Faculty of Economic and Business Sciences (ICADE)

IDENTIFYING LIKELIHOOD OF MERGER PARTICIPANTS THROUGH NATURAL LANGUAGE PROCESSING IN THE EUROPEAN BANKING SECTOR

Juan Sánchez-Blanco Gómez
Clave: 201702244

MADRID | June 2022

Acknowledgements

To my tutor María and my advisor Eduardo for all of their help and dedication. To my brother Carlos, my parents and my entire family for their continued support. To all the teachers and classmates from whom I have learned so much.

Abstract

The economic climate in which the European banking sector operates is very different to that of other advanced economies. Our study identifies these differences and seeks to understand the current need for mergers and acquisitions in the European banking sector. Through the use of natural language processing techniques and sentiment analysis models, our study defines the linear correlations between the textual data found on European bank annual reports and the likelihood of banks being subject to a merger or acquisition. The correlations of different sections of an annual report are analysed. In order to identify these different sections, clustering algorithms are applied to the text data extracted from the annual reports. The data shows that certain sections have different effects on the resulting likelihood. Findings suggest that sections covering financial statements and risk management have the most impact on the resulting merger or acquisition likelihood.

Keywords — Banking, Mergers and Acquisitions (M&A), Consolidation, Natural Language Processing (NLP), Sentiment Analysis, Clustering, European Banks

Contents

List of Tables

List of Figures

Listings

1 Introduction

The following section defines the objectives of our study and interprets the significance of examining the current and future state of the European banking sector, and the effects of consolidation of banks in Europe.

1.1 Objectives

The aim of this study is to analyse the European banking market and identify the relationship between financial sentiment in European bank annual reports and the likelihood of each bank undergoing a merger or acquisition (M&A). This is the first study to apply such techniques to the European banking sector, and as such we seek to lead the way in the advancement of this field, and prove the effectiveness of applying natural language processing and artificial intelligence to such problem.

1.1.1 Main Objective

Through the use of sentiment analysis, our study aims to find a correlation between financial sentiment in annual reports from European banks and the likelihood of a merger or acquisition taking place. In order to find this possible relationship, we will make use of a pre-trained lightweight sentiment analysis model based on BERT. The phrases inside each annual report will also be classified into separate groups or 'clusters' based on the information being presented, to further investigate the relationship between M&As and the individual sentiment of each section of an annual report.

1.1.2 Secondary Objectives

The project has other objectives that expand upon the original vision of the main objective, and propose new lines of study based on the original findings:

- Analyse the need for cross-border M&As in the European market
- Identify different sections of annual reports through artificial intelligence
- Define the correlation between each section of the report and the target variable

- Predict future M&A candidates through financial sentiment

1.2 Motivation for the study

There is a growing need for restructuring inside the European banking market. Since the 2008 financial crisis, a third of banks operating in the Euro Area have disappeared (Figueiras, Isabel et al., 2021), and still most banks suffer from low profitability and excessive costs. There are several reasons that have led to the current banking climate. Imposed requirements on leverage ratios and levels of reserve capital are strict, as dictated by Basel III (detailed in section 2.3.1) under the supervision of the Basel Committee on Banking Supervision (BCBS). Mergers and acquisitions are regarded as a solution to combat low profitability and reduce overcapacity (Figueiras, Isabel et al., 2021). In fact, mergers and acquisitions (M&As) would result in an increase in efficiency and stability (Andreeva, 2019).

Banks are required to keep large amounts of equity, with the goal of putting the risk burden on shareholders instead of account holders. These practices ensure the survival of the banking sector during periods of economic downturn, but they also severely limit a bank's ability to make profit due to the unusually large weighted average cost of capital (WACC).

Differing from other developed peers, the European banking sector presents a much more fragmented system (Figueiras, Isabel et al., 2021) riddled with different regulations and concerns for sovereignty of assets by each member State. Many national M&As have already been established across Europe, and there is a growing need for cross-border mergers in the continent. This is an uphill battle when all member States have worrying concerns for the sovereignty of their own assets when dealing with foreign banks.

In such a climate, it is key to identify potential acquisition candidates, both for national and cross-border mergers and acquisitions in the European banking sector. Previous studies have shown a correlation between bank negative reporting sentiment and an eventual merger, applied in the US banking sector (Katsafados et al., 2021), but no studies have yet attempted to analyse the European banking sector thought NLP techniques.

The main motivation for this study is to further investigate the findings of Katsafados et

al. (2021), advance the field of natural language processing (NLP) and prove the useful application of NLP for financial analysis.

In addition to this, the study will serve as a thorough research to eventually publish an article on the findings of sentiment analysis applied to the identification of M&A likely participants in the European market.

1.3 Methodology

Aiming to test existing theories on the relationship between sentiment and banking M&A participants (detailed in section 1.1.1), the study's main objective will require a deductive methodology. The study will expand upon the existing state of the art bringing in new approaches to natural language processing of financial data.

The secondary objectives of the study (detailed in section 1.1.2) require their own methodology. Analyzing the need for mergers and understanding the benefits of further consolidation will also entail a deductive methodology: reviewing research and financial reports on benefits and disadvantages of such operations, both for individual banks and for the market as a whole. On the other hand, the attempt at building a prediction model for future M&A candidates constitutes both deductive and inductive methodology.

1.3.1 Software Tools

Most of the development will be carried out using the Python programming language, a multi-paradigm language with extensive documentation in data science. Specifically, the project will make use of Python 3.8 even though versions 3.9 and 3.10 exist, because version 3.8 is widely regarded as a stable version of the language for which most libraries have complete compatibility, and the version still receives long term support and security updates.

The most relevant python packages used are detailed below:

Numpy This library adds support for a wide array of mathematical operations with multi-dimensional arrays and matrices. The code base is written Python and C, a low-level programming language that provides efficiency and optimization (The NumPy community, 2022). Numpy offers speeds comparable to operations made

in the MATLAB language.

Pandas The Pandas library is designed for data analysis and manipulation in form of series and dataframes. Similar to Numpy it is highly optimized for performance and partially written in C (McKinney and Pandas Development Team, 2022).

NLTK The Natural Language Toolkit is a group of libraries dedicated to natural language processing (mainly in the English Language, with growing support for other languages) (Bird, Klein, and Loper, 2019). It offers support for many fields of study, and implements basic functionalities needed for any NLP study, including but not limited to parsing, tokenization, stemming, PoS tagging, etc.

SpaCy Powerful natural language processing library that implements complex models that are part of the current state of the art. It makes use of neural network libraries like Tensorflow and PyTorch to run these specialised text analysis models.

Scikit-Learn A Python library for machine learning built on top of SciPy. It offers a wide range of supervised and unsupervised learning algorithms for a variety of data analysis and machine learning tasks (Scikit-learn developers, 2022). It offers support for both supervised and unsupervised learning models. This study will used unsupervised clustering models to find clusters of phrase topics, and will use supervised learning regression and classification models to predict and find the correlation between sentiment and mergers.

Gensim Gensim is a Python library for topic modeling, document indexing and similarity retrieval with large text bases (Řehůřek et al., 2022). It implements various vectorizing models such as word2vec, in order to translate words into arrays of data.

The full list of python libraries and the versions used can be found in table 9 in the attachments section.

Development will take place in Jupyter Notebooks which are part of the Jupyter Project. Jupyter notebooks provide a web-based interface that can execute snippets of code and show results in an interactive platform.

1.4 Structure

The study will be carried out by building a foundation and reviewing existing knowledge on natural language processing, as well as mergers and acquisitions. Following this initial review, the document will cover the methodology used during the development and data processing phase. Finally, any results and conclusions will be analysed and documented. The full layout of the study can be found in the table of contents.

2 State of the Art

This section details the state of the art in the different fields relevant to this study.

2.1 Natural Language Processing

The origins of NLP date back to 1940, during the development of the first machine translation (MT) techniques, mainly between English and Russian, and later MT developments oriented to the Chinese language (Hutchins, 1986).

During the 1960s, iterations of primitive AIs based on Q-A (question & answer), such as BASEBALL (Green Jr., 1961), ELIZA and LUNAR (Woods, 1978), were presented in which users could launch a series of questions on which the AI was trained, and it was capable of returning a coherent answer. In the case of LUNAR, one of the objectives was to use artificial intelligence as an interface between the human being and a large database (Hendrix et al., 1978), without the need for the use of structured programming languages.

In the early 1980s, an academic current began to emerge dedicated to the study of language through mathematical techniques ("grammar theory"), seeking the ability to detect logic, meaning, search for the beliefs and intentions of the writer, and draw emphasis and theme from texts. At the end of the decade, the first general-purpose tools capable of parsing and processing texts began to emerge, such as "SRI's Core Language Engine" (Alshawi, 1994) and "Alvey Natural Language Tools" (Briscoe et al., 1987).

During the 1990s, great emphasis was placed on the statistical and probabilistic study of language (Manning, Schütze, and Weikurn, 1999), making heat maps and text correctors. Another topic of great interest turned out to be the simplification and synthesis of texts (Mani and (Eds.), 2000), looking for the parts of the text with the greatest amount of relevant information.

2.1.1 Current Fields of Study

Recent new areas of study that have emerged from the field of NLP are, among others: Sentiment analysis, "Speech to Text" (and "Text to Speech"), Named Entity Recognition

(NER), Emotion detection, etc.

Opinion analysis seeks to extract from the entire text a final score on the author's opinion (sentiment) with respect to a certain topic, whether it be the reviews of a store, the criticism of a novel, or the comments of a publication in the media. social media. Conventional opinion analysis (Yi et al., 2003) makes use of databases of words and expressions that refer to positive or negative opinions, in such a way that it makes an estimate based on previously supplied patterns. New opinion analysis models make use of Deep Learning techniques (deep neural networks) to classify opinions without the need to generate a database of negative and positive terms.

"Speech to Text" techniques consist of the recognition of spoken language to write a text. These techniques combine auditory recognition models based on neural networks or SVMs (Support Vector Machines), and use NLP probabilistic analysis to fill in gaps in cases of doubt, or to fill in the necessary punctuation marks.

Named Entity Recognition (NER) has a very necessary use on the Internet, since for algorithms such as the Google search engine or the classification of "trending topics" on Twitter (Ritter et al., 2011), these algorithms must know what person or issue is being talked about, even if it is written in different languages, is misspelled, and even in instances where the person being talked about is not directly mentioned. In the past, web pages had to write a list of words or search terms to help the Google search engine in its ranking of the most relevant pages.

Finally, emotion detection is similar to the sentiment analysis field of study, but is used to a greater extent to classify social media posts into 6 groups depending on their predominant emotion (Bhargava, Y. Sharma, and S. Sharma, 2016).

