Facultad de Ciencias Económicas y Empresariales,
Universidad Pontificia Comillas (ICADE)

Predicting M&A targets in the U.S. tech industry using ALBERT, FinBERT and Longformer models

Author: Lucía Elegido Ojanguren
Director: María Coronado Vaca

MADRID | June 2024

Abstract

In recent years, the volume of Mergers and Acquisitions ("M&A") has increased significantly, driving heightened interest in predicting these transactions due to their potential profitability for investors, attributed to the premium paid by the acquirer. This study investigates the use of Natural Language Processing ("NLP") Transformers to predict potential M&A targets within the U.S. technology sector by analyzing textual data from companies' 10-K filings, specifically the Management Discussion and Analysis ("MD&A") sections.

Traditional prediction models often rely on financial metrics. However, this research explores the potential of integrating textual information to improve prediction accuracy. Utilizing deep learning models such as ALBERT, FinBERT, and Longformer, the study finds that combining textual and financial data enhances model performance. Among these models, FinBERT, which is trained in financial texts, demonstrates the highest accuracy. Despite these promising results, the study's small and imbalanced dataset limits the generalizability of the findings.

Therefore, future research should focus on expanding the dataset and incorporating additional financial and textual variables to improve the performance of the analysis

Keywords: Natural Language Processing, Mergers & Acquisitions, Transformers and Text analysis

Resumen

En los últimos años, el volumen de Fusiones y Adquisiciones ha aumentado significativamente, generando un interés creciente en predecir estas transacciones debido a su potencial rentabilidad para los inversores, atribuida a la prima pagada por el adquirente. Este estudio investiga el uso de Transformadores de Procesamiento de Lenguaje Natural para predecir posibles objetivos de F&A en el sector tecnológico de EE.UU. mediante el análisis de datos textuales de los informes 10-K de las empresas, específicamente en las secciones de Discusión y Análisis de la Gestión.

Los modelos de predicción tradicionales suelen basarse en métricas financieras; sin embargo, esta investigación explora el potencial de integrar información textual para mejorar la precisión de las predicciones. Utilizando modelos de aprendizaje profundo como ALBERT, FinBERT y Longformer, el estudio encuentra que la combinación de datos textuales y financieros mejora el rendimiento del modelo. Entre estos modelos, FinBERT, entrenado en textos financieros, demuestra la mayor precisión. A pesar de estos resultados prometedores, el pequeño y desequilibrado conjunto de datos del estudio limita la generalización de los hallazgos.

Por lo tanto, la investigación futura debería centrarse en expandir el conjunto de datos e incorporar variables financieras y textuales adicionales para refinar el análisis

Keywords: Procesamiento de Lenguaje Natural, Fusiones y Adquisiciones, Transformadores, Análisis de Texto

Table of Contents

Figures Table of Contents

Acronyms

ALBERT	A Lite Bert
BERT	Bidirectional Encoder Representations from Transformers
EDGAR	Electronic Data Gathering, Analysis and Retrieval system
GRU	Gated Recurrent Unit
GPT-3	Generative Pre-Trained Transformer
LTSM	Long-Short Term Memory
MD&A	Management Discussion and Analysis
M&A	Mergers and Acquisitions
NLP	Natural Language Processing
PTM	Pre-trained Models
RIC	Regulated Investment Company
SMOTE	Synthetic Minority Over-Sampling Technique
R&D	Research and Development
SEC	Securities and Exchange Commission
T5	Text-to-Text Transfer Transformer
ULMFiT	Universal Language Model Fine-Tuning for Text Classification

Chapter 1
Introduction

1. Introduction

1.1. Objectives

The objective of the paper is to analyze companies' reports using Natural Language Processing ("NLP") Transformers to predict possible targets for a Merger and Acquisition ("M&A") buyside. The primary aim of the paper is to test whether the use of textual information rather than financial indicators is statistically meaningful in predicting a merger target/bidder. To delimit the data for the study, the focus of the study will be on the technological sector in United States.

The choice of industry and geography is based on the volume of transactions and the characteristics of the industry. There has been a lot of consolidation over the 25 years in the technology sector as many companies are looking to buy competitors to acquire core competencies, emerging technologies, and specialized talent in order to remain competitive (Institute of Mergers, Acquisitions & Alliances, 2024). In addition, the development of new technologies over the last 25 years has favored this consolidation. In terms of geography, the study will focus on the United States, as this is where the vast majority of technology companies have their headquarters, and it is one of the places with the greatest technological innovation. The concentration of startups, large investments in Research and Development ("R&D") with their respective research centers, frequency of operations and competitiveness make the US a key geography for the study.

1.1.1. Specific objectives

- Description of M&A and main motives behind the transactions and contextualize their relevance and impact in defining the industry landscape and fostering business and economic growth.
- Develop a detailed literature review to identify and understand the quantitative techniques, particularly in the realm of Natural Language Processing, to understand and model human language patterns and structures.
- Analysis of qualitative reports using Transformer Neural Networks

- Assessment and evaluation of the accuracy of the model and its viability
- Explain the main limitations of the approach, conceptually and of the model itself.

1.2. Topic motivation

Since the late 1900s, the volume of Mergers and Acquisitions in terms of the number and value of transactions has increased exponentially, with a peak in 2021 with 58,308 transactions and a total value of c.5,235 billion dollars (Institute of Mergers, Acquisitions & Alliances, 2024). Along with the increase in the number of deals goes the growing interest in predicting M&A operations and the companies that will be involved, because it has a lot of value for investors and other key stakeholders. Identifying targets before transactions take place is a strategy that can be highly profitable for investors due to the premium paid by the acquirer on the intrinsic value of the target share (Katsafados et al., 2021). This implies that foreknowledge of future transactions enables the investor to preempt the market by purchasing said securities, which will subsequently be acquired by a bidder, thereby profiting from the premium to be paid for those securities in the future. Moreover, other research papers have demonstrated that investors could also benefit from prediction because the share prices of firms involved in M&A operations tend to increase right after the announcement (Parungao et al., 2022).

As will be concluded from the literature review presented in section 2 of this paper, although there are studies on the prediction of M&A targets and bidders, the vast majority focus on the analysis of financial variables and key ratios such as: leverage, capital expenditure, or liquidity, which potentially limits the analysis as much of the information reported by the company is non-financial and contained in the management report, annual accounts, letters to shareholders or, increasingly, in sustainability reports. At its core, the alignment between a company's approach to acquiring capability targets and its overarching strategy is paramount, with the strategy of a corporation typically elucidated through textual exposition in most instances. This gap in literature allows us to seek a solution to the same problem from a much more qualitative approach.

Drawing from my experience in the realm of investment banking, I perceive an even greater significance in predicting the parties involved in an M&A transaction, as it offers substantial value to the stakeholders of the companies, investors, regulators, and external advisors — primarily law firms and investment banks.

Moreover, with respect to the technology sector, characterized by its dynamic nature where innovations from small start-ups have the potential to disrupt the market, companies are compelled to remain vigilant and act swiftly and efficiently to avoid lagging behind. Moreover, from an investor's perspective, conducting such analyses for the tech sector is highly prudent, given the substantial premiums associated with acquiring technological assets, reaching multiples of up to 25 times Enterprise Value/EBITDA (Bain & Company, 2022), which can translate into significant gains for those able to identify opportunities.

1.3. Methodology

To conduct this quantitative analysis, we will rely on Python, leveraging its extensive libraries and frameworks known for their robustness and flexibility in data analysis and machine learning tasks. Regarding our data, we will construct our dataset using a comprehensive array of tools, notably Bloomberg and LSEG Refinitiv. In regard to the techniques used, we based our analysis on Natural Language Processing Deep learning models, more precisely on transformers, as this type of models have the ability to learn long-range dependencies and structures between words allowing the model to understand the context. More specifically, we will use ALBERT, FinBERT and Lonformer models to predict whether a company is likely to be a target of an M&A operation or not, based on the Management Discussion and Analysis ("MD&A") of the 10-K filing

However, it is imperative to underscore that this model does not seek to discredit the relevance of models that only incorporate financial variables. The model presents a series of constraints, as non-financial information alone does not gather the entirety of motives driving transactions. Moreover, numerous other variables remain unaccounted for in the study, such as exogenous market variables, macroeconomic indicators like interest rates, technological variables, as well as myriad other factors relating to corporate structures and governance, which ultimately dictate the occurrence or absence of M&A transactions.

1.4. Structure

The development of this research is made up of a second section in which we go over existing literature to further develop on the key elements of the paper. The third section describes the methodology employed, detailing the processes from data collection to the presentation of our model results. The subsequent section is dedicated to discussing and debating our findings, as well as outlining the limitations of our analysis. Finally, the concluding section summarizes the key findings of our research and suggests potential avenues for future investigation.

Chapter 2

Literature Review

2. Literature review

2.1. Definition and existing literature on prediction of Mergers and Acquisitions

Companies to grow, expand or reinforce their market positioning can either rely on organic growth, i.e. with their own operations, or grow inorganically by acquiring transformative capabilities to scale up, using M&A. Mergers and Acquisitions are operations conducted by corporates that involve restructuring the shareholder structure of a company and imply a change of control within companies. These corporate strategies are conducted with the intention of improving firm performance through the obtention of synergies, access to new markets, or acquisition of a range of capabilities that could not be acquired organically and ultimately lead to considerable movements in stock prices.

Predicting this type of transaction allows investors to benefit from timely information. According to existing literature (Bhabra & Hossain, 2017), an investor can achieve substantial returns by purchasing shares of the acquiring company two days before the announcement and selling them two days afterward. This strategy capitalizes on the typical market reaction where stock prices tend to rise following the announcement of a transaction. Having insider information or the ability to accurately predict these transactions is highly valuable as it allows investors to strategically position themselves to benefit from these market movements.

Over the years, many authors have studied the main indicators for predicting this type of transaction, approaching the subject from different perspectives including econometric models, financial ratios (Flannery et at., 2020), qualitative approaches (Delis, et al., 2022) or using artificial intelligence and machine learning for the analysis (Katsafados et al., 2021). From all of them is derived this list of key indicators for assessing whether a company might be the target of an acquisition. Including, being small in size yet growing companies that have not yet reached maturity, being undervalued in the market, having poor management or agency problems, and exhibiting a significant mismatch between resources and growth potential. This statement has important implications for our study, as while company size and market valuation are purely financial variables, factors such as management quality, growth expectations, strategic planning, and how a company intends to finance its expansion plans cannot be adequately represented through ratios alone, thus needing the analysis of textual information.

