden_Markov_Model_Approach. [Accessed 30 10 2023].

[20] T. C. P. L. M. M. A. &. W. L. Baldwin, “How Noisy Social Media Text, How Diffrnt Social Media Sources?,” 2013. [Online]. Available: https://aclanthology.org/bibkey/baldwin-etal-2013-noisy. [Accessed 30 10 2023].