2.1.2 Applications of NLP

Natural language processing is popular in applications such as: Automatic translation, detection of fraudulent or spam emails, information extraction, generation of summaries (summarization), and question-answer (Q-A) applications.

The world is increasingly digital, with immediate access to all kinds of data from our homes with a simple search. One of the main barriers to this access to information is

language. The main function of machine translation is to convert sentences from one language to another like Google Translate or DeepL do, and the main challenge is not to change each word from one language to another, but to preserve the meaning. This is a constant struggle that improves with each iteration and each new paper published on it, but it is infinitely complex, unique for each language and in some cases illogical for the translation machine.

Applications capable of synthesizing texts, generating summaries and extracting information help to identify relevant data and entities: subjects, sites, events, dates, prices, etc. Clear identification of the most relevant data can help improve the efficiency and effectiveness of search engines or ML models based on text inputs. For the extraction of information, word count models and the application of multi-nomial models (McCallum and Nigam, 1998) have historically been used, after cleaning the words that do not provide relevant information.

2.1.3 Processing texts

Preprocessing tools are based on a series of steps to extract the information most relevant of each analysed text, seeking to reduce the processing load that entails parsing all the raw text, without this load reduction entailing a significant reduction of information. These techniques include, among others, the segmentation of phrases, tokenization, stemming, lemmatisation, removal of stopwords, etc. (Khurana et al., 2017)

2.1.3.1 Tokenization

Tokenization is the process of breaking down a string of text into individual tokens. These tokens can be words, numbers, punctuation, or other pieces of text (sentences, paragraphs, pages, etc.). Tokenization is a common pre-processing step in most Natural Language Processing (NLP) applications.

2.1.3.2 Stopword filtering

Stopword elimination or filtering is the process of removing words that are not important to the meaning of a text. This can be done by removing words that are not essential to the meaning of the text, such as "a" and "the", or by removing words that are not relevant to the specific topics of interest being analysed. These words are typically high

frequency words, therefore it is important to filter them out in order to reduce the size of a text document and to improve the efficiency of text processing.

2.1.3.3 Stemming & Lemmatisation

Stemming is the process of reducing a word to its base or root form, which is the part of the word that is common to all its inflected forms. For example, the stem of the word "running" is "run", and the stem of the word "runs" is also "run". This is done in order to normalize words for comparison, since words with the same stem often have the same meaning. Stemming is a simple process of cutting suffixes and prefixes, with a very simple set of instructions, valuing efficiency over accuracy.

Lemmatisation is also a process of reducing a word to its base form. Lemmatisation is a much more labor-intensive process with many more rules, and takes into account irregular forms of words. This is done by removing inflectional endings, such as "-ed" and "-ing", and by returning the word to its dictionary form, which may be different from its original form, such as "ran" becoming "run". This process requires finding the lemma, or dictionary form, of a word, which can be a labor intensive process if the aim is to cover every single word in a language. This process values accuracy over efficiency, and there are many variations on the instruction set and amount of complexity.

2.1.3.4 Vectorization & Word embeddings

Vectorization is the process of representing text as numerical vectors. This can be done in a number of ways, but the most common is to create a vector for each word in the text, and then represent each sentence or document as a vector of the sum of the vectors of its words (Bag Of Words). This process can be further refined by weighting the vectors according to their importance in the text, or by using more sophisticated methods such as word embeddings.

Word embeddings are a type of word representation that allows words with similar meaning to have a similar representation. They are a distributed representation for text that is perhaps one of the key breakthroughs for the impressive performance of deep learning methods on challenging natural language processing problems.

Word embeddings are in many ways similar to the distributed representations of words

that are used in neural language models. A neural language model predicts the next word in a sentence, given the previous words in the sentence. The hidden layer in the neural language model is a continuous vector representation of the input words, and this vector representation captures the syntactic and semantic relationships between the words.

The key difference between word embeddings and the hidden layer representations in neural language models is that the word embeddings are learned from data, while the hidden layer representations are learned from the neural network architecture itself.

Word embeddings are typically learned from large amounts of unannotated text data. The most popular word embedding models are the word2vec models introduced by Mikolov et al. (Mikolov et al., 2013). There are two main types of word2vec models: the continuous bag-of-words model (CBOW) and the skip-gram model.

The CBOW model predicts the current word given the context of surrounding words, while the skip-gram model predicts the surrounding words given the current word. The CBOW model is typically faster to train than the skip-gram model, but the skip-gram model is more accurate for infrequent words.

Both the CBOW and skip-gram models are trained using a hierarchical softmax, which is a neural network architecture that is efficient for training word embeddings.

Once the word embeddings are learned, they can be used in a variety of ways. The most common way to use word embeddings is to initialize the weights of a neural network for a natural language processing task. For example, word embeddings can be used to initialize the weights of a Long Short-Term Memory (LSTM) network for part-of-speech tagging.

Word embeddings can also be used as features in a traditional machine learning model. For example, the word2vec models can be used to train a logistic regression model for sentiment analysis.

Finally, word embeddings can be used to create a word similarity task, where the goal is to predict how similar two words are. This can be useful for finding synonyms or similar words.

2.2 Sentiment Analysis

Sentiment analysis is a branch of NLP that deals with extracting opinion from text. The main task of sentiment analysis is to determine the overall sentiment of a text, be it positive, negative, or neutral. It is a valuable tool for a variety of applications, such as opinion mining, market research, and customer service. It can be used to understand what people think about a product, service, or brand, and it can also be used to identify areas of improvement. There are various methods for performing sentiment analysis, such as rule-based methods, lexicon-based methods, and machine learning-based methods.

2.2.1 Rule-based methods

Rule-based methods involve manually defining a set of rules that are then used to classify text as positive, negative, or neutral. This approach is often used when dealing with small amounts of data, or to prioritise speed over accuracy.

2.2.2 Lexicon-based method

Lexicon-based methods make use of pre-existing lists of words that are associated with positive or negative sentiment. One such model that has become very popular in recent years is VADER (Hutto and Gilbert, 2014). These lists can be generated manually or through algorithms. This approach is often used when dealing with larger amounts of data.

Common approaches also identify modifying words such as 'intensifiers', that denote a stronger sentiment. These words can act as multipliers, making a positive sentiment stronger, but also making a negative sentiment even more negative when the same word is used.

2.2.3 Machine learning-based methods

Sentiment analysis has been traditionally approached using rule-based methods, where a set of hand-crafted rules is used to identify and classify emotions in text. However, rule-based methods are often limited in their accuracy and ability to adapt to new data. Machine learning-based approaches have emerged as a promising alternative, as they can

learn from data and improve their performance over time.

Machine learning-based methods make use of algorithms to learn automatically from data and identify patterns that can be used to classify text not only as positive, negative, or neutral; but can detect a wide array of emotions or patterns to classify texts. This approach has been gaining in popularity for its overwhelming improvement in accuracy across many fields of sentiment research.

The most common emotions that are targeted are those of joy, anger, fear, sadness, and disgust. However, there is a growing body of work that is also looking at more nuanced emotions such as love, hate, trust, and doubt. One of the challenges in machine learning-based sentiment analysis is that emotions are often expressed in a subtle and complex way. For instance, a sentence may contain irony, sarcasm, or other forms of figurative language that can be difficult for a machine to understand. Additionally, the same words can often be used to express multiple emotions, depending on the context in which they are used. This can make it difficult for a machine learning algorithm to learn how to accurately identify emotions from text.

Despite these challenges, machine learning-based sentiment analysis has seen significant progress in recent years and is now used in a variety of applications. For instance, it is commonly used to analyse customer reviews to identify positive and negative sentiment. Additionally, it is used to analyse social media data to track the emotions of users over time.

2.2.3.1 BERT

BERT is a pre-trained transformer model that was introduced in 2018 (Devlin et al., 2018). It was developed by Google and is based on the transformer architecture. The model can be fine-tuned on a variety of tasks, including sentiment analysis.

Sentiment analysis is the task of classifying a text as positive, negative, or neutral. It is often used to gauge the public opinion on a given topic. BERT can be used for sentiment analysis by first fine-tuning the model on a large dataset of labeled texts. Once the model has been trained, it can be used to predict the sentiment of new, unseen texts.

BERT has been shown to outperform other state-of-the-art models on a variety of sen-

timent analysis tasks. In one study, BERT was found to achieve an accuracy of 95.63% (Sun et al., 2019) on a dataset of 50,000 IMDB movie reviews (Maas et al., 2011). This is significantly higher than the previous state-of-the-art model, which achieved accuracy below 90%.

BERT has also been used to build a sentiment analysis model for social media data. In one study, BERT was used to fine-tune a model on a dataset of 1.6 million tweets (Hovy et al., 2016). The model was found to achieve an accuracy of 77.5%.

In conclusion, BERT is a powerful tool for sentiment analysis. It has been shown to outperform other state-of-the-art models on a variety of tasks.

2.2.3.2 DistilBERT

Developed by a team of researchers at Google AI (Sanh et al., 2019), it is a distilled version of BERT, a popular pre-trained model for natural language processing (NLP). DistilBERT is smaller and faster than BERT, while retaining much of its predictive power. The model is trained using a distillation technique known as knowledge distillation, which involves training a small model to mimic the output of a larger model.

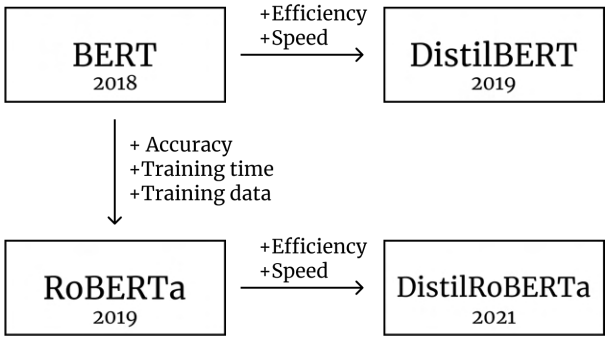
DistilBERT maintains 97% of BERT's original accuracy and still manages to reduce the model's original size and complexity by over 40%.

2.2.3.3 RoBERTa & DistilRoBERTa

The RoBERTa (which stands for "Robustly Optimized BERT Pre-training Approach") model is an improvement over the BERT model, it has been subjected to much longer training times and using much more data, and presents a much higher accuracy. It was introduced by the Facebook AI team (Liu et al., 2019) following the development of BERT. The RoBERTa model also has a much faster training time.

DistilRoBERTa follows the same training procedure as DistilBERT but applied to the RoBERTa model instead. According to its developers, the model has 6 layers, 768 dimension and 12 heads, totaling 82M parameters (compared to 125M parameters for $RoBERTa_{BASE}$). On average, DistilRoBERTa is twice as fast as $RoBERTa_{BASE}$.