Therefore, it can be concluded that with M&A predictors classified into two broad categories: financial variables and non-financial variables, the sentiment communicated by managers through corporate strategy, market dynamics, and management may not correspond to current financial results and a common topic when explaining merger failure is the tendency of focusing on financial variables overlooking the human and organizational elements (Calipha et al., 2010). This discrepancy further underscores the importance of considering both types of variables when developing a predictive model.

2.1.1. Artificial Intelligence applied to M&A prediction.

With our focus on non-financial predictors, existing literature shows that the sentiment shown in annual reports has direct effect on the probability of a company becoming a target and, companies with a higher proportion of negative sentiment shown in their non-financial reported information are much more likely to become M&A targets (Katsafados et al., 2021).

Furthermore, there are additional studies that utilize state-of-the-art transformer-based sentiment analysis to enhance the predictive capabilities of traditional statistical models. These studies explore the potential of analyzing the sentiment in company-specific news texts to predict M&A targets (Hajek & Henriques, 2024). They aim to leverage advanced sentiment analysis techniques to provide deeper insights and improve the accuracy of predictions in identifying potential M&A activities.

Additionally, numerous studies have focused on various industries and attempted to utilize machine learning to automate the analysis process when predicting whether a company is a suitable candidate to merge based on text documents from the Securities Exchange Commission, such as the full 10-K filing (Jiang, 2021) or specific sections like the Management Discussion (Routledge et al., 2013). Hence, our study broadens the literature on employing text analytics on market news or annual reports for M&A target predictions in an industry where research is still lacking.

However, all these models have certain limitations, primarily in the realm of data availability. The vast majority of data and reports for private companies are often not public, which limits the generalizability of the models. In addition to this, there is also the challenge of language ambiguity, which can be difficult for models to capture, and the issue of model interpretability.

2.2. Technology industry in the United States; consolidation and growth prospects

In recent years, the US Tech sector has undergone significant consolidation, with the need to acquire new technical and technological skills as the main driving force. More specifically, the high technology industry has been, by far, the industry with the largest number of deals, representing 19.9% of all deals announced between 2000 and 2018, and ranking third in terms of overall value (Institute of Mergers, Acquisitions & Alliances, 2024), with some companies engaging in as many as 30 deals per year (Bain & Company, 2022).

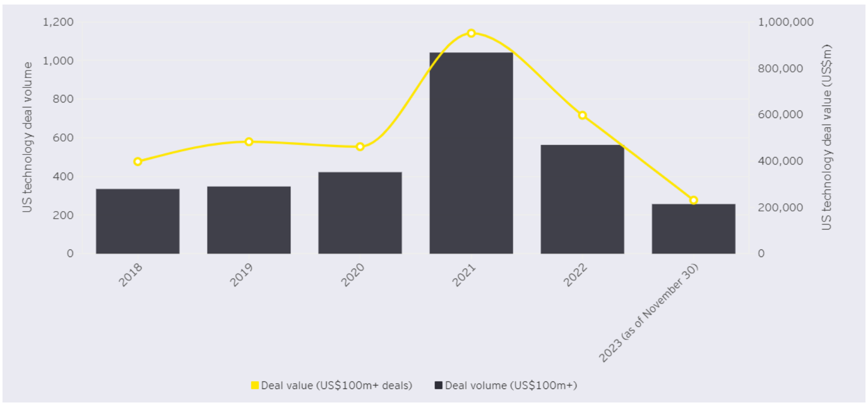


Figure 1. US technology M&A activity evolution. Deals valued at +US$100m

Source: Ernst & Young, 2023

As shown in figure 1, in the aftermath of COVID-19 crisis, most global M&A deals focused on technological innovation, a cross-sectional and transformative element of the contemporary economy, proving to be one of the most resilient sectors to the crisis. However, as shown in figure 1, in recent years the industry has experienced a significant reduction in both the volume and size in terms of value of deals. This trend can be attributed to the conservative approach that companies have adopted regarding M&A strategies, emerging in response to the interest rate hike in March 2022 due to Covid-19 crisis and exacerbated by the Ukraine conflict. Furthermore, the decrease in deal value can also be explained by market dynamics. It is not solely high-value transactions that companies are focusing on, but also thousands of low-profile/lower-value transactions aimed at adding capabilities that enhance performance rather than transformative transactions that seek to disrupt the core business.

Despite these trends, the surge in artificial intelligence is expected to drive a new cycle of growth for the industry, with cloud computing and cybersecurity also identified as key growth drivers (Deloitte, 2023). Consequently, companies may need to acquire these new capabilities into their product portfolio, reactivating the M&A market in the upcoming years. Due to past volumes and expected future consolidation, we argue it would add great value to existing literature to study this industry in depth and to develop models to understand how consolidation functions within it.

2.2.1. Drivers of consolidation

Innovation is the source of competitive advantage for enterprises and a key growth driver. Additionally, buying external capabilities allows for quicker time-to-market rather than relying on internal R&D to improve technology, that is normally associated with high risk, large capital investment and long research and development cycle. Therefore, technology M&A is one of the most effective strategies for enterprises to quickly acquire innovative resources and enhance their technological innovation capabilities to cope with changes in their business models (Suo et al., 2023)

In regards of motivation sources for mergers and acquisitions in this industry, research has demonstrated that the primary motivation for technology mergers and acquisitions is to acquire high-quality and scarce technological resources from the target enterprise, thereby enhancing innovation capabilities. Existing studies indicate that the largest technological companies—Apple, Alphabet, Amazon, Facebook, and Microsoft—all based in the United States, continuously compete in terms of products and services, and rely on these acquisitions to constantly update their ecosystems, restricting competition and consolidating the platform's position in the market (Gautier & Lamesch, 2020).

Existing literature also concludes that the specificity of the business model and the inherent uncertainty associated with companies whose values depend on future outcomes are among the primary drivers of this consolidation (Rossi et al., 2013). In this context, the acquisition of a small and promising startup can represent a significant competitive advantage in the future, consequently, the introduction of new technologies and developments will be associated with a high volume of M&A transactions in this industry.

2.3. Natural Language Processing (NLP)

2.3.1. Introduction to NLP

With the arrival of GPT-3 by OpenAI and similar technologies designed to process and represent language as humans do, Natural Language Processing has gained significant attention. This discipline of artificial intelligence used for understanding and processing human language and with applications in various fields such as customer service, healthcare, and education, traces its origins back to the 1950s. It emerged as a confluence of artificial intelligence and linguistics, seeking to create systems that could understand and interact using human language. Nadkarni et al. (2011) highlight that the earliest efforts during the Cold War included simplistic approaches such as automatic translation programs, developed with the objective of translating Russian sentences into English. These initial steps marked the beginning of a journey towards increasingly sophisticated computational understanding of human language.

Prior to the 1980s, the vast majority of natural language processing systems were based on linguistics and were predominantly symbolic, relying on handcrafted sets of rules (Cambria & White, 2014). Due to these types of models not being economically feasible for everyday applications, owing to the cost and intensive requirements of computer resources (Liberman, 1991), a reorientation occurred during the 1980s. This shift led to the emergence of statistical NLP and the introduction of Hidden Markov Models for speech recognition (Rabiner & Juang, 1989). The following decades were marked by approaches based on supervised machine learning algorithms, based on training labelled texts for automatic classification (Manning & Schutze, 1999), until approximately the year 2010, when deep learning models and transformers were introduced. These models are based on the application of deep neural network algorithms—or Deep Learning—and emerged as a response to the contextual limitations of previous models and to manage the inherent complexity of natural language. Thus, the neural network does not base its output solely on the input, but also on previous and even subsequent inputs, allowing the algorithm to understand the context.

2.3.2. Deep learning transformers

Prior to the emergence of the Transformer architecture by Google creators in 2017, the most pioneering deep learning techniques for handling sequential data relied on convolutional and recurrent neural networks. Despite the capability of the recurrent network Long-Short Term Memory ("LSTM") architecture or Gated Recurrent Unit ("GRU") to use feedback connections to store representations of input events and maintain a type of memory (Hochreiter & Schmidhuber, 1997), these models still presented significant limitations when processing and storing large volumes of text. Therefore, the arrival of the Transformer architecture radically changed NLP technologies and machine translation by allowing models to manage long-range dependencies in the text simultaneously and non-sequentially, with greater computational efficiency.

Transformers are a type of neural network based on attention mechanisms with significant contextual memory (Vaswani et al., 2017), introduced in 2017 as an evolution of previously mentioned sequential models. These models have the capability to relate different inputs over time, producing an output, accordingly, thereby providing a semblance of memory and solving issues with contextualization. Moreover, it is no longer necessary to present tokens in their natural order; instead, all words can be processed in parallel. Currently, most state-of-the-art NLP systems are based on deep Transformer models, generally comprised of several stacked transformer layers (Xiao & Zhu, 2023)

As we will go through in Section 2.3.3., the transformer architecture is based on an encoder that reads an input string and a decoder that prints an output string. The connection between encoder and decoder is made by an attention mechanism, allowing modeling of dependencies without regard to their distance in the input or output sequences (Vaswani et al., 2017).

Among the principal applications of Transformers, we find automatic translation, generation of outputs in chatbots and automatic responses, text and sentiment classification, and, in general terms, text comprehension and the extraction of patterns and structures (Devlin et al., 2018).

2.3.3. Attention mechanisms and Transformer architecture

Prior to the publication of the paper "Attention is All You Need" (Vaswani et al., 2017), other researchers had begun to explore how attention mechanisms applied to encoder-decoder models could enhance performance in automatic translation tasks (Bahdanau et al., 2014). These models implemented an attention mechanism in the decoder, easing the task of the encoder because it no longer had to compress all the information from the source sentence into a fixed-length vector and could decide which parts of the input sentence to pay attention to, laying the groundwork for future studies. It was in 2017 when the Transformer network architecture was introduced, the first transduction model relying entirely on self-attention. To understand how this model properly functions, it is important to distinguish between two types of attention mechanisms: self-attention and encoder-decoder attention.

Multi-head self-attention is named for its use of more than one attention matrix and serves as an attention mechanism that relates different positions of a single sequence to compute a representation of the sequence and to know which other words in the sequence are related to the one being processed. The input matrix, with dimension nx512, is decomposed into submatrices of self-attention that display the relationship between each of the tokens, and for each word, it calculates how much attention should be paid to the rest of the words in the sentence.

On the other hand, encoder-decoder attention is similar to that described in the previous article but will only appear in the sub-layers of the decoder. The decoder layer indicates which output vectors from the encoder, and with assigned weights, should be used to formulate each of the output vectors of the decoder.

Once the mechanisms of self-attention and encoder-decoder attention have been defined, we can examine the basic architecture of a transformer and how the attention mechanisms are integrated with the other elements.

Both the encoder and the decoder consist of six identical layers, and each token input undergoes a process of embedding, which is a vector representation of each token in the first layer of the neural network where the dimensions represent semantic or syntactic aspects of the words (Mikolov et al., 2013), and a process of positional encoding, which describes the location of any token in a sequence so that each position is assigned a unique representation. The result of both transformations is a matrix of Nx512, where N refers to the total number of tokens. The attention units follow this position and embedding tags, calculating a sort of algebraic map of the way each vector relates to the others. The entire operation of the transformers revolves around the algebraic processing of this vector, which is multiplied and added repeatedly as the information progresses from input to output.

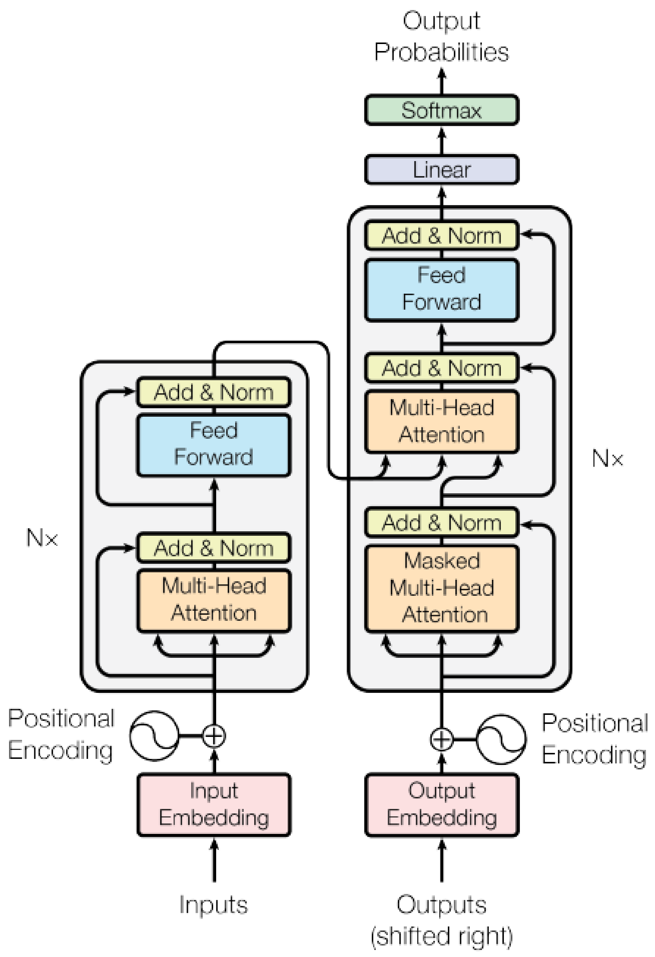


Figure 2. Transformer model architecture. *Source: Vaswani et al., 2017, p.3*

As shown in Figure 2, each layer of the encoder (left hand-side of the figure) consists of two sublayers: a multi-head self-attention mechanism and a feed-forward network, each of which is applied separately and identically to each position. Both elements are followed by a layer normalization operation, ensuring that the resulting vectors maintain a unit dimension. These representations are then fed into the decoder (right hand-side of the figure). The first sublayer of the decoder receives the output and includes a self-attention unit. Subsequently, the second sublayer is an encoder-decoder attention unit that receives queries from the previous decoder sublayer and keys and values from the encoder's output. This configuration allows the decoder to attend to all the words in the input sequence. Finally, similar to the encoder, it includes a feed-forward network to introduce non-linearity and allow the model to learn more complex patterns.

Lastly, the model uses linear transformations and SoftMax function to transform the output of the decoder into predicted next token probabilities.

2.3.4. Overview of pre-trained language models.

The emergence of Transformers has led to the development of numerous language models, all of which are based on the Transformer architecture and offer increased accuracy and versatility for Natural Language Processing tasks. Among the most popular languages, we find; Generative Pre-Trained Transformer (GPT-3), Universal Language Model Fine-Tuning for Text Classification (ULMFiT), Bidirectional Encoder Representations from Transformers (BERT) o Text-to-Text Transfer Transformer (T5), each one focused on a specific NLP task. For instance, GPT-3, specifically designed for natural language generation, is focused on producing coherent text and automatic responses, while BERT is specialized in tasks related to reading comprehension.

Pre-trained transformer models, consisting of dozens of layers and millions of parameters, have achieved in the last years state-of-the-art performance on NLP tasks and have been adopted as key tools for tasks such as question answering, natural language inference or sentiment analysis (Sajjad et al., 2023). The idea behind pre-training relates to transfer learning and applying existing knowledge to new tasks such as token prediction and existing languages are based on two approaches for model-training; fine-tuning and feature-based (Devlin et al., 2018). The fine-tuning strategy involves adjusting the entire pre-trained model to a new task using data specific to that task, in contrast to the feature-based approach, where task-specific architectures of the model are used, and specific features are employed as input for the model dedicated to each task.

Therefore, despite its limitations concerning interpretability, reasoning capability, and robustness, the emergence of Pre-trained Models (PTMs) has facilitated a significant breakthrough in the field of Natural Language Processing. It has been demonstrated that pre-trained models clearly outperform those without pre-training (Wang et al., 2023) and further findings in this realm are anticipated in the coming years.

Chapter 3
Empirical study

3. Empirical study

In this chapter, we will go through the methodology employed to carry out this analysis, which will rely on the Python computer programming language because of its extensive available libraries and commands specifically designed to perform Natural Language Processing tasks, more specifically, Deep Learning Transformers.

Fundamentally, our analysis aims to address a text-based classification task, categorizing companies as either “0”, meaning that are unlikely to be acquired, or "1", more likely to be acquired. This classification will be based on various textual and financial attributes derived from the dataset, enabling us to predict acquisition probabilities with higher accuracy.

3.1. Data

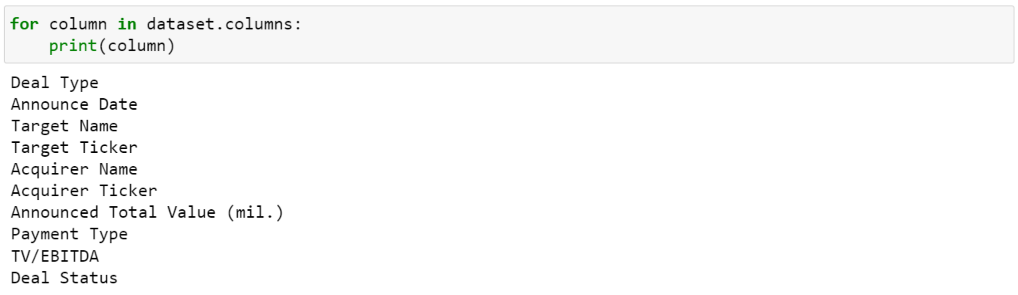
In this first section, the aim is to describe the data acquisition process and the sources, as well as to detail the data preprocessing tasks conducted prior to the implementation of the models.

3.1.1. Data collection

The first step in conducting our analysis involves obtaining the dataset. As previously mentioned, this analysis will focus on predicting M&A targets within the technology sector in the United States. To achieve this, we must first obtain historical data on transactions conducted in this sector over the last ten years. As perDealogic data, since 2014, there have been over 39,126 acquisitions in the U.S. tech sector, with a total volume exceeding $4,460 billion. Therefore, this period is particularly relevant for our study, as it provides a comprehensive overview of recent trends and patterns in tech M&A activities, ensuring our analysis is based on a substantial and pertinent dataset.

Our final dataset will consist of the merger of two subsets: *Targets* and *Non-Targets*. In this section, we will describe the process of obtaining each subset.

To gather the *Targets* data, we will make use of the Bloomberg Terminal, which allows us to download data with specific filters. We exclusively selected transactions categorized purely as "M&A", announced after "01-01-2014", and whose targets are public companies, ensuring access to their annual financial statements. After applying these filters via the Bloomberg Terminal, we exported the data containing the categories shown in Figure 3:



```
for column in dataset.columns:
    print(column)
```

```
Deal Type
Announce Date
Target Name
Target Ticker
Acquirer Name
Acquirer Ticker
Announced Total Value (mil.)
Payment Type
TV/EBITDA
Deal Status
```

Figure 3. Overview of model variables. *Source: Own elaboration*

Once the dataset, hereinafter referred to as *Targets*, is loaded, we apply additional filters using Python. These filters include the following steps: removing all transactions that contain 'N.A.', retaining only those transactions with an 'Announced Total Value (mil.)' greater than 100, and ensuring the 'Deal Status' is marked as Completed. By implementing these filters, we ensure that the dataset is refined to include only relevant and complete data, which is crucial for the accuracy and reliability of our subsequent analysis.

To construct the *non-targets* dataset, we will use the LSEG Refinitiv Data platform API, specifically employing the RDP peer-screening function. Choosing a set of comparable companies involves looking for firms that are similar to the company we are trying to assess, and therefore using peers, contributes positively towards determining whether a company is fit for your comparable universe, ensuring that the analysis and valuation are accurate and reflective of industry standards.

We define the peers' function from the LSEG Refinitiv Data platform in such a way that, for each input entry, which will be the Regulated Investment Company ("RIC"), a similar identifier to the Ticker, and the announcement date of each Target, we will obtain 50 peers for the respective company and year. This ensures that the information remains comparable over time. Upon applying the function, we obtain a dataset with 870 rows, to which we will apply a series of filters.

Since we will focus our analysis on the United States, we will only use RICs ending in “. OQ”. Once filtered, we will use a random command to retain only 125 non-targets, maintaining a ratio of one-to-five targets to non-targets. The rationale for this ratio is based on existing literature. Approximately 13.1% of public firms engage in tech activity, acquiring smaller firms that are younger and more efficient (Jin et al., 2023) and public companies in the top or bottom deciles for growth, have on average a 21% probability to become acquisition targets in any given year (Moeller & Vitkova, 2016). Therefore, it is reasonable to assume a one-to-five ratio.

After merging both datasets, *Targets* and *Non-Targets*, we will proceed to obtain the financial and textual variables necessary to construct our final dataset.

<u>Extract textual variables</u>

This textual variable will be the most relevant for the study. To carry out our analysis, we will use section 7 of the 10-K filings of our companies. The 10-K is a document that publicly traded companies in the United States are required to submit annually to the Securities and Exchange Commission (“SEC”). This document reports both their financial results and provides a highly detailed view of the company regarding ESG matters, key risks, and business outlook. One of the main advantages of this document is its standardized format, which allows us to access specific sections using the Electronic Data Gathering, Analysis, and Retrieval system (“EDGAR”) API. EDGAR is an online database introduced in 1984 to enhance transparency and facilitate the dissemination of information. Over the years, an API has been developed that enables users to query this database and extract data in various formats, such as JSON, XML, and CSV and by leveraging Python libraries such as *requests* or *edgar*, you can programmatically send queries to the API and handle the responses.