Figure 1: Evolution of BERT-based models



Source: Own elaboration

It is important to note that there are a wide variety of BERT-based models apart from the ones shown in figure 1, including ALBERT, ELMo, XLNet, etc.

2.3 Mergers & Acquisitions in the Banking Sector

The banking sector has seen a significant increase in mergers and acquisitions (M&A) activity in recent years. This is driven by a number of factors, including the need to increase scale and efficiency in the face of intensifying competition, regulatory pressures and the low interest rate environment.

M&A activity in the banking sector reached a record high in 2015, with 1,453 deals worth a total of US$2.5 trillion, according to data from Thomson Reuters. This represented a significant increase from the previous year, when there were 1,111 deals worth US$1.6 trillion.

Data provided by Thomson Reuters also shows that the most active countries for M&A in the banking sector in 2015 were the United States, the United Kingdom, China, France and Germany. M&A activity in the banking sector is expected to continue to be driven by the need to increase scale and efficiency. This is particularly important in the current environment of intensifying competition and regulatory pressures such as the Basel III framework detailed in section 2.3.1. In addition, the recent low interest rate environment has put great pressure on banks' margins, making M&A a more attractive option for increasing profitability.

2.3.1 Basel III regulatory framework

The Basel III agreement is an international regulatory accord that introduced a set of reforms designed to improve the regulation, supervision and risk management within the banking sector (EBA - European Banking Authority, 2021). The agreement was reached by the members of the Basel Committee on Banking Supervision in 2010 and implemented in phases between 2013 and 2019.

The Basel III reforms were introduced in response to the global financial crisis of 2007-08, which revealed shortcomings in the existing Basel II framework. The crisis highlighted the need for banks to hold more capital and to improve their risk management practices.

The key provisions of Basel III include:

- Requiring banks to hold more Tier 1 capital, which is the core capital that can absorb losses in a crisis.
- Introducing a new capital conservation buffer of 2.5% of risk-weighted assets, to be met from Tier 1 capital. This buffer is designed to ensure that banks have a cushion of capital to draw on in times of stress.
- Introducing a new global liquidity coverage ratio (LCR) of 100%, to ensure that banks have enough high-quality liquid assets to cover their net cash outflows over a 30-day period of stress.
- Introducing a new net stable funding ratio (NSFR) of 100%, to promote safe and sustainable funding practices.
- Enhancing the supervisory and regulatory framework, including new powers for national authorities to require banks to hold additional capital.

The Basel III reforms have been designed to make the banking sector more resilient to shocks and to protect taxpayers from having to bail out failing banks.

2.3.2 FinTechs

FinTech is short for Financial Technology. It includes any technology that is used to help provide financial services (Schueffel, 2016). This can include anything from mobile apps

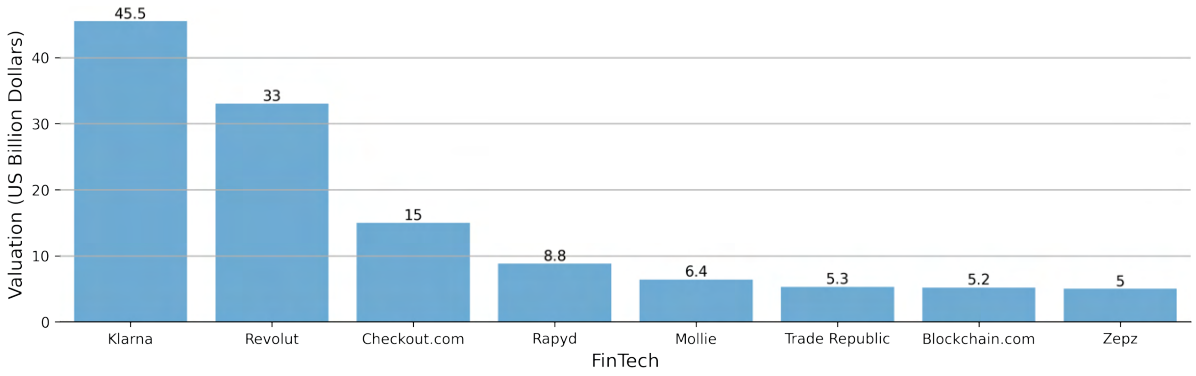
that help you budget to complex algorithms that are used by investment banks to trade stocks.

The term FinTech was first coined in the early 1990s (Buckley, Arner, and Barberis, 2016), but the concept has become much more prevalent in recent years as the technology landscape has changed dramatically (Schueffel, 2016). In the past, most financial services were provided by large institutions, such as banks and insurance companies. However, with the advent of the internet and mobile devices, there has been a shift towards digital-based services that are often provided by startups and small businesses.

This has led to the creation of a whole new industry known as FinTech. FinTech companies often use cutting-edge technology to provide innovative services that are often more convenient and affordable than traditional financial services.

There are a wide range of FinTech companies, but some of the most popular include companies that offer mobile payments, peer-to-peer lending, and investment management. As of August 2021, the biggest FinTechs in Europe were Klarna (based in Sweden) and Revolut (based in the UK) (Norrestad, 2021), valued at $45.5B and $33B respectively, as shown in fig. 2. Other notable mentions operating in Spain are N26 (based in Germany) and BNext (based in Spain).

Figure 2: Top European FinTechs Valuations in Aug. 2021



Source: Own elaboration from Norrestad, 2021

These companies circumvent many regulations put in place by financial authorities and can therefore offer services at a fraction of the price that traditional banks can offer. Traditional banks are no longer in a competitive position and have little leg room to reduce their margins further and offer the same services as FinTechs.

On the other side, these FinTechs, escaping the regulations of regular banks, put their clients at higher risks that could propagate through the whole economy during periods of downturn. The strict regulations put in place for banks (such as BASEL III) were established after the 2008 financial crisis to prevent a similar situation from destabilizing the entire financial system. Circumventing these practices may be an easy way to offer competitive prices, but it is also a way of shifting the risk of default onto the consumer instead of on shareholders.

The rise of FinTechs has put even more pressure on traditional banks to find ways of reducing overhead costs and expand their business. This has made the need for mergers even clearer than before.

2.3.3 Benefits and downsides of M&As

There are a number of pros and cons to mergers and acquisitions in the European banking sector, detailed by the ECB (European Central Bank, 2000).

On the plus side, mergers and acquisitions can lead to increased market share and scale for the banks involved. This can in turn lead to increased profits and shareholder value. Additionally, mergers and acquisitions can provide the opportunity for cost savings through economies of scale. Banks with similar offerings in a given territory can often consolidate assets and resources, and use half as many resources (physical offices, workforce, legal structures, etc.) to offer the same service as before, becoming more competitive (Calipha, Tarba, and Brock, 2010).

On the downside, mergers and acquisitions can be expensive and can lead to disruptions in service for customers. Additionally, there can be cultural clashes between the employees of the different banks involved (Weber, Tarba, and Reichel, 2011), not to mention the pressure from authorities to preserve sovereignty over national assets when cross-border M&As take place (Datta and Puia, 1995). Finally, there is always the potential for antitrust concerns to arise inside given territories.

2.3.4 The European Market

The European banking sector is currently undergoing a period of consolidation (Figueiras, Isabel et al., 2021). This is due to a number of factors, including the need to increase

efficiency and profitability in the face of tougher regulation and competition, as well as the need to meet the growing demand for financial services from consumers and businesses.

M&As can help banks to achieve these goals by providing access to new markets, customers and products, as well as economies of scale and scope. They can also help to create a more efficient and effective banking sector, which is better able to meet the needs of European consumers and businesses.

However, M&As are not without their challenges, and banks need to carefully consider the risks and benefits before embarking on any deal. They also need to ensure that they have the necessary resources and capabilities to make the most of any opportunity.

2.3.4.1 Cross-border M&As in Europe

Cross-border mergers in Europe are a great resource in order to create a level playing field for banks operating in the European Union. By consolidating banks across borders, banks are able to reduce costs and compete more effectively with one another and with rising FinTechs (Section 2.3.2). This, in turn, helps to create a more stable and efficient banking system in Europe.

While the business case for consolidation seems clear, there are a number of obstacles preventing mergers and acquisitions, which arise from various economic, political, regulatory and cultural factors. These obstacles are related to the business environment in which European banks currently operate, and are made worse by global and regional geopolitical uncertainty and rapid changes in the global banking industry's outlook due to innovation and technological change.

Member States of the EU often raise concerns over bank consolidation because of fears of loss of sovereignty over their own assets if many banks consolidate in Europe (Boer and Portilla, 2020), with smaller states depending on foreign banks completely. The concerns of all member States need to be addresses by the EU before the necessary consolidations can be made across all of Europe.

2.3.5 AI applied to M&A Detection

The detection of M&A targets and acquirers is a field of study where many advancements have been made in recent years using a wide array of techniques. A recent study (Venuti, 2021) used graph models detailing entities and relationships to predict mergers and acquisitions of companies based on these connections between organisations.

There are very few studies covering the identification of M&A participants through the use of machine learning techniques. One study used SEC filings to extract text data and train a text regression model based on word frequencies to detect merger participants (Routledge, Sacchetto, and Smith, 2013). The study used pure statistics-based models relying on word frequency to make its predictions, and it already showed very promising results. In addition to this, the study by Katsafados et al. (2021) mentioned throughout the introduction has also shown promising results by applying NLP techniques to reports filed by American banks. Both these studies have been solely applied to data originating from American companies.

Just last year Miguel Crespo, a Law & Business Analytics student from Comillas Pontifical University, studied the application of different clustering techniques on European Banking data to uncover possible relationships between the resulting clusters and the likelihood of a merger or acquisition taking place (Crespo, 2021). It is the first study that uses clustering techniques from financial variables to identify potential banking M&A participants, specifically in the EU banking industry. Instead of directly predicting the target using supervised learning models, he provided a framework to predict the cluster a bank should be allocated to, and he proposed specific cross-border mergers that are most likely to occur in the short-medium term, looking for synergy between banks from different clusters, combined with traditional financial and strategic analysis.

Our study is the first to approach the analysis of European banking M&A likely participants through the use of NLP and sentiment analysis. Our study seeks to keep iterating on this field and improve on the findings of studies like Routledge, Sacchetto, and Smith (2013) and Katsafados et al. (2021) that focus on applying natural language processing techniques to this field, as well as offer a new perspective on the methodology used to process and analyse the text data.