For our purposes, we will use the Ticker and the Announcement Date year to perform these queries. From these queries, we will obtain the URL of 10-K filing and store the content in our dataset, under the variable named “MD&A”. More specifically, our analysis will focus on obtaining the text contained in Section 7 of the 10-K, known as the Management Discussion and Analysis. In the MD&A, management provides an overview of the company’s past performance, current financial condition, and future projections and outlook. This section is extremely useful for analysis, as any indication of potential acquisitions or other negative sentiments that could signal high growth, scarce resources, or funding needs would be interpreted by our model as a sign of a possible target.

After iterating through the data requests from the SEC and storing them in our dataset, we will proceed to the final step of data acquisition, which involves extracting the financial variables.

Extract financial variables

Lastly, to obtain the financial variables, we will again rely on the LSEG Refinitiv platform. Specifically, we will use the Excel Add-In, as making requests from Python entails higher computational consumption and fails when tried. We can directly download the data from Excel using a series of functions that will always be based on the RIC. For each record on our dataset, we will obtain the following data points.

- The Price to Earnings ratio, hereinafter P/E, metric commonly used to assess the company's valuation and discuss whether is overvalued or undervalued on the market
- Revenue to determine the relative size of each company
- EBITDA, to construct the EBITDA margin, calculated as EBITDA/Revenue. This measure helps understanding whether the company is profitable relative to its revenue

We will use these three financial variables, assuming that, as previously discussed in the literature review section, acquisition targets are typically smaller companies that achieve high margins and are undervalued in the market, making them more attractive to potential buyers. The attractiveness of these companies is further heightened by their growth potential, which, despite their small size, positions them as prime candidates for acquisition.

Once these financial variables have been downloaded, the data collection process is completed, and we will start preprocessing the data prior to developing our models.

3.1.2. Data pre-processing

Data preprocessing involves the process of cleaning and preparing the data that will be used in the model (Siino et al., 2024). This includes distinguishing between the preprocessing of numerical variables and the preprocessing of textual data that will be processed by the Transformer. The former typically involves operations such as normalization, standardization, and handling missing values, while the latter primarily entails tokenization.

In the data preprocessing stage, we convert the "MD&A" column to a text data type to ensure uniform treatment in subsequent analyses. We also eliminate columns such as "EBITDA" and "Market Cap at Announcement date", which were only needed to derive financial metrics that we will use. This reduces dimensionality and noise, thereby improving model performance (Guyon & Elisseeff, 2003). Subsequently, we standardize the "Revenue" variable, as it is the only non-ratio numeric variable, using the *StandardScaler* from the *sklearn* library. By adjusting its mean to 0 and standard deviation to 1, we align the variable on the same scale, mitigating the effects of range differences and enhancing the convergence of machine learning algorithms. This approach ensures that the data is adequately prepared for use in predictive models, optimizing the efficiency and accuracy of subsequent analysis.

Regarding the text variables, Transformer-based models do not require extensive data preprocessing. These models are designed to handle raw text inputs directly, as the Transformer architecture effectively encodes long-range dependencies in input sequences through self-attention mechanisms (Rahali & Akhloufi, 2023). Utilizing self-attention mechanisms, they are highly effective at processing raw text data, thus reducing the need for traditional preprocessing steps such as stop word removal, lemmatization, or stemming. Therefore, it is not necessary for our models to perform these preprocessing tasks.

3.2. Methodology

To test our initial hypothesis, that textual variables have predictive power in determining whether a company can or cannot be an M&A target, we will train three different models and test them on a test subset to evaluate their performance. ´

All of the models we used for our analysis; ALBERT, FinBERT and Longformer, are based on the Transformers architecture, but each of them specializes in one type of tasks, on which we will elaborate further. And, for each of the models, which are different in nature, we will calculate a series of statistics that will allow us to evaluate the predictive capacity of each model and make them comparable to each other for later discussion.

3.3. Model

In this section, we will provide a concise description of the three pre-trained Transformer models, highlighting their unique characteristics

3.3.1. ALBERT

A Lite BERT ("ALBERT") is a model inspired by BERT that utilizes factorized embedding parameterization and cross-layer parameter sharing to reduce the number of parameters, thus enhancing speed and lowering memory usage and consumption. Nevertheless, it achieves performance comparable to other models (Casola et al., 2020). The technique of "factorized embedding parameterization" splits the embedding matrix, shared across all model layers, into two smaller matrices. This separation allows the dimension of the input word representation to be distinct from the final hidden layer representation dimension. This approach not only reduces the number of parameters but also decreases the correlation between word embeddings and the hidden layer, thereby aiding in the model's generalization capability (Lan et al., 2020).

One of the primary reasons for our decision to employ this model for the task at hand is that ALBERT has proven particularly effective for small datasets due to its architectural innovations that reduce model size without compromising performance. It has been shown to be highly suitable for scenarios where data is scarce, as is the case with our project. Due to limitations in obtaining data from the SEC, our dataset is quite limited, making ALBERT an excellent choice.

For training this model, we will rely on the libraries *PyTorch* and *Hugging Face Transformers*. As previously mentioned, since it is not necessary to preprocess the text before modeling, we utilize the ALBERT tokenizer to tokenize both training and test data, specifically the textual variable MD&A.

Once we have completed these preparations, we proceed to load the ALBERT model, which is tailored to perform classification tasks. In our case, it is configured to classify the 'Target' variable. The next step involves defining the model parameters within the training arguments. Here, we specify various settings, including the number of epochs, the value of the regularization term to prevent overfitting, and the batch size used in both the training and evaluation of the model.

After the parameters are set, we create the trainer, conduct the model training, and then generate predictions along with results. These outcomes will be discussed in detail in Chapter 4. This structured approach ensures that the model is fine-tuned to our specific requirements, maximizing its efficacy in achieving accurate and robust classification results.

3.3.2. FinBERT

FinBERT is a pre-trained language model designed specifically for financial sentiment analysis, built by further training the BERT language model on financial corpora. Since the BERT model is initially pre-trained with general texts, it might not perform optimally on financial texts, which are often tailored for professional investors.

By pre-training FinBERT with financial-specific texts, it achieves more accurate results within the financial realm. This model focuses on polarity analysis, classifying texts as positive, negative, or neutral. More specifically, FinBERT is applied to text classification where sentiment classification is conducted by adding a dense layer after the last hidden state of the token (Araci, 2019). This approach enhances the model's ability to interpret the nuanced language of financial discourse effectively.

For this model, we will initially deploy a tailored FinBERT model to analyze the sentiment of the textual variable “MD&A”. Subsequently, we will train a Random Forest, incorporating numerical variables into the analysis. Once we obtain outputs from this model, which will categorize sentiments as Neutral (0), Negative (1), and Positive (2), we will store these results in a new variable called "Average Sentiment." This variable will then be used as an independent variable in training the Random Forest, which will rely on the *sklearn.ensemble* library. Random Forest is a supervised ensemble learning method extensively utilized for classification tasks. It constructs multiple decision trees during training and delivers classification based on the mode of the classes predicted by the individual trees, correcting for decision trees' habit of overfitting to their training set" (Louppe, 2014).

This approach leverages the strengths of both FinBERT and Random Forest by combining the nuanced understanding of financial language provided by FinBERT with the robust classification capabilities of Random Forest.

3.3.3. Longformer

This model was introduced as a solution to the shortcomings of traditional transformer-based models in processing long text sequences. Designed to efficiently manage extended sequences, this model incorporates a mechanism of attention specifically tailored for lengthy texts. In our particular use case, Item 7 of the SEC's 10-K filings, which can contain up to 3,040 tokens, the employment of a capable model is crucial. Therefore, we will utilize the model from Allen Institute's AI2, capable of processing sequences of up to 4092 tokens.

The efficiency of this model in handling long sequences is achieved through a combination of local windowed attention and task-specific global attention. Unlike traditional full self-attention, the proposed attention pattern scales linearly with the input sequence, making it exceptionally suitable for applications involving longer documents (Beltagy et al., 2020). For this application, minimal preprocessing is conducted on the independent variable to properly structure and delineate the input data. This ensures that the model maximizes its ability to process relevant information within the confines of the attention window, thereby optimizing performance for extensive textual analyses.

3.4. Results

To compare the results obtained from each of the models, we will focus on analyzing the confusion matrices produced as a result of the classification tasks conducted by each model. A confusion matrix is essentially a summary of the prediction results on a classification problem to check the performance of a classification, where each element of the matrix represents the count of predictions made by the model for each actual class compared to the predicted class (IBM, 2024).

This table allows us to visualize the performance of the classification model by showing the true positives, false positives, true negatives, and false negatives. From it, other metrics for measuring model performance are derived, such as accuracy and precision. However, it must be noted that in cases where the dataset is imbalanced, as it is in our case, accuracy can be misleading. This is because a model that predicts the majority class for all instances will have high accuracy but will fail to correctly identify instances of the minority class. These limitations will be addressed in Chapter 4.

For ALBERT, the confusion matrix resulting is presented in Figure 4

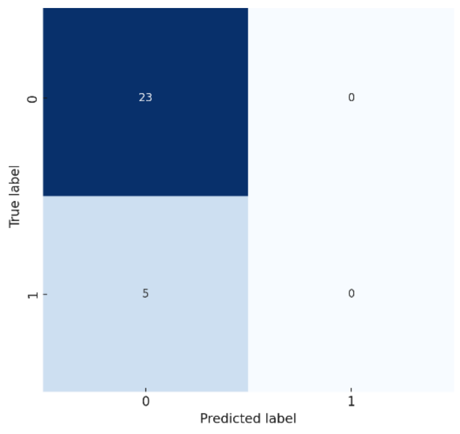


Figure 4. ALBERT Model Confusion Matrix. *Source: Own elaboration*

For FinBERT, the confusion matrix resulting is presented in Figure 5

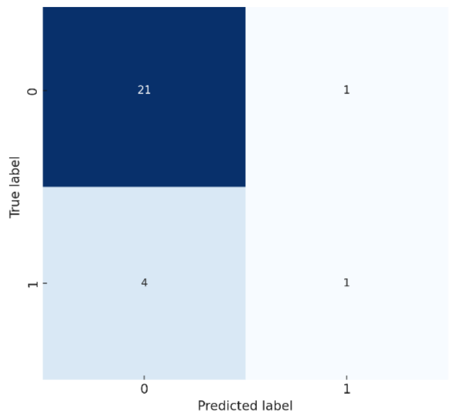


Figure 5. FinBERT Model Confusion Matrix. *Source: Own elaboration*

For Longformer, the confusion matrix resulting is presented in Figure 6

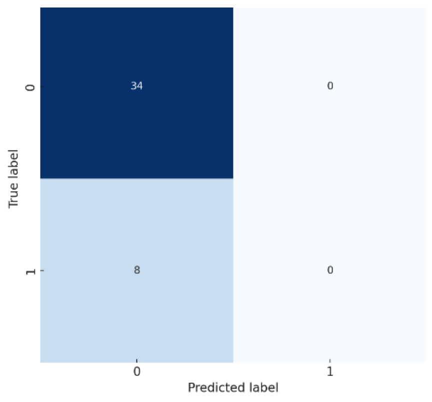


Figure 6. Longformer Model Confusion Matrix. *Source: Own elaboration*

Moreover, to effectively discuss our results, it is imperative to extract a series of statistical measures from these confusion matrices. These statistics will enable us, in the subsequent section, to thoroughly analyze and compare the outcomes derived from our models and we will mainly rely on accuracy and precision

As shown in Figure 7, on the one hand we have accuracy that measures the proportion of correct predictions out of the total number of predictions made, and therefore will be useful when evaluating our models. On the other hand, we calculated precision, that aims to measure how reliable a model's prediction is in determining whether a particular point belongs to the predicted class

Model	***ALBERT***	***FinBERT***	***Lonformer***
Accuracy	0.821	0.815	0.809
Precision	0.675	0.5	n.a.

Figure 7. Summary of key statistics by model. *Source: Own elaboration*

Chapter 4
Discussion

4. Discussion of the results

In this section, we aim to discuss and compare the results of our analysis, focusing on the strengths and limitations of each pre-trained Transformer model employed in our study. Moreover, we will address the general limitations of our research methodology and potential implications of our findings.

Regarding the results obtained, despite the relatively high accuracy of our models, as shown in Figure 7, where the accuracy is above 80% in all cases, the sample size is not large enough to conclude the generalizability of the model. As illustrated in the confusion matrices (refer to Figures 4, 5, and 6), each model included in the test set at least one datapoint that belongs to the Target class. However, only one model, specifically FinBERT, was able to correctly predict that a datapoint would be a Target. The other models achieved approximately 80% accuracy but only correctly predicted the non-Target instances, which is not the primary focus of our study. This issue is primarily due to the fact that our dataset is not balanced, meaning that the distribution of classes within the dataset is uneven and one class or a few classes have significantly more instances than others. To address the mentioned issue, aside from increasing the dataset size, particularly focusing on obtaining more samples of the minority class, we could also apply techniques such as oversampling and undersampling or adjusting class weights. Oversampling and undersampling consist of replicating minority class samples or reducing majority class samples, relying on techniques like Synthetic Minority Over-Sampling Technique ("SMOTE"). Additionally, adjusting class weights in the loss function can penalize errors on the minority class more heavily, thereby ensuring a more balanced model performance.

Regarding the comparison of the results of the models used, it is important to highlight several points. Firstly, the ALBERT model offers the advantages of lower memory consumption and increased training speed for BERT, resulting in improved efficiency. This is reflected in its performance metrics, achieving an accuracy of 0.821 and a precision of 0.675 as shown in the confusion matrix in Figure 4. However, its accuracy in analyzing financial data is somewhat limited due to its further training on the English

Wikipedia and Book CORPUS datasets, which are not specifically tailored to the financial domain.

Secondly, the FinBERT model excels in capturing language knowledge and semantic information within the financial domain due to its training with financial corpora. This is evidenced by its strong performance metrics, with an accuracy of 0.815. This model not only performs well in identifying discussions related to Environment, Social, and Governance (ESG) issues, which significantly impact company valuations (Ernst & Young, 2021), but also allows for the combination of text and numerical variables, facilitating a more comprehensive analysis of available data. Despite its strengths, FinBERT faces the challenge of processing input texts longer than 512 tokens. To address this, we fine-tuned model is used to return logit values for each chunk, and the average sentiment is computed across all chunks. This process necessitates dividing the text in the "MD&A" variable into fragments of up to 512 tokens.

Lastly, the Longformer model extends the maximum input sequence length from 512 to 4,096 tokens, thereby enhancing its ability to model long-term dependencies in lengthy texts. This capability is crucial for analyzing extensive financial documents, as demonstrated in Figure 7, maintaining a competitive accuracy of 0.809. However, the increased complexity of the Longformer model makes it more demanding in terms of computational resources to train and execute.

Taking all these factors into account, we conclude that FinBERT provides the best results for our classification task. The superior performance of this model can be partly attributed to the incorporation of financial variables alongside the sentiment from MD&A. Furthermore, it demonstrates that the sentiment presented in the Management Discussion and Analysis provides valuable information when running a classification using the Random Forest model.

In any case, it is imperative to discuss the main limitations of our models and methodology. Firstly, many algorithms tend to learn better from situations that have more weight in the dataset, which means that the model sometimes fails to correctly predict the minority class. In our case, we have an imbalanced dataset where the proportion of one class is considerably smaller. This presents a significant issue, as our primary interest lies in the model's ability to accurately predict the infrequent events in our dataset and should be addressed by either increasing minority class samples, applying oversampling and undersampling techniques to increase the data points of the underrepresented category, or adjusting class weights to penalize errors on the minority class more heavily.

Another major limitation of the model lies in the sample size chosen. The main constraint in expanding the dataset is the acquisition of data from the SEC website via an API, as the number of data points in the dataset is tied to a limited number of requests. To achieve better results and develop a more robust model with greater generalization capability, it would be necessary to work with a larger dataset. Additionally, due to existing regulation, that only requires public companies to disclose their financial statements, our dataset is composed solely of public companies as there is no disclosure obligation for private companies. This implies that the sample is not fully representative of the entire market landscape. The absence of data from private companies could lead to biased model predictions, as it excludes a significant portion of the corporate sector that may behave differently from public companies. This limitation underscores the need for regulatory changes or alternative data acquisition strategies to obtain a more comprehensive dataset that includes private entities, thereby enhancing the model's accuracy and reliability.

Chapter 5
Conclusions

5. Conclusion

Although the dataset was not sufficiently large to assert the robustness of the model's predictive power, the evaluation of the model highlights the importance of NLP-based techniques, as they can enhance the model's predictive power. Therefore, it can be concluded that the analysis of the MD&A sections within the 10-K filings improves the accuracy of our model, and Transformer-based models prove to be useful for this type of task.

The main finding of our analysis is that, after evaluating the results obtained from the three models used, we conclude that despite the significance of textual variables for the model and the predictive power of Transformers, the best performing model is the one that combines both financial and textual regressors.

Among the limitations of the model, we highlight three main points: our analysis is theoretical in nature and lacks generalizability due to the small size of our sample. Constructing a larger dataset, composed of both public and private entities, would help mitigate these limitations and improve the results obtained from our analysis. Additionally, our model has greater predictive power in forecasting if a firm will not be involved in an M&A transaction, rather than the focus of our research, which was to predict if a firm will be involved in a transaction as a target. To address this limitation, a larger and more balanced sample would likely contribute positively to the model's performance for the intended task.

Lastly, regarding the main implications of our findings, we highlight the repercussions of the analysis for financial investors, companies, and regulators. The integration of NLP-based techniques, particularly Transformer models like FinBERT, provides a more accurate tool for predicting a company's involvement in mergers and acquisitions transactions. This facilitates better decision-making and more effective risk management by anticipating significant market movements.

Furthermore, using this same approach, improved models can be constructed to perform more detailed assessments of companies' financial health and prospects, relying not only on numerical data but also on text analysis of key documents such as the MD&A sections of 10-K reports. For investors, such analysis will translate into higher returns; for companies, it will enhance their internal analysis and forecasting capabilities; and for regulators, it will offer greater transparency and insight for decision-making.

5.1. Future lines of research

As a continuation of our analysis, it would be interesting to conduct a similar study using data from the same sector and period but including additional variables to determine which regressors better predict the possibility of being a target or not.

On one hand, we could conduct an analysis with other sections of the 10-K filings or other documents, such as company press releases and 8-K filings, and on the other hand, rely on other financial metrics that may have greater predictive power, such as the EV/EBITDA ratio, revenue growth, or any metric that assesses the company's leverage.

Additionally, future research could also aim to explore the integration of sentiment analysis on earnings call transcripts to gauge the management's outlook and confidence, which may serve as an additional predictor for M&A activities. This approach would combine textual sentiment data with traditional financial metrics to enhance the predictive accuracy of the model.

Declaración de Uso de Herramientas de Inteligencia Artificial Generativa en Trabajos Fin de Grado

ADVERTENCIA: Desde la Universidad consideramos que ChatGPT u otras herramientas similares son herramientas muy útiles en la vida académica, aunque su uso queda siempre bajo la responsabilidad del alumno, puesto que las respuestas que proporciona pueden no ser veraces. En este sentido, NO está permitido su uso en la elaboración del Trabajo fin de Grado para generar código porque estas herramientas no son fiables en esa tarea. Aunque el código funcione, no hay garantías de que metodológicamente sea correcto, y es altamente probable que no lo sea.

Por la presente, yo, Lucía Elegido Ojanguren, estudiante de Administración y Dirección de Empresas y Análisis de Negocio de la Universidad Pontificia Comillas al presentar mi Trabajo Fin de Grado titulado " Predicting M&A targets in the U.S. tech industry using ALBERT, FinBERT and Longformer models", declaro que he utilizado la herramienta de Inteligencia Artificial Generativa ChatGPT u otras similares de IAG de código sólo en el contexto de las actividades descritas a continuación.

1. **Brainstorming de ideas de investigación:** Utilizado para idear y esbozar posibles áreas de investigación.
2. **Referencias:** Usado conjuntamente con otras herramientas, como Science, para identificar referencias preliminares que luego he contrastado y validado.
3. **Metodólogo:** Para descubrir métodos aplicables a problemas específicos de investigación.
4. **Interpretador de código:** Para realizar análisis de datos preliminares.
5. **Estudios multidisciplinares:** Para comprender perspectivas de otras comunidades sobre temas de naturaleza multidisciplinar.
6. **Corrector de estilo literario y de lenguaje:** Para mejorar la calidad lingüística y estilística del texto.
7. **Sintetizador y divulgador de libros complicados:** Para resumir y comprender literatura compleja.
8. **Revisor:** Para recibir sugerencias sobre cómo mejorar y perfeccionar el trabajo con diferentes niveles de exigencia.
9. **Traductor:** Para traducir textos de un lenguaje a otro.