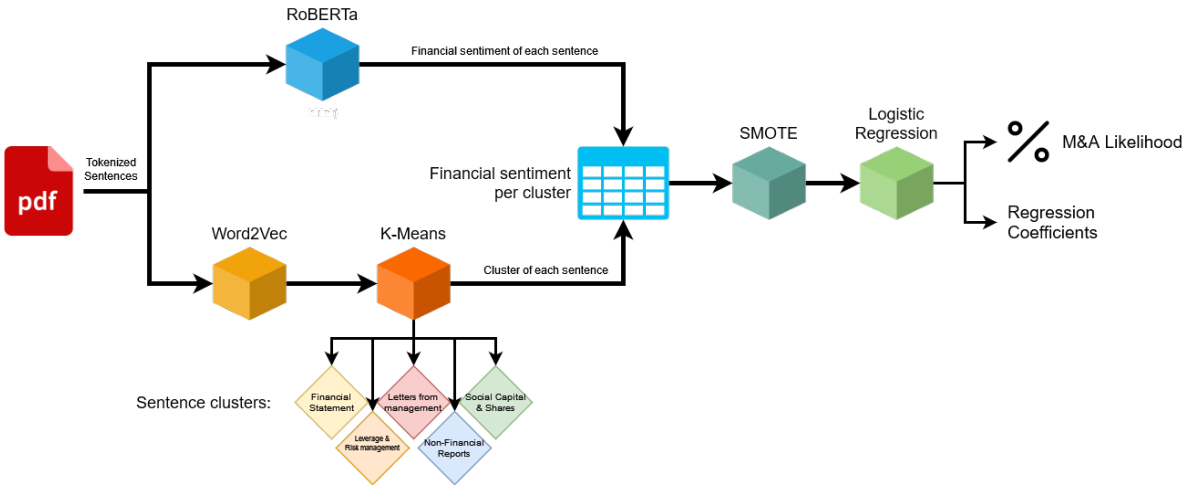
3 Methodology

This section details the methodology followed during the development phase of the study, and the results found throughout the project.

3.1 Summary of Architecture

The different phases of data processing and models detailed in the study are summarised into a single diagram in order to understand the flow of information from one point to another in the different phases. This diagram is shown in figure 3.

Figure 3: Diagram of complete data architecture & models



Source: Own elaboration

The diagram above shows the complete process followed by the data present in an annual report PDF. Each PDF is initially separated (tokenized) into individual sentences, which are processed separately throughout the data pipelines. These data pipelines and models used will be detailed in the following sections.

3.2 Data Processing

The data used for this study constitutes 289 annual reports from 28 different banks operating Europe.

Table 1: European Banks present in Dataset

Bank	Starting Year	Final Year	Phrases Analysed
ABBEY NATIONAL	2003	2008	20,407
BANKIA	2014	2020	26,545
BANKINTER	2012	2021	28,548
BANK BGZ	2008	2016	12,800
BARCLAYS	1999	2021	131,896
BBVA	2014	2021	27,508
BNP PARIBAS	2009	2020	19,977
CAIXA	2014	2021	18,429
COMMERZBANK	2006	2014	45,074
CREDIT AGRICOLE	2008	2021	26,075
CREDIT SUISSE	2007	2021	46,887
DEUTSCHE BANK	2007	2018	78,193
DEUTSCHE POSTBANK	2007	2017	75,678
DRESDNER	2003	2008	12,075
ABANCA	2014	2019	22,656
BANCO VALENCIA	2011	2012	5,748
CATALUNYA BANC	2013	2015	8,439
HSBC	2010	2021	136,365
ING	2018	2018	9,847
INTESA	1999	2006	36,375
NATWEST	2005	2021	170,207
NOVO BANCO	2014	2019	26,623
POPULAR	2009	2016	5,919
SANPAOLO	1999	2006	57,739
SANTANDER	2009	2019	115,845
SOCIETE GENERALE	2017	2021	7,101
UNICREDIT	2011	2021	86,862
WGZ BANK	2013	2015	18,351

Source: Own elaboration

3.2.1 Raw Processing

Each annual report is in PDF format, and files are read one by one using the PyPDF2 python library. The data is extracted page by page in raw format. In order to clean this data, regular expressions (known as 'regexp') are used, to filter out unwanted characters, line breaks, numbers, etc. Once the text is filtered and clean, it is separated or tokenized into individual sentences, resulting in a dataset that is separated by bank, year, page and sentence.

It is important to note that the process of automatically separating or tokenizing texts

into sentences is a harder task than it might seem. Sentences are identified by either line-breaks of full stops, but can be cut off by pages, or formatted differently by the PDF, or can suffer any other range of changes that make the task harder. Therefore, many sentences appear incomplete or cut off, but this does not invalidate the analysis, as the dataset is so big that these small errors have little effect on the outcome.

Table 2: Sample data (chosen at random) from filtered bank sentences

Bank	**Year**	**Page**	**N**	**Sentence**
Barclays	2009	96	6	Finally additional analysis of debt...
		99	7	Corporate accounts that are deemed...
		145	36	Annual Report Barclays PLC Annual...
		153	87	Meetings were held with our instit...
		217	13	Other services Fees payable for the...
		244	5	The Group has agreed funding contr...
BNP Paribas	2019	24	1	A major strategic alliance in Latin...
		32	21	These ratings which take into account...
La Caixa	2016	81	16	The restructuring cost was EUR...
		92	17	The gross income of the investments...
Commerzbank	2013	62	25	This was mainly the result of a...
		91	25	The service which is offered in...
		126	46	Minimum ratings establish the qualit...
		235	6	Notes Responsibility statement by the...
HSBC	2016	20	34	This was partly offset by lower...
		163	7	The amount is disclosed on a paid...
		256	31	Perpetual subordinated capital sec...
Societe Generale	2019	24	16	Societe Generale is helping to meet...

Source: Own elaboration

Table 2 is the result of picking sample data at random from 6 banks from the dataset from specific years. The entire dataset including the 28 banks and 289 annual reports comes to a total of 1,202,495 different sentences. The total processing of this raw data is a labor-intensive process. The processing was timed over several runs, with the total amount of time needed to cover the entire data set being 30 minutes, 9 seconds and 593 milliseconds.

3.2.1.1 Optional translation layer

Certain defunct banks only had their annual reports published in their local language (different from English). These sentences are sent through an additional layer of processing, being translated automatically using the Google Translate API (through the adapted

API provided by the 'googletrans' python library).

Translating each phrase is a slow process that depends on the response of the API, the network connection etc. In terms of processing times, sentences processed in English can be cleaned and prepared at a rate of 5842.47 sentences/s, whereas phrases passing through the translation layer can be processed at a mere 1.68 sentences/s.

3.3 Financial Sentiment Analysis

Once all phrases have been processed correctly, they can be sent through the neural network in charge of predicting the financial sentiment. The model in question is based on DistilRoBERTa (2.2.3.3), based on a distilled version of Facebook AI's RoBERTa which itself is based on Google's original BERT (2.2.3.1) architecture. The details pertaining each model can be found in section 2.2.3. The model's full name is "`distilroberta-finetuned-financial-news-sentiment-analysis`", and has had further training on a financial phrasebank (Malo et al., 2014) dataset that categorises financial news into 'positive', 'neutral' and 'negative' sentiment.

Figure 4: Financial phrasebank example data point


```
{
  "sentence": "Pharmaceuticals group Orion Corp
               reported a fall in its third-quarter
               earnings that were hit by larger
               expenditures on R&D and marketing .",
  "label": "negative"
}
```


Source: Own elaboration

In order to process sentences, each word has to be translated into tensors. A Tensor is a mathematical object that represents a generalization of vectors and matrices to potentially higher dimensions. In essence, a tensor is a multidimensional array of data. The specific properties of a tensor, such as its number of dimensions, its shape, and its size, define how the data is arranged in the array.

In physics, tensors are used to describe the physical properties of objects in space-time. In engineering, tensors are used to describe the stresses and strains on objects. Much in the same way, machine learning tensors are used to represent data as it makes its way through the layers of a neural network, representing any number of dimensions needed,

and describing the relationship between different neurons as the network learns from the data it observes.

In order to turn the data into tensors we use a pre-trained auto-tokenizer which generates word embeddings (explained in section 2.1.3.4) for each phrase to input into the model. The model consists of 82 million different parameters. Therefore, the model takes a long time for each prediction, averaging 50.56 predictions per second which at first can seem like a fast rate, but at this rate the entire dataset is processed in 6 hours, 36 minutes and 19 seconds.

The resulting prediction outputs three different parameters:

Figure 5: Financial sentiment prediction output example


```
"input" : {
   "sentence": "Operating profit totaled EUR 9.4 mn
                down from EUR 11.7 mn in 2004 .",
}
"output" : [
  {
    "label": "negative",
    "score": 0.9987391829490662
  },
  {
    "label": "neutral",
    "score": 0.0007296210387721658
  },
  {
    "label": "positive",
    "score": 0.0005312705179676414
  }
]
```


Source: Own elaboration

Observing figure 5 we can clearly see that the three output parameters range from 0 to 1, and represent the probability of each phrase being negative, neutral or positive; with the sum of the three parameters totalling 1.

After processing the entire dataset, we get these three parameters for all sentences. By ordering data in terms of sentiment, the effectiveness of the neural network model can be appreciated:

Table 3: Sample from highly negative rated sentences in dataset

Sentence	% **negative prob.**
"Financial performance of the group: as expected, the WGZ BANK Groups operating profit fell considerably in the year under review"	99.8923
"Consolidated net income for the year therefore amounted to million euro with a considerable decrease compared to the previous year while the Parent Company's net income"	99.8922
"Overall the segment posted a pretax profit of m for which represents a year-on-year decrease of around a third"	99.8904
"although Spain's net profit fell significantly due to high impairment charges"	99.8898
"Investment Banking posted a pretax loss of CHF million for compared to pretax income of CHF million in"	99.8898
"posted pretax income of CHF million a decline of compared to mainly driven by fair value losses of CHF million on the Clock Finance transaction in"	99.8898
"Overall the Mittelstandsbank segment posted pretax profit of m for which represents a year-on-year decrease of"	99.8898
"Releases and recoveries in Corporate Banking continued to decline at m compared to m in the previous"	99.8895
"In Residential Property although reservations for new homes fell by more than compared with the decline was well below the French market average which fell by"	99.8895

Source: Own elaboration

It must be reminded that digits, percentage symbols and other non alphabetic characters are filtered from sentences, so certain sentences are harder to comprehend.