Afirmo que toda la información y contenido presentados en este trabajo son producto de mi investigación y esfuerzo individual, excepto donde se ha indicado lo contrario y se han dado los créditos correspondientes (he incluido las referencias adecuadas en el TFG y he explicitado para que se ha usado ChatGPT u otras herramientas similares). Soy consciente de las implicaciones académicas y éticas de presentar un trabajo no original y acepto las consecuencias de cualquier violación a esta declaración.

Fecha: 20 June 2024

Firma: Lucía Elegido Ojanguren

Bibliography

Araci, D. (2019). FinBERT: Financial Sentiment Analysis with Pre-trained Language Models. arXiv. https://arxiv.org/abs/1908.10063

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv. https://arxiv.org/abs/1409.0473

Bain & Company. (2022). M&A is back: 2021 saw the highest M&A deal value in history, exceeding expectations at nearly $6 trillion. https://www.bain.com/about/media-center/press-releases/2022/global-ma-report-2022/

Beltagy, I., Peters, M. E., & Cohan, A. (2020). Longformer: The Long-Document Transformer. arXiv. https://arxiv.org/abs/2004.05150

Bhabra, H. S., & Hossain, A. T. (2017). Impact of SOX on the returns to targets and acquirers in corporate tender offers. The North American Journal of Economics and Finance, 42, 1-19. https://doi.org/10.1016/j.najef.2017.06.001

Calipha, R., Tarba, S., & Brock, D. (2010). Mergers and acquisitions: A review of phases, motives, and success factors. Advances in Mergers & Acquisitions, 9, 1-24. https://doi.org/10.1108/S1479-361X(2010)0000009004

Cambria, E., & White, B. (2014). Jumping NLP curves: A review of Natural Language Processing Research. IEEE Computational Intelligence Magazine, 9(2), 48-57. https://doi.org/10.1109/MCI.2014.2307227

Casola, S., Lauriola, I., & Lavelli, A. (2022). Pre-trained transformers: An empirical comparison. Machine Learning With Applications, 9, 100334. https://doi.org/10.1016/j.mlwa.2022.100334

Dealogic. (2024). Dealogic Database. Dealogic. https://dealogic.com/

Delis, M. D., Machin, S., McGowan, M., & Mihaylov, E. (2022). Management practices and M&A Success. Journal of Banking & Finance, 134, 106355. https://doi.org/10.1016/j.jbankfin.2021.106355

Deloitte. (2023). 2023 Technology, Media, and Telecommunications Predictions. https://www2.deloitte.com/global/en/insights/industry/technology/technology-media-and-telecom-predictions.html

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv. https://arxiv.org/abs/1810.04805

Ernst & Young. (2023). M&A sector of the year: Tech leads M&A activity in 2023. https://www.ey.com/en_us/insights/mergers-acquisitions/m-and-a-activity-report

Ernst & Young. (2021). Why ESG performance is growing in importance for investors. https://www.ey.com/en_us/insights/assurance/why-esg-performance-is-growing-in-importance-for-investors

Flannery, M. J., Chen, L., Christopherson, J., & Radecki, L. (2020). M&A activity and the capital structure of target firms. SSRN Electronic Journal. https://doi.org/10.2139/ssrn.3688545

Gautier, A., & Lamesch, J. (2020). Mergers in the Digital Economy. SSRN Electronic Journal. https://doi.org/10.2139/ssrn.3529012.

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. Journal of Machine Learning Research, 3, 1157-1182. https://dl.acm.org/doi/10.5555/944919.944968

Hajek, P., & Henriques, R. (2024). Predicting M&A targets using news sentiment and topic detection. Technological Forecasting and Social Change, 201, 123270. https://doi.org/10.1016/j.techfore.2024.123270

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735

IBM. (2023). Confusion Matrix. IBM. https://www.ibm.com/topics/confusion-matrix

Institute of Mergers, Acquisitions and Alliances (IMAA) (2024). Homepage. https://imaa-institute.org/

Jiang, T. (2021). Using machine learning to analyze merger activity. Frontiers in Applied Mathematics and Statistics, 7. https://doi.org/10.3389/fams.2021.649501

Jin, G. Z., Leccese, M., & Wagman, L. (2023). M&A and Technological Expansion. Journal of Economics & Management Strategy, 33(2), 338-359. https://doi.org/10.1111/jems.12551

Katsafados, A. G., Fetscherin, M., & Steiner, H. (2021). Using textual analysis to identify merger participants: Evidence from the U.S. Banking Industry. SSRN Electronic Journal. https://doi.org/10.2139/ssrn.3474583

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., & Soricut, R. (2020). ALBERT: A lite BERT for self-supervised learning of language representations. arXiv. https://arxiv.org/abs/1909.11942

Liberman, M. Y. (1991). The trend towards statistical models in Natural Language Processing. In Natural Language and Speech (pp. 1-7). Springer. https://doi.org/10.1007/978-3-642-77189-7_1

Louppe, G. (2014). Understanding random forests: From theory to practice (Doctoral dissertation, University of Liège). arXiv. https://arxiv.org/abs/1407.7502

Manning, C. D., & Schutze, H. (1999). Foundations of statistical natural language processing. MIT Press. https://doi.org/10.7551/mitpress/2077.001.0001

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv. https://arxiv.org/abs/1301.3781

Moeller, S., & Vitkova, V. (2016). What makes a company an attractive M&A target? Bayes Business School. https://www.bayes.city.ac.uk/study/courses/postgraduate/mergers-and-acquisitions

Parungao, M., Hildebrandt, T., & Recker, J. (2022). Exploring qualitative data as predictors for M&A: Empirical analysis of target firms' letters to shareholders. Cogent Business & Management, 9(1). https://doi.org/10.1080/23311975.2022.2084970

Rabiner, L., & Juang, B. (1986). An introduction to hidden Markov models. IEEE ASSP Magazine, 3(1), 4-16. https://doi.org/10.1109/massp.1986.1165342

Rahali, A., & Akhloufi, M. A. (2023). End-to-End Transformer-Based Models in Textual-Based NLP. *AI*, *4*(1), 54-110. https://doi.org/10.3390/ai4010004

Rossi, M., Tarba, S., & Raviv, A. (2013). Mergers and acquisitions in the high-tech industry: a literature review. International Journal of Organizational Analysis, 21(1), 66-82. https://doi.org/10.1108/19348831311322542

Routledge, B. R., Sacchetto, S., & Smith(2013). Predicting merger targets and acquirers from text. Semantic Scholar. https://api.semanticscholar.org/CorpusID:4931312

Sajjad, H., Dahlmeier, D., Ng, H. T., & Schultz, T. (2023). On the effect of dropping layers of pre-trained transformer models. Computer Speech & Language, 77, 101429. https://doi.org/10.1016/j.csl.2022.101429

Siino, M., Tinnirello, I., & La Cascia, M. (2024). Is text preprocessing still worth the time? A comparative survey on the influence of popular preprocessing methods on Transformers and traditional classifiers. Information Systems, 121, 102342. https://doi.org/10.1016/j.is.2023.102342

Suo, L., Yang, K., & Ji, H. (2023). The impact of technological mergers and acquisitions on Enterprise Innovation: A Review. Sustainability, 15(17), 12883. https://doi.org/10.3390/su151712883

Nadkarni, P. M., Ohno-Machado, L., & Chapman, W. W. (2011). Natural language processing: An introduction. Journal of the American Medical Informatics Association, 18(5), 544–551. https://doi.org/10.1136/amiajnl-2011-000464

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. arXiv. https://arxiv.org/abs/1706.03762

Wang, H., Lin, Y., Sun, X., Zhao, X., & Zhou, J. (2023). Pre-Trained Language Models and their applications. Engineering, 25, 51–65. https://doi.org/10.1016/j.eng.2022.04.024

Xiao, T., & Zhu, J. (2023). Introduction to Transformers: an NLP Perspective. arXiv. https://arxiv.org/abs/2311.17633

Appendix

Predicting M&A targets in the U.S. tech industry using ALBERT, FinBERT and Longformer models

In [ ]:

```
!pip install edgar
!pip install openpyxl
!pip install sec-api
!pip install beautifulsoup4
!pip install nltk
!pip install refinitiv.dataplatform
!pip install xgboost
!pip install plotly
!pip install eikon
!pip install pandas scikit-learn torch transformers accelerate
!pip install transformers torch datasets
```

In [ ]:

```
import pandas as pd
import edgar
import requests
from bs4 import BeautifulSoup
import time
import json
import configparser
import datetime
import numpy as np
from numpy import mean
from numpy import std
import os
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
 from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import tree
from sklearn.ensemble import RandomForestClassifier
import xgboost as xgb
from xgboost import XGBClassifier
from sklearn.decomposition import PCA
from transformers import BertTokenizer, BertForSequenceClassification
import torch
from sklearn import metrics
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.metrics import roc_auc_score
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_validate
from sklearn.model_selection import RepeatedStratifiedKFold
from scipy.stats import norm
import warnings
import refinitiv.dataplatform as rdp
from sec_api import QueryApi
from sec_api import ExtractorApi
from transformers import AlbertTokenizer, AlbertForSequenceClassification, Trainer,
TrainingArguments
from sklearn.model_selection import train_test_split
import torch
from transformers import BertTokenizer, BertForSequenceClassification, pipeline
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, classification_report
```

Step 1 - Data collection

Target datasets

In [ ]:

```
#Upload BBG excel
dataset= pd.read_excel (r'C:\Users\Lucia Elegido\Desktop\5º\TFGs\TFG
Analytics\Código\Datos_v01.xlsx')

#Commands to filter our dataset
data= dataset.dropna(subset=['Target Ticker', 'Announced Total Value (mil.)'])
data = data[data['Announced Total Value (mil.)'] >= 100]
data = data[data['Deal Status'] == "Completed"]
data = data[data['Deal Type'] == "M&A"]

#Target dataset exploration
dataset.shape
print(dataset.shape)
data.shape
print(data.shape)

#We have 353 transactions - we randomly select 30 operations
data= data.sample (n=30, random_state=42)
```

Non-targets dataset

In [ ]:

```
#Connect to Refinitiv

import refinitiv.dataplatform as rdp
app_key = "fb14d6e106e84f068981ebb1c8fd03b814366e30"
try:
    session = rdp.open_desktop_session(app_key)
    print("¿Está la sesión abierta?", session.is_open())
except Exception as e:
    print("Error al abrir la sesión:", e)
```

Manually using Refinitiv Workspace we add a new column to our Target data including the RIC - Identifier in BBG

In [ ]:

```
target_rics = ['BIRT.OQ^A15', 'MCFE.OQ^C22', 'COUP.OQ^C23', 'CY.OQ^D20', 'AMCC.OQ^A17',
              'MANT.OQ^I22', 'EPAY.OQ^E22', 'IMS.N^J16', 'DTLK.OQ^A17', 'SYKE.OQ^H21',
              'IXYS.OQ^A18', 'AVID.OQ^K23', 'ESMT.N^A24', 'CSOD.OQ^J21', 'CAVM.OQ^G18',
              'MENT.OQ^C17', 'RVBD.OQ^D15', 'USER.N^A23', 'COTV.N^H18', 'BRCD.OQ^K17',
              'RAX^K16', 'TUBE.O^L16', 'PS.OQ^D21']

target_dates = ['2013-12-05', '2021-11-05', '2022-12-12', '2019-06-03', '2016-11-21',
               '2022-05-16', '2021-12-17', '2016-05-03', '2016-11-07', '2021-06-18',
               '2017-08-28', '2023-08-09', '2023-10-23', '2021-08-05', '2017-11-20',
               '2016-11-14', '2014-12-15', '2022-10-27', '2018-06-19', '2016-11-02',
            '2016-08-26', '2016-11-10', '2020-12-13']
```

In [ ]:

```
def peers(RIC, date):
   '''
   Get peer group for an individual RIC along with required variables for the models

   Dependencies
   ------------
   Python library 'refinitiv.dataplatform' version 1.0.0a8.post1
   Python library 'pandas' version 1.3.3

   Parameters
   -----------
      Input:
         RIC (str): Refinitiv Identification Number (RIC) of a stock
         date (str): Date as of which peer group and variables are requested - in yyyy-mm-dd
      Output:
         peer_group (DataFrame): Dataframe of 50 peer companies along with requested variables

   '''
   fields = ["TR.F.TotCap","TR.ExchangeCountryISO"]
   instruments = 'SCREEN(U(IN(Peers("{}"))))'.format(RIC)
   peer_group, error = rdp.legacy.get_data(instruments = instruments, fields = fields, parameters =
{'SDate': date})

   return peer_group
```

In [ ]:

```
no_peers = []
no_dates = []
peer_data = pd.DataFrame()

for i in range(len(target_rics)):
   try:
      #request Peer function for each target company in the lits
      vals = peers(target_rics[i], target_dates[i])
      #drop peers with missing values
      vals.dropna(inplace = True)
      #add a column for 30 days prior to the M&A announcement
      vals.insert(loc = 1, column = 'AD-30', value = target_dates[i])
      #append target company's peer data to the main dataframe of all peers
      peer_data = pd.concat([peer_data, vals], ignore_index = True, axis = 0)

   #if error is returned, store ric and request date in a separate list
   except:
      no_peers.append(target_rics[i])
      no_dates.append(target_dates[i])
      continue
```

In [ ]:

```
peer_data.head(10)
peer_data.to_excel(r'C:\Users\Lucia Elegido\Desktop\5º\TFGs\TFG Analytics\Código\Datos_NT.xlsx')
```

In excelwe ammend both datasets so they have the same variables and names in order to properly concatenate

In [ ]:

```
targets= pd.read_excel (r'C:\Users\Lucia Elegido\Desktop\5º\TFGs\TFG Analytics\Código\targets.xlsx',
sheet_name='Sheet1')
nontargets= pd.read_excel (r'C:\Users\Lucia Elegido\Desktop\5º\TFGs\TFG
Analytics\Código\non_targets.xlsx', sheet_name='Sheet1')
merged_dataset = pd.concat([targets, nontargets], ignore_index=True)
print(merged_dataset.head())
```

Download MD&A for each company

#Hay que ir target por target descargando la información que necesitamos y añadiendolo a nuestro dataset para ir almacenando la información relevante

In [ ]:

```
#Example of query for a given ticker and date

from sec_api import QueryApi
queryApi =
QueryApi(api_key="7c812487edf3aa8d88b0cd9bf2df50b4679f56b549076ad133291254cdf95838")
formType:("10-K", "10-KT", "10KSB", "10KT405", "10KSB40", "10-K405")

query = {
  "query": { "query_string": {
      "query": "formType:\"10-K\" AND ticker:BIRT AND filedAt: [2014-01-01 TO 2014-12-31]" ,
  }},
  "from": "0",
  "size": "1"
}

response = queryApi.get_filings(query)
from sec_api import ExtractorApi
ExtractorApi =
ExtractorApi(api_key="7c812487edf3aa8d88b0cd9bf2df50b4679f56b549076ad133291254cdf95838")
filing_url = str(response['filings'][0]['linkToTxt'])
section_html = ExtractorApi.get_section(filing_url,"7")
print(section_html) #Copy this section to our dataset
```

Extract financial variables

In [ ]:

```
dataset['Revenue'] = None
dataset['EBITDA'] = None
dataset['Enterprise Value'] = None
```

In [ ]:

```
import refinitiv.dataplatform as rdp
app_key = "fb14d6e106e84f068981ebb1c8fd03b814366e30"
try:
   session = rdp.open_desktop_session(app_key)
   print("¿Está la sesión abierta?", session.is_open())
except Exception as e:
   print("Error al abrir la sesión:", e)

base_url = 'https://api.refinitiv.com/data'
```

As get_data_from_refinitiv is not working - we obtain the variables from Refinitiv's excel add-in

Error - obtain directly from Python the financial variables

```
def get_data_from_refinitiv(ric, year): headers = { 'Content-Type': 'application/json', 'Authorization':
f'Bearer {app_key}' }

    # Endpoints para cada métrica

    endpoints = {

        'Revenue': f'{base_url}/content/data/company/{ric}/fundamentals/income-
    statement?periodType=FY&fiscalYear={year}',

        'EBITDA': f'{base_url}/content/data/company/{ric}/fundamentals/income-
    statement?periodType=FY&fiscalYear={year}&fields=EBITDA',

        'EnterpriseValue':
    f'{base_url}/content/data/company/{ric}/valuation?periodType=FY&fiscalYear={year}&fiel
    ds=EnterpriseValue'

    }

    data = {}

    for metric, url in endpoints.items():

        response = requests.get(url, headers=headers)

        if response.status_code == 200:

            result = response.json()

            # Extraer el valor de la métrica de la respuesta JSON

            data[metric] = result.get('data', [{}])[0].get(metric, None)

        else:

            data[metric] = None

    return data

for index, row in dataset.iterrows(): ric = row['RIC'] year = row['Year'] data =
get_data_from_refinitiv(ric, year)

    # Actualizar el dataframe con los nuevos datos

    dataset.at[index, 'Revenue'] = data.get('Revenue')

    dataset.at[index, 'EBITDA'] = data.get('EBITDA')
```

```
        dataset.at[index, 'Enterprise Value'] = data.get('EnterpriseValue')

dataset.to_excel(r'C:\Users\Lucia Elegido\Desktop\5º\TFGs\TFG Analytics\Código\Datos_vF.xlsx', 
index=False)

print("Datos extraídos y guardados correctamente en refinitiv_data.xlsx")
```

Data pre-processing

In [ ]:

```
data= pd.read_excel(r'C:\Users\Lucia Elegido\Desktop\5º\TFGs\TFG Analytics\Código\Datos_vF0.xlsx')
print(data.head())
```

In [ ]:

```
#Data exploration
data['Target'].value_counts()
data.isnull().sum()
data['MD&A'] = data['MD&A'].astype(str)
```

In [ ]:

```
#Data cleaning
columns_drop=['EBITDA','Price closing announcement date','Net income','Market Cap at Announcement 
date','Shares outstanding','Market cap v2']
data.drop(columns=columns_drop, inplace=True)
#Standarization
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
data['Revenue']= scaler.fit_transform(data[['Revenue']])

print(data.head())
```

Step 2 - ALBERT Model

In [ ]:

```
#Split dataset
class_1 = data[data['Target'] == 1]
class_0 = data[data['Target'] == 0]
class_1_train, class_1_test = train_test_split(class_1, test_size=0.2, random_state=42)
class_0_train, class_0_test = train_test_split(class_0, test_size=0.2, random_state=42)
train_data = pd.concat([class_1_train, class_0_train])
test_data = pd.concat([class_1_test, class_0_test])
X_train = train_data['MD&A']
y_train = train_data['Target']
X_test = test_data['MD&A']
y_test = test_data['Target']
print(f"Tamaño del conjunto de entrenamiento: {X_train.shape[0]}")
print(f"Tamaño del conjunto de prueba: {X_test.shape[0]}")
```

In [ ]:

```
tokenizer = AlbertTokenizer.from_pretrained('albert-base-v2')
# Función para tokenizar los textos
def tokenize_data(texts):
    return tokenizer(texts.tolist(), padding=True, truncation=True, return_tensors='pt')

train_encodings = tokenize_data(X_train)
test_encodings = tokenize_data(X_test)

class Dataset(torch.utils.data.Dataset):
    def __init__(self, encodings, labels):
        self.encodings = encodings
        self.labels = labels

    def __getitem__(self, idx):
        item = {key: val[idx] for key, val in self.encodings.items()}
        item['labels'] = torch.tensor(self.labels[idx])
        return item

    def __len__(self):
        return len(self.labels)

train_dataset = Dataset(train_encodings, y_train.tolist())
test_dataset = Dataset(test_encodings, y_test.tolist())

model = AlbertForSequenceClassification.from_pretrained('albert-base-v2', num_labels=2)

training_args = TrainingArguments(
    output_dir='./results',          # Directorio de salida
    num_train_epochs=3,              # Número de épocas de entrenamiento
    per_device_train_batch_size=8,   # Tamaño del lote para entrenamiento
    per_device_eval_batch_size=8,    # Tamaño del lote para evaluación
    warmup_steps=500,                # Pasos de calentamiento
    weight_decay=0.01,               # Decaimiento de peso
    logging_dir='./logs',            # Directorio de logs
    logging_steps=10,                # Pasos de logging
    evaluation_strategy="epoch",     # Estrategia de evaluación: cada época
    save_strategy="epoch",           # Estrategia de guardado: cada época
)

def compute_metrics(pred):
    labels = pred.label_ids
    preds = pred.predictions.argmax(-1)
    precision, recall, f1, _ = precision_recall_fscore_support(labels, preds, average='weighted')
    acc = accuracy_score(labels, preds)
    return {
        'accuracy': acc,
        'precision': precision,
        'recall': recall,
        'f1': f1,
    }

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=test_dataset,
    compute_metrics=compute_metrics
)

# Train
trainer.train()

# Evaluate
results = trainer.evaluate()
print(results)

# Obtain predictions
```

```
predictions = trainer.predict(test_dataset)
preds = predictions.predictions.argmax(-1)
labels = predictions.label_ids

print(f"Tamaño de las predicciones: {len(preds)}")
print(f"Tamaño de las etiquetas reales: {len(labels)}")

# Confussion matrix
cm = confusion_matrix(labels, preds)

disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot(cmap=plt.cm.Blues)
plt.title('Confusion Matrix')
plt.show()

# Include predictions to existing dataset
test_results = X_test.reset_index(drop=True)
test_results = pd.DataFrame(test_results)
test_results['Actual'] = y_test.reset_index(drop=True)
test_results['Predicted'] = preds

test_results.to_excel(r'C:\Users\Lucia Elegido\Desktop\5º\TFGs\TFG
Analytics\Código\Datos_Predicted.xlsx', index=False)
```