Positive sentiment phrases can also be extracted:

Table 4: Sample from highly positive rated sentences in dataset

Sentence	% **positive prob.**
"Global Banking Markets trading profit before tax increased by m to m m reflecting strong performance in both Rates and Equity business due to increased transactional flow arising from close cooperation with other parts of Abbey and the beneficial trading environment available from diverging spreads in an illiquid market"	99.9751
"Global Banking Markets non-interest income increased by m to m m which reflected strong performance in both Rates and Equity business due to increased transactional flow arising from close cooperation with other parts of Abbey and the beneficial trading environment available from diverging spreads in an illiquid market"	99.9747
"Wealth Management net interest income increased by m to m m reflecting higher liability balances in Cater Allen as a result of improved sales performance and higher cash balances under administration in James Hay"	99.9746
"Outcome reported risk adjusted revenue increased primarily due to favourable movements on the fair value of own debt"	99.9744
"Net interest income grew Rm to Rm Rm driven by growth in loans and advances and deposits at improved margins"	99.9743
"Wealth Management net interest income increased by m to m m rejecting higher liability balances in Cater Allen as a result of improved sales performance"	99.9743
"Financial Markets non-interest income increased by m to m m due largely to stronger results within Derivative Structured Products which benefited from favourable market conditions and also new external business through stronger sales and marketing teams including almost"	99.9743
"from the IRUs nonstrategic business net interest and current income was up around over the previous year"	99.9742
"addition revenue rose from increased net interest income driven by wider spreads due to higher interest"	99.9742

Source: Own elaboration

3.4 Traditional clustering of phrases

The aim of using clustering on the sentences is to extract the individual sentiment of each part of an Annual Report. To extract the different sections, a simple approach of dividing the dataset into sections is erroneous, as each country, region and even bank has a different format that presents the information in a different manner and in different order. To overcome this obstacle, the phrases can be classified using a non-supervised model into different sections in an efficient way.

3.4.1 Word2Vec

Word2Vec is a machine learning algorithm that takes in a corpus of text and outputs a vector space model. The model is made up of a set of vectors, each of which represents a word in the corpus. The vectors are generated such that they are close together in the vector space if the words they represent are often used together in the text, and far apart if the words are rarely used together.

In this case, Word2Vec is used to translate each sentence into numerical vectors, by applying the algorithm to each word in the sentence and then creating a vector that is the mean of all word vectors. Using this numerical representation of each vector, the data can pass through a traditional clustering algorithm to classify sentences by their content.

3.4.2 K-Means

The K-Means clustering algorithm is a method of cluster analysis that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean. It is an unsupervised machine learning algorithm that is used to cluster data points into a predefined number of clusters. The algorithm works by randomly initializing clusters centroids and then assigning each data point to the cluster that has the closest centroid. The centroids are then updated to be the mean of all the data points assigned to that cluster. This process is repeated until the centroids no longer change or a predefined number of iterations has been reached.

3.4.3 Optimal number of clusters

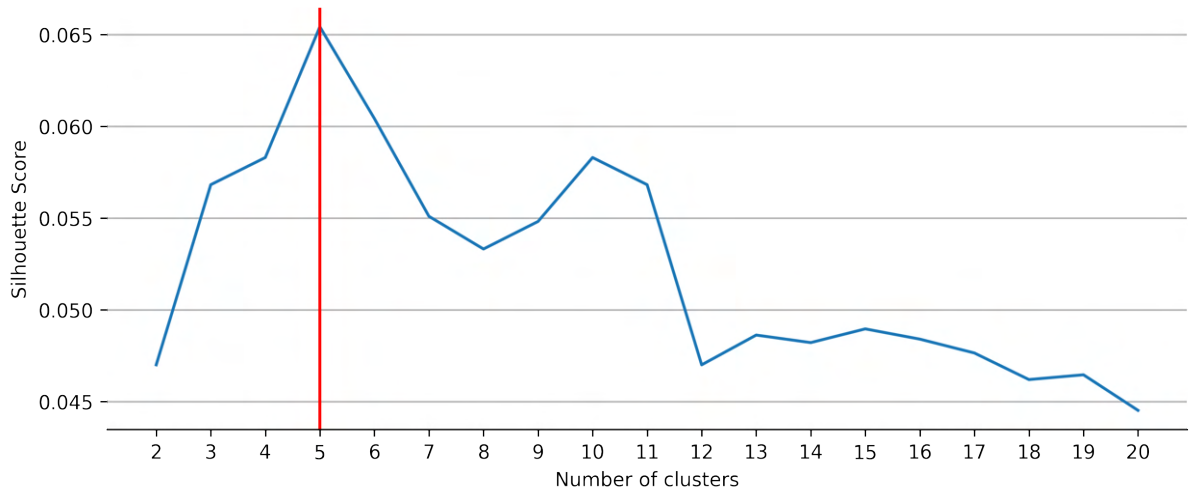
In order to choose the optimal number of clusters, the algorithm is executed on many different values of clusters, and the Silhouette Score is calculated. The silhouette score is a measure of how close each point in one cluster is to points in the neighboring clusters, and thus how well clustered the data is. The score is calculated using the mean intra-cluster distance (a) and the mean nearest-cluster distance (b) for each sample:

$$\text{Silhouette Score} = \frac{(b-a)}{max(a,b)}$$

The silhouette score is a measure of how well defined a cluster is. A high silhouette score means that the cluster is well defined (i.e., there is a small distance between points in the cluster and points in neighboring clusters). A low silhouette score means that the cluster is not well defined (i.e., there is a large distance between points in the cluster and points in neighboring clusters).

Iterating over the dataset with different values of the number of clusters, we arrive at the following results:

Figure 6: Silhouette scores for different numbers clusters



Source: Own elaboration

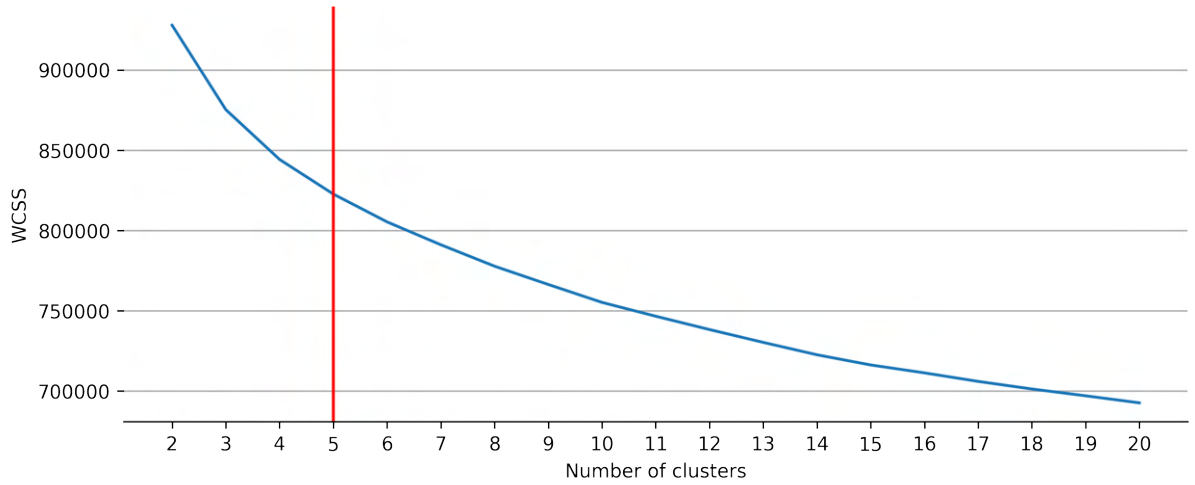
Figure 6 shows a clear winner in terms of maximizing the silhouette score, with a value of clusters k=5.

Another method of finding the optimal number of clusters is through the Elbow method. The Elbow method looks at the within-cluster sum of squares (WCSS or inertia) and

finds the point at which the WCSS begins to decrease at a slower rate. This point is the elbow of the WCSS curve and is the optimal number of clusters.

Iterating again over the dataset with different values of the number of clusters, we arrive at the following WCSS results:

Figure 7: WCSS distance for different numbers clusters (Elbow method)



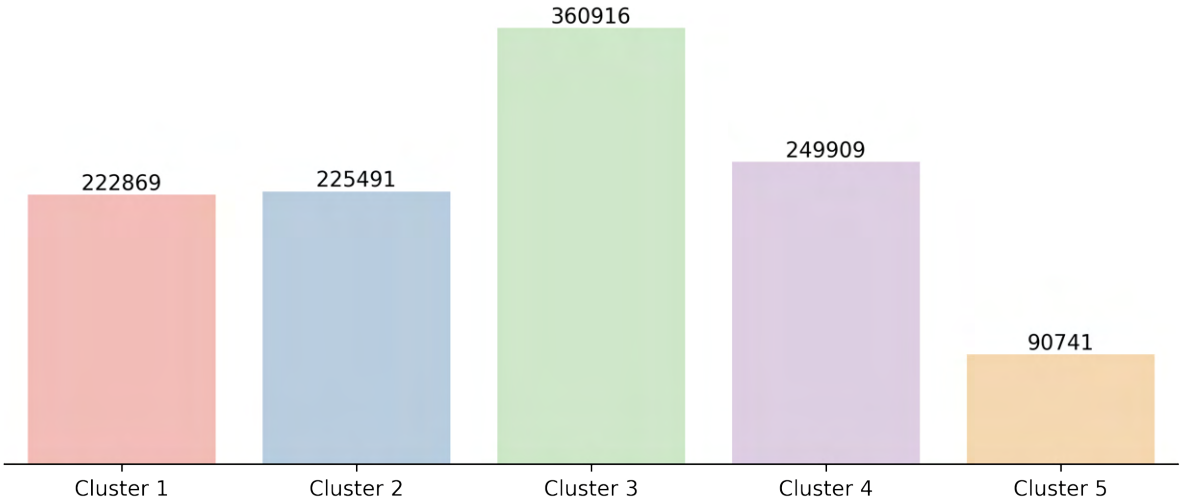
Source: Own elaboration

Figure 7 gives little information about what point is the start of a deacrease in the WCSS distance. Observing previous results from the silhouette method in figure 6, the sentences in the dataset will be categorised into 5 different clusters.

3.4.4 Clustering results

Following the choice of k=5 clusters, the results on the entire phrasebank are the following: phrases are distributed in a relatively uniform manner between clusters. It is important to check that no cluster has a small amount of phrases, as it would indicate that less clusters are needed. The opposite is true as well, a larger cluster than the rest could indicate that more clusters are needed.

Figure 8: Sizes of sentence clusters



Source: Own elaboration

3.4.4.1 Plotting results with t-SNE

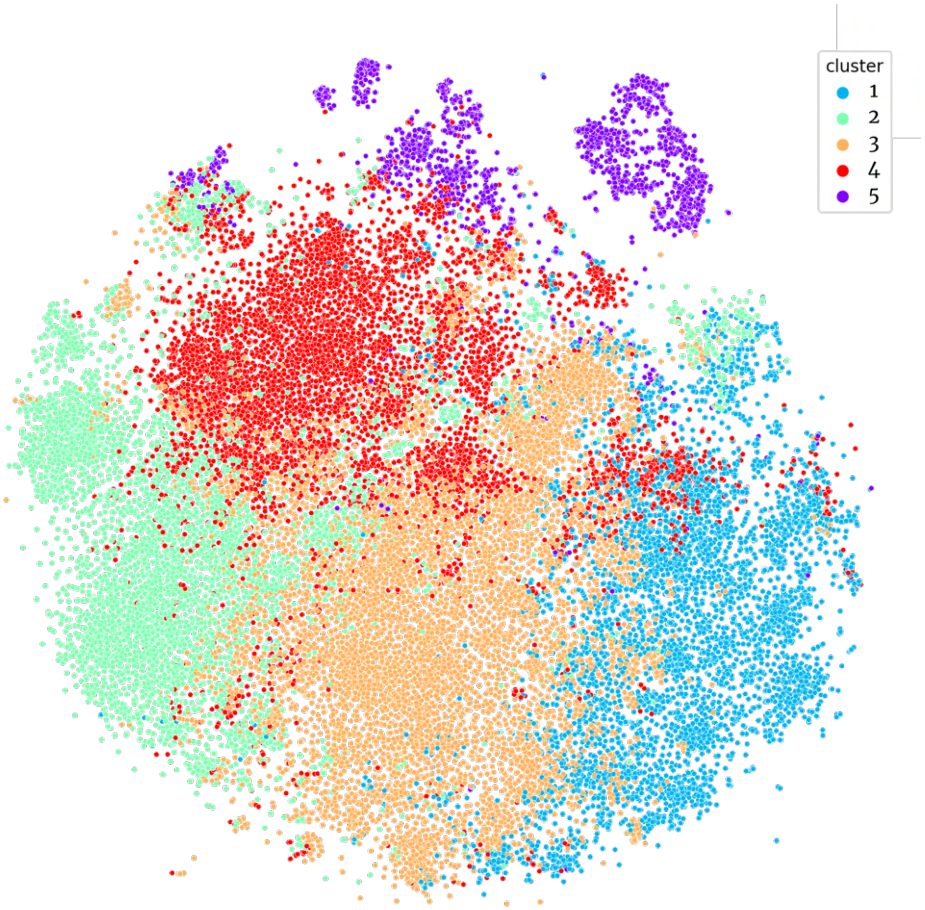
It is possible to plot datasets of higher dimensions onto a 2-dimensional space using t-SNE. T-distributed stochastic neighbor embedding (t-SNE) is a machine learning algorithm for visualization (Van der Maaten and Hinton, 2008). It is a nonlinear dimensionality reduction technique that is particularly well suited for visualizing high-dimensional data sets. The algorithm is based on a probabilistic model of the data set called the t-distributed stochastic neighbor embedding model. The model defines a probability distribution over possible mappings from the high-dimensional data to a lower-dimensional space. The mapping is represented as a set of points in the lower-dimensional space, and the probability of a data point being mapped to a point in the lower-dimensional space is given by a Gaussian distribution centered on that point. The t-SNE algorithm searches for a mapping that maximizes the likelihood of the data under the t-distributed stochastic neighbor embedding model.

The t-SNE algorithm has several advantages over other dimensionality reduction techniques. First, it is well suited for visualizing data sets with many variables. Second, the algorithm is able to preserve the local structure of the data set, which is important for visualizing data sets with complex structure. Third, t-SNE is relatively efficient, and can be applied to data sets with millions of data points. Finally, the algorithm is highly flexible, and can be used for a variety of tasks such as visualizing data sets with different

levels of detail, or for visualizing data sets with multiple views.

Using t-SNE, even if the original sentence arrays are 500-dimensional objects, the data can be visualized in a much more efficient way than other dimensionality reduction techniques such as PCA. The resulting clusters can be seen plotted onto the embeddings of 100,000 samples from the data set:

Figure 9: Clusters plotted using T-distributed stochastic neighbor embedding (t-SNE)



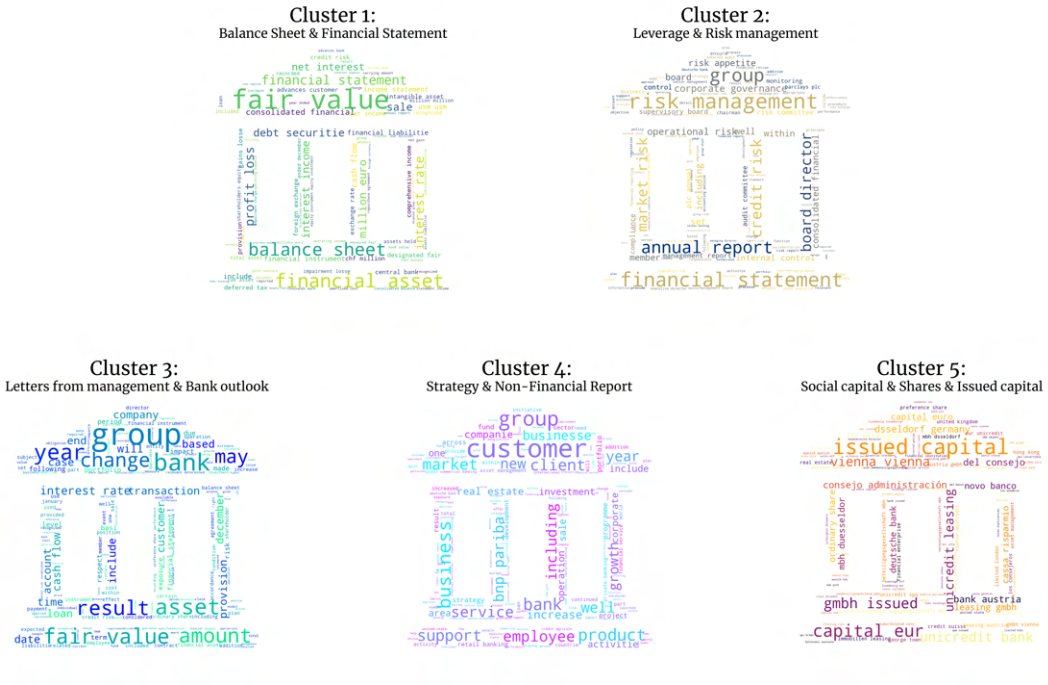
Source: Own elaboration

3.4.4.2 Identifying cluster characteristics

In order for the clustering to be of any use to this study, a name and set of defining features must be given to each cluster. In order to achieve this, word frequencies can be obtained from each of the clusters to find out the most defining words of each group. In order to present this data, the most efficient way is through the use of a wordcloud. A wordcloud is a graphical representation of the most common words in a given text. The

more common a word is, the larger it appears in the wordcloud.

Figure 10: Wordclouds for each cluster



Source: Own elaboration

Figure 10 shows five different groups classified and identified with different titles based on their content. This algorithm can select the cluster of each individual phrase in order to appropriately assign a sentiment to each section of the annual report. The five different clusters are as follows:

Balance sheet & financial statement (Cluster 1) Shows the financial activity of the bank for a specific period of time, in this case the year in question. It includes the income statement, balance sheet, and statement of cash flows.

Leverage & risk management (Cluster 2) The bank's annual report is required to disclose information about its leverage and risk management practices. This information helps investors and creditors assess the bank's financial health and its ability to manage its financial risks. The annual report must include a description of the bank's risk management policies and procedures, as well as information about the bank's exposure to different risks (market risk, credit risk, and even environmental risks).

Letters from management & bank outlook (Cluster 3) The Letters from management section of an annual report typically contains a letter from the bank's CEO or other senior executives, discussing the bank's overall performance during the year and outlining its plans and outlook for the future.

Strategy & non-Financial reports (Cluster 4) Shows the bank's performance in non-financial metrics, including environmental measures, social & equality measures, etc.

Social capital, shares & issued capital (Cluster 5) Includes information on shareholders, the number of shares outstanding, and the bank's issued capital.

Using these new found clusters, financial sentiment can be extracted from each of these sections individually, to be fed as input into our M&A prediction model.

3.5 Clustering with Transformers

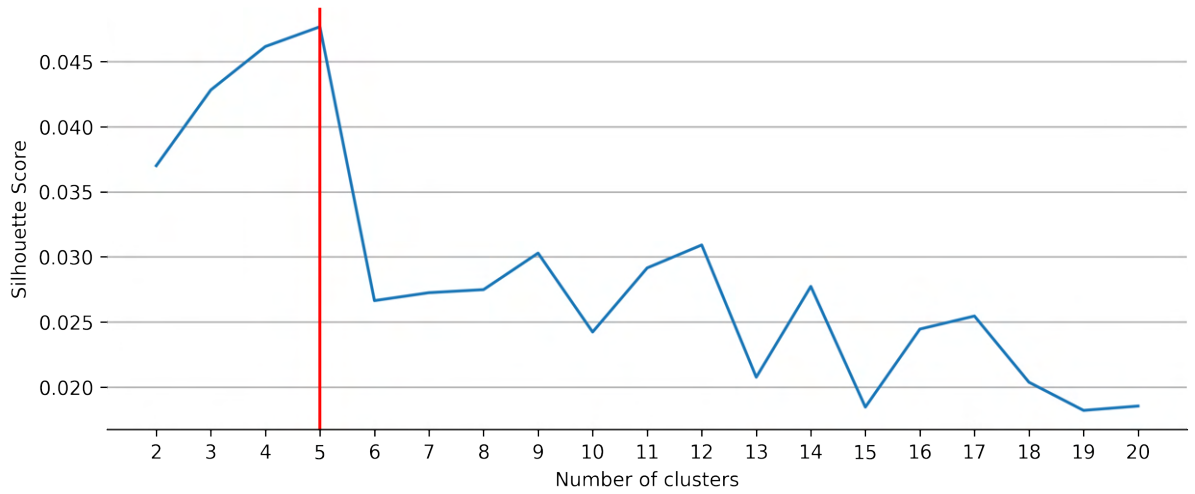
This section details the use of a $RoBERTa_{BASE}$ transformer in order to generate the embeddings of each sentence before applying the kmeans algorithm. In essence, the exact same procedure is followed as in section 3.4, but the traditional Word2Vec model is substituted by the embeddings 2.1.3.4 of $RoBERTa_{BASE}$. $RoBERTa_{BASE}$ has a total of 123 million parameters, and its execution without a CPU would take around 12 hours for the entire data set. For the generation of the embeddings, a specialised Google-Colab GPU was used.

The resulting embedding dataset of all 1.2 million sentences is over 4Gb in size. Loading such a file on to RAM memory is an undertaking, so training these clustering algorithms has been done on Google-Colab machines as well.