Step 3 - Finbert Model

In [ ]:

```
# Dividir el dataset en conjunto de entrenamiento y prueba
X = data[['Sentiment']]
y = data['Target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

# Entrenar un modelo de clasificación
clf = RandomForestClassifier(random_state=42)
clf.fit(X_train, y_train)

# Hacer predicciones en el conjunto de prueba
y_pred = clf.predict(X_test)

# Evaluar el modelo
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)

print(f"Accuracy: {accuracy}")
print(f"Classification Report:\n{report}")

# Guardar los resultados en un nuevo archivo Excel
output_file_path = '/mnt/data/data_with_sentiment_and_predictions.xlsx'
data['Predicted'] = clf.predict(data[['Sentiment']])
data.to_excel(output_file_path, index=False)

print(f"Predictions saved to {output_file_path}")
```

In [ ]:

```
# Divide MD&A in fragments up to 512 tokens each
def chunk_text(text, chunk_size=512):
    tokens = tokenizer.tokenize(text)
    chunks = [' '.join(tokens[i:i + chunk_size]) for i in range(0, len(tokens), chunk_size)]
    return chunks

chunks = data['MD&A'].apply(chunk_text)

max_chunks = max(chunks.apply(len))

for i in range(max_chunks):
    data[f'MD&A_chunk_{i+1}'] = chunks.apply(lambda x: x[i] if i < len(x) else '')

# Guardar los resultados en un nuevo archivo Excel
output_file_path = r'C:\Users\Lucia Elegido\Desktop\5º\TFGs\TFG 
Analytics\Código\Datos_Divided.xlsx'
data.to_excel(output_file_path, index=False)

print(f"Data with chunks saved to {output_file_path}")
```

In [ ]:

```
# Cargar el archivo Excel proporcionado con los fragmentos
data= pd.read_excel(r'C:\Users\Lucia Elegido\Desktop\5º\TFGs\TFG 
Analytics\Código\Datos_Divided.xlsx')

# Cargar el modelo y el tokenizador de FinBERT
finbert = BertForSequenceClassification.from_pretrained('yiyanghkust/finbert-tone', num_labels=3)
tokenizer = BertTokenizer.from_pretrained('yiyanghkust/finbert-tone')

# Verificar si hay una GPU disponible y mover el modelo a la GPU si es posible
#device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
#finbert.to(device)

# Crear el pipeline de análisis de sentimiento
nlp = pipeline("sentiment-analysis", model=finbert, tokenizer=tokenizer, device=0 if 
torch.cuda.is_available() else -1)

# Función para obtener el sentimiento de un texto
def get_sentiment(texts):
    results = nlp(texts, truncation=True, padding=True, max_length=512)
    sentiments = []
    for result in results:
        sentiment = result['label']
        if sentiment == 'positive':
            sentiments.append(2)
        elif sentiment == 'negative':
            sentiments.append(0)
        else:
            sentiments.append(1)
    return sentiments

# Aplicar FinBERT a cada fragmento y calcular el promedio de sentimiento
sentiment_columns = [f'MD&A_chunk_{i}' for i in range(1, 21)]  # Nombres de las columnas de 
fragmentos
sentiment_scores = []

batch_size = 16  # Tamaño del lote para el procesamiento por lotes

for index, row in data.iterrows():
    fragments = [row[col] for col in sentiment_columns if pd.notna(row[col]) and row[col].strip() != '']
    if fragments:
        # Procesar en lotes para mejorar la eficiencia
        batch_sentiments = []
        for i in range(0, len(fragments), batch_size):
            batch = fragments[i:i + batch_size]
            batch_sentiments.extend(get_sentiment(batch))

        if batch_sentiments:
            average_sentiment = sum(batch_sentiments) / len(batch_sentiments)
        else:
            average_sentiment = 1  # Neutral si no hay fragmentos
    else:
        average_sentiment = 1  # Neutral si no hay fragmentos

    sentiment_scores.append(average_sentiment)

# Añadir la nueva columna con el promedio de sentimiento
data['Average_Sentiment'] = sentiment_scores

# Verificar los resultados
print(data[['Average_Sentiment']].head())

# Guardar los resultados en un nuevo archivo Excel
output_file_path = r'C:\Users\Lucia Elegido\Desktop\5º\TFGs\TFG
Analytics\Código\Result_Finbert.xlsx'
data.to_excel(output_file_path, index=False)

print(f"Data with average sentiment saved to {output_file_path}")
```

In [ ]:

```
#Only interested in the sentiment not the actual MD&A
columns_to_keep = ['Target', 'Revenue', 'P/E', 'EBITDA Margin', 'Average_Sentiment']
data_v2 = data[columns_to_keep]
output_file_path = r'C:\Users\Lucia Elegido\Desktop\5º\TFGs\TFG Analytics\Código\Data_Modelo.xlsx'
data_v2.to_excel(output_file_path, index=False)
```

In [ ]:

```
data_v2 = pd.get_dummies(data_v2, drop_first=True)
data_v2 = data_v2.apply(pd.to_numeric, errors='coerce')
output_file_path = r'C:\Users\Lucia Elegido\Desktop\5º\TFGs\TFG
Analytics\Código\Data_Modelo2.xlsx'
data_v2.to_excel(output_file_path, index=False)
data= pd.read_excel(r'C:\Users\Lucia Elegido\Desktop\5º\TFGs\TFG
Analytics\Código\Datos_Modelo2.xlsx')
```

In [ ]:

```
print(data_v2.head())
columns_to_keep = ['Target', 'Revenue', 'P/E', 'EBITDA Margin', 'Average_Sentiment']
data_v2 = data_v2[columns_to_keep]
data_v2 = data_v2.fillna(0)
```

In [ ]:

```
print(data_v2.head())
```

In [ ]:

```
# Random Forest
X = data_v2.drop('Target', axis=1)
```

```
y = data_v2['Target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y)
```

In [ ]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Inicializar el modelo Random Forest
clf = RandomForestClassifier(random_state=42)

# Entrenar el modelo
clf.fit(X_train, y_train)
```

In [ ]:

```
#Evaluate and predict
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

y_pred = clf.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

print(f"Accuracy: {accuracy}")
print(f"Classification Report:\n{report}")
print(f"Confusion Matrix:\n{conf_matrix}")
```

In [ ]:

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

disp = ConfusionMatrixDisplay(confusion_matrix=conf_matrix)
disp.plot(cmap=plt.cm.Blues)
plt.title('Confusion Matrix')
plt.show()
```

Step 4 - Longformer model

Los modelos como Longformer y BigBird están específicamente diseñados para manejar secuencias de texto más largas que los modelos tradicionales como BERT, ALBERT, y FinBERT

In [ ]:

```
data2=pd.read_excel(r'C:\Users\Lucia Elegido\Desktop\5º\TFGs\TFG Analytics\Código\merged.xlsx')

def preprocess_text(text):
    return text.replace('&#146;', "'").replace('&#8217;', "'")

data2['MD&A'] = data2['MD&A'].apply(preprocess_text)
```

In [ ]:

```
train_data, temp_data = train_test_split(data2, test_size=0.3, stratify=data2['Target'], random_state=42)
test_data_0 = temp_data[temp_data['Target'] == 0]
test_data_1 = temp_data[temp_data['Target'] == 1]

if len(test_data_1) == 0:
    train_data, temp_data = train_test_split(data2, test_size=0.3, stratify=data2['Target'], random_state=43)
    test_data_0 = temp_data[temp_data['Target'] == 0]
    test_data_1 = temp_data[temp_data['Target'] == 1]
```

```
test_data = pd.concat([test_data_0, test_data_1])

train_dataset = Dataset.from_pandas(train_data)
test_dataset = Dataset.from_pandas(test_data)
```

In [ ]:

```
tokenizer = LongformerTokenizer.from_pretrained('allenai/longformer-base-4096')

def tokenize_function(examples):
    return tokenizer(examples['MD&A'], padding='max_length', truncation=True, max_length=4096)

train_dataset = train_dataset.map(tokenize_function, batched=True)
test_dataset = test_dataset.map(tokenize_function, batched=True)

# Renombrar la columna 'Target' a 'labels' para que el Trainer la reconozca
train_dataset = train_dataset.rename_column('Target', 'labels')
test_dataset = test_dataset.rename_column('Target', 'labels')
```

In [ ]:

```
model = LongformerForSequenceClassification.from_pretrained('allenai/longformer-base-4096',
num_labels=2)

metric = load_metric('accuracy')

def compute_metrics(p):
    return metric.compute(predictions=np.argmax(p.predictions, axis=1), references=p.label_ids)

training_args = TrainingArguments(
    output_dir='./results',
    evaluation_strategy='epoch',
    learning_rate=2e-5,
    per_device_train_batch_size=2,
    per_device_eval_batch_size=2,
    num_train_epochs=3,
    weight_decay=0.01,
)

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=test_dataset,
    compute_metrics=compute_metrics,
)

trainer.train()

trainer.evaluate()
```

In [ ]:

```
predictions = trainer.predict(test_dataset)
pred_labels = np.argmax(predictions.predictions, axis=1)
true_labels = predictions.label_ids

# Matriz de confusión
cm = confusion_matrix(true_labels, pred_labels, labels=[0, 1])
cmd = ConfusionMatrixDisplay(cm, display_labels=[0, 1])
```

```
cmd.plot()

plt.title('Confusion Matrix')
plt.show()
```

In [ ]:

```
report = classification_report(true_labels, pred_labels, target_names=['Class 0', 'Class 1'])
print("Classification Report:\n", report)

# Curva ROC y AUC
from sklearn.metrics import roc_curve, auc

fpr, tpr, _ = roc_curve(true_labels, predictions.predictions[:, 1])
roc_auc = auc(fpr, tpr)

plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
plt.show()
```