3.5.1 Optimal number of clusters

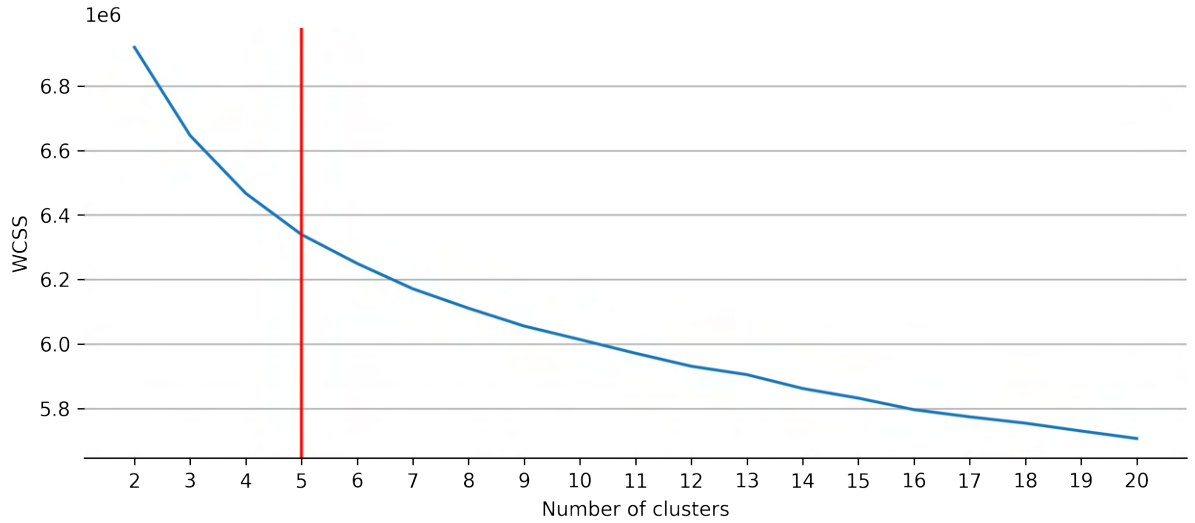
In an identical fashion to section 3.4.3, iterating over different numbers of clusters, models are trained and compared by their WCSS and Silhouette scores, shown in figures 11 and 12.

Figure 11: Silhouette scores for transformer embeddings clusters



Source: Own elaboration

Figure 12: WCSS distance for transformer embeddings clusters (Elbow method)



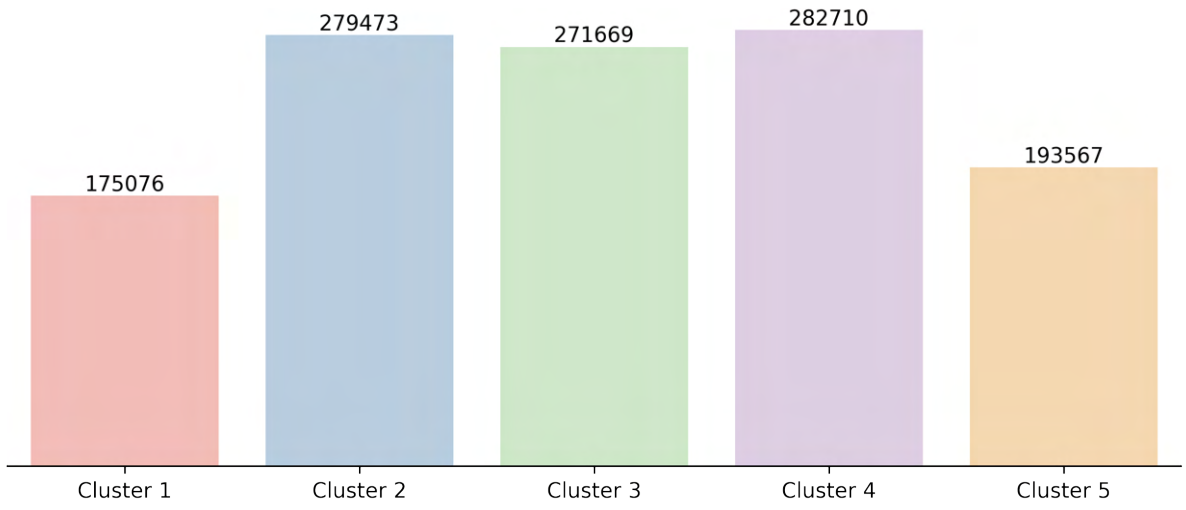
Source: Own elaboration

Similar to the data from section 3.4, the Silhouette scores show a clear number of clusters with 5 having the highest score, whereas the Elbow method gives little information about what point is the start of a decrease in the WCSS distance. In order to be consistent with previous results and compare the outputs, the number of clusters will be kept at 5.

3.5.2 Clustering results

Mimicking the procedure detailed in section 3.4.4, this section will detail the clustering results after using sentence embeddings from RoBERTa. Sentences fall into the different clusters following the distribution shown in figure 13.

Figure 13: Sizes of sentence clusters from transformer embeddings



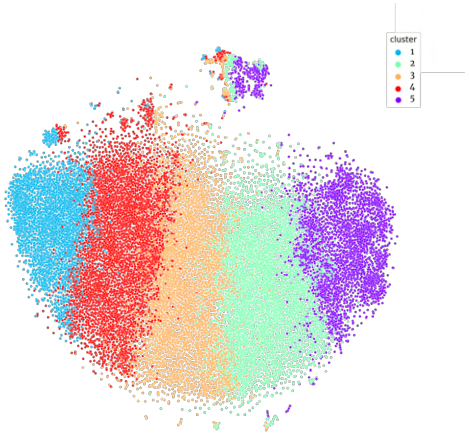
Source: Own elaboration

It is important to note that these 5 new clusters differ in size and content from the original 5 clusters presented in section 3.4.

3.5.2.1 Plotting results with t-SNE

Using the t-distributed stochastic neighbor embedding model, the results can be fitted into two dimensions in order to plot the data, and observe the clusters:

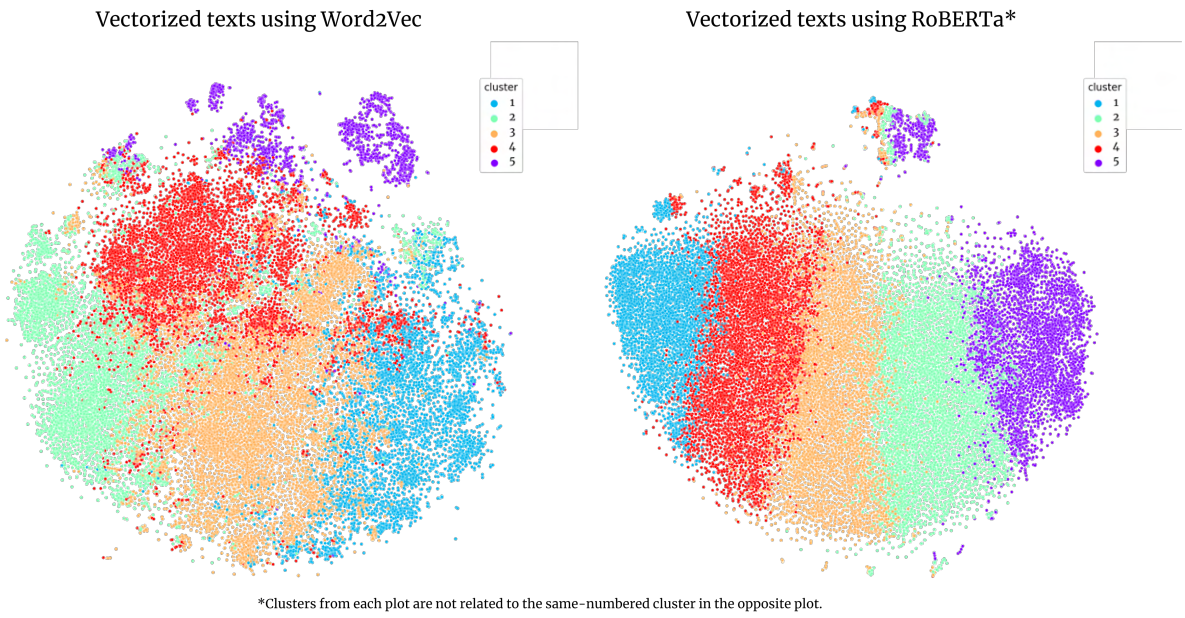
Figure 14: Embeddings clusters plotted using t-SNE



Source: Own elaboration

These results seem to have a much more uniform distribution, wit cluster borders being well defined. It may seem like these results are better than the Word2vec results, but such a uniform distribution may indicate that no major differences could be extracted from each of the clusters. Figure 15 shows the comparison between results of both methods.

Figure 15: Clustering comparison between Word2Vec and RoBERTa using t-SNE



Source: Own elaboration

Surprisingly the Word2Vec model seems to render better results. After analyzing the different sentences inside the clusters generated with RoBERTa, there didn't seem to be much difference between the individual clusters. Taking into account that RoBERTa requires a GPU and takes exponentially longer to process than the Word2Vec model, for the remainder of this study the clustering from the first method (Word2Vec) will be used. Further studies could implement a Transformer that has been previously trained on a financial dataset, in order to detect specific patterns in the data.

The $RoBERTa_{BASE}$ model has not been trained to detect different segments of annual reports, or to detect different parts of a financial report, so it's poor results are to be expected. Training such a model further falls out of the scope of this study, especially with such promising results from the Word2Vec vectorizing method.

3.6 Correlation between sentiment and M&As

This section details the findings on possible correlations between financial sentiment and resulting mergers and acquisitions.

3.6.1 Preparing the sentiment data

Sections are automatically divided by making use of the clustering model, detailed in section 3.4. Two metrics are computed: 'positivity', represented by ρ and 'negativity', represented by the letter η are measures of the overall financial sentiment in the phrases inside each cluster. For example, ρ=0.07 would indicate that 7% of phrases display positive sentiment within the given cluster. An example of the model input data is shown in table 5.

Table 5: Example of Annual Report computed sentiment

Bank	**Year**		ρ	η
BANKIA	2020	Cluster 1	0.089355	0.029282
		Cluster 2	0.020489	0.004674
		Cluster 3	0.057648	0.026072
		Cluster 4	0.114990	0.020933
		Cluster 5	0.000053	0.000069

Source: Own elaboration

Using accumulated data from 28 banks, totalling over 220 annual reports, all the data is transformed into the format shown in table 5.

3.6.2 Oversampling the training data

The training section of the data set has a total of 8 M&As, and 147 instances of non-mergers. In order to give the model a more balanced training set, Synthetic Minority Oversampling TEchnique (SMOTE) (Chawla et al., 2002) will be used. This technique is used to generate synthetic data points for the minority class in order to balance the dataset so that there is a balanced number of data points for each class.

The SMOTE algorithm works by creating synthetic minority samples. To do this, it selects a minority example at random and then finds the k-nearest neighbors of that example. From those k-nearest neighbors, it randomly chooses one and creates a new

synthetic minority example that is a combination of the two. This process is repeated until there are enough synthetic minority examples to balance the dataset.

The advantage of SMOTE is that it can create a more balanced dataset without having to remove any of the original data. This is especially helpful when the original dataset is small and removing data would result in a loss of information. The disadvantages of SMOTE are that it can sometimes overfit the data and it can be computationally expensive.

3.6.3 Logistic Regression

The data detailed in sections 3.6.1 and 3.6.2 is used to train a logistic regression model (70% for training, 30% for validation). The regression model is shown the sentiment of each section of an annual report.

It is important to note that the logistic regression algorithm fits a linear regression model on the log odd (or logit) of the result, using the following equation:

$$\log(\frac{p(x)}{1-p(x)}) = \beta_0 + \Sigma_{k=1}^{5}(\beta_{\rho_k} \cdot \rho_k + \beta_{\eta_k} \cdot \eta_k)$$

Where k is the cluster number, ρ is a measure of positive sentiment and η is a measure of negative sentiment.

The log odd is the logarithm of the odds, which is the probability of an event happening divided by the probability of it not happening. The output in logistic regression is the probability that an event will happen, which is calculated using the logistic function. The logistic function takes the log odds as its input and outputs a value between 0 and 1. The output event being fitted is the possibility of a merger happening on that given year.

A total of eleven β regression coefficients have to be fitted, one for each measure of ρ and η, plus the intercept β_0. The training renders the following regression coefficients:

Table 6: Regression coefficients for each cluster sentiment

Cluster	ρ	η
Balance sheet & financial statement (1)	-1.1979	0.5118
Leverage & risk management (2)	-1.2456	0.5379
Letters from management & bank outlook (3)	0.1104	0.0188
Strategy & non-Financial reports (4)	-0.0868	-0.1740
Social capital, shares & issued capital (5)	-0.9320	0.8187

Source: Own elaboration

Interpreting the regression of the log odds, a positive regression coefficient indicates that as the value of the explanatory variable increases, the probability of the dependent variable also increases. A negative regression coefficient indicates that as the value of the explanatory variable increases, the probability of the dependent variable decreases.

$$y^* = \log(\tfrac{p(x)}{1-p(x)}) = \beta_0 + \Sigma_{k=1}^{5}(\beta_{\rho_k} \cdot \rho_k + \beta_{\eta_k} \cdot \eta_k)$$

$$y = p(x) = \frac{e^{\beta_0 + \sum_{k=1}^{5}(\beta_{\rho_k} \cdot \rho_k + \beta_{\eta_k} \cdot \eta_k)}}{e^{\beta_0 + \sum_{k=1}^{5}(\beta_{\rho_k} \cdot \rho_k + \beta_{\eta_k} \cdot \eta_k)} + 1}$$

Then the relationship between y^* and y is:

$$y = \frac{e^{y*}}{e^{y*}+1}$$

The latter equation proves that positive β coefficients not only indicate a positive correlation of the log odds y^*, but of the output variable y as well. The same can be asserted in reverse about negative coefficients indicating negative correlation.

The results in table 6 clearly show on average a positive relationship between negative sentiment and the bank participating in an M&A, the opposite can be said for positive sentiment. Table 6 shows, for example, that a high average ρ in the balance sheet section has an inverse relationship with the bank being acquired.

Certain clusters show a much higher regression coefficient, indicating that the sentiment present in those groups has a bigger impact on the outcome than others. For instance, cluster 3, which contains sentences from the "Letters from management" and "Bank outlook" have little significance in the overall performance of the bank, whereas a high ρ in the "Leverage and risk management" section may indicate a high likelihood of the bank not being acquired or part of a merger.

3.6.3.1 Model characteristics

The complete details on the regression coefficients are detailed in table 7.

Table 7: Logistic regression coefficients

		coef	std err	z	$P > \|z\|$	[0.025	0.975]
ρ	Cluster 1	-1.1979	0.213	-5.620	0.000	-1.616	-0.780
	Cluster 2	-1.2456	0.278	-4.476	0.000	-1.791	-0.700
	Cluster 3	0.1104	0.155	0.711	0.477	-0.194	0.415
	Cluster 4	-0.0868	0.216	-0.402	0.688	-0.510	0.336
	Cluster 5	-0.9320	0.301	-3.094	0.002	-1.522	-0.342
η	Cluster 1	0.5118	0.219	2.341	0.019	0.083	0.940
	Cluster 2	0.5379	0.213	2.528	0.011	0.121	0.955
	Cluster 3	0.0188	0.220	0.085	0.932	-0.412	0.450
	Cluster 4	-0.1740	0.199	-0.872	0.383	-0.565	0.217
	Cluster 5	0.8187	0.271	3.024	0.002	0.288	1.349

Source: Own elaboration

$$\text{p-value} = 6.570 \cdot 10^{-30}$$

$$\text{Pseudo } R^2 = 0.2739$$

The p-value computed represents the probability of observing the test statistic assuming the null hypothesis H_0 and rejecting the alternative hypothesis H_A:

$$H_0 : \hat{\beta}_n = 0 \; \forall \; n \in (1, 11)$$
$$H_A : \hat{\beta}_n \neq 0 \; \forall \; n \in (1, 11)$$

Where $\hat{\beta}_n$ represents each regression coefficient, including the intercept. The null hypothesis states that all coefficients in the model are equal to zero, meaning that none of the predictor variables would have statistical significance. On the other hand, the alternative hypothesis states that not all coefficients are simultaneously zero, providing proof that some if not all coefficients have statistical significance.

The estimates of the coefficients and intercept in logistic regression are found with the maximum-likelihood estimation (MLE). These estimates are denoted as $\hat{\beta}_n$. The parameter of interest with which our estimates are contrasted is $\hat{\beta}_0$, which in this case is 0, because the aim is to check whether the estimated coefficients differ from 0. Following asymptotic theory, the difference between each coefficient $\hat{\beta}$ and $\hat{\beta}_0$ will approximately follow a normal distribution with mean 0 (Wasserman, 2004). Dividing the normal distribution with mean 0 by its standard deviation, the resulting distribution should follow

a normal distribution with mean 0 and a standard deviation equal to 1. This is known as the Wald statistic, also known as the Wald Chi-Squared test:

$$\mathcal{W} = \frac{(\hat{\beta} - \beta_0)}{\hat{se}(\hat{\beta})} \sim \mathcal{N}(0, 1)$$

Because the parameter of interest $\hat{\beta}_0$ is 0, the expression can be simplified:

$$\mathcal{W} = \frac{\hat{\beta}}{\hat{se}(\hat{\beta})} \sim \mathcal{N}(0, 1)$$

Which is equivalent to the following squared expression:

$$\mathcal{W}^2 = \frac{\hat{\beta}^2}{\hat{Var}(\hat{\beta})} \sim \mathcal{X}_1^2$$

The square of a normal distribution is equal to the Chi-Squared distribution with one degree of freedom.

Following the Wald test, a p-value is obtained using the Chi-Squared test statistic. Given that the overall p-value is extremely low (below 0.05), this presents strong evidence that the null hypothesis is invalid, which indicates that certain predictor variables have significant statistical significance.

Certain coefficients are much more important to the model than others. If we look at the column representing $P > |z|$, which indicates the p-value for the null hypothesis of each of the individual coefficients, we can observe 4 coefficients that have a p-value below 0.05, these are ρ_1, ρ_2, ρ_5 and η_5.

Observing these metrics, certain conclusions can be made. The logistic regression model has a very low p-value, indicating that the model is significant. These results, especially the low p-value are very favorable, and confirm our suspicions of a positive correlation between negative financial sentiment and an eventual M&A.

The Pseudo R^2 represents McFadden's $\tilde{R}^2$ approximation of R^2 for Logistic regression models (McFadden, 1974) where a higher $\tilde{R}^2$ represents a better fit of the model. It is represented by the following expression:

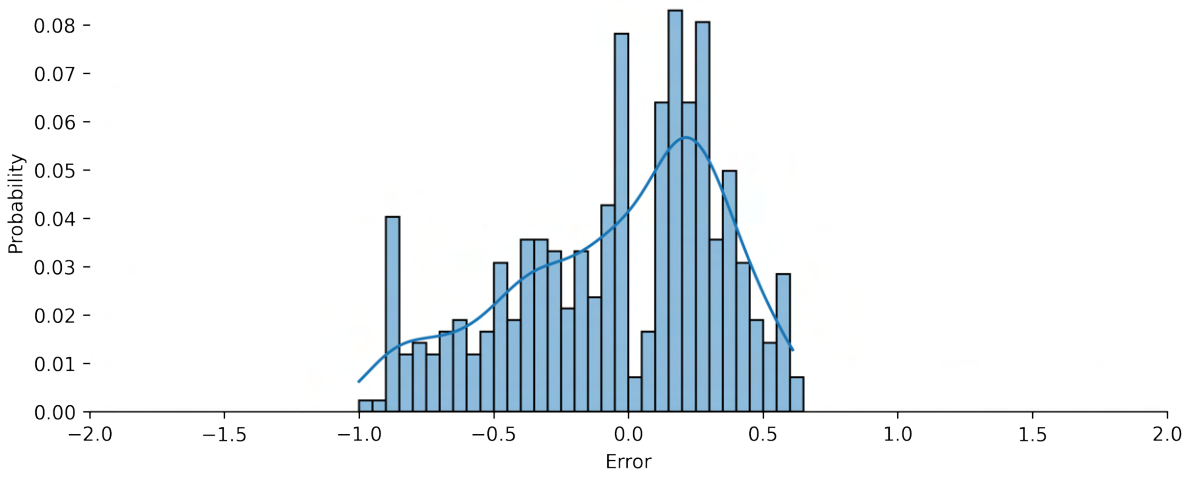
$$\tilde{R}^2 = 1 - \frac{\ln \hat{L}(M_{Full})}{\ln \hat{L}(M_{Null})}$$

Where $\hat{L}(M_{Full})$ represents the likelihood of the fitted model and $\hat{L}(M_{Null})$ represents the likelihood of the null model. In McFadden's words, when interpreting the results of this

metric, it is stated that a $\tilde{R}^2$ between 0.2 - 0.4 represents an excellent fit.

The model seems to have a high unexplained variance, this is to be expected. We are trying to predict a merger or acquisition based on a single set of variables of financial sentiment. The unexplained variance can be corroborated by observing the residues (errors) of the model on the training set. The Central Limit Theorem tells us that the distribution of the sum of a large number of random variables will tend towards a normal distribution, regardless of the distribution of the individual variables. So even if the individual residues are not Normally distributed, the sum of all the residues should be. This means that if the model itself is a good approximation of the true underlying relationships, the residues will tend to be Normally distributed.

Figure 16: Distribution of logistic regression training residues



Source: Own elaboration

$$\mu = -0.0417 \neq 0,\ \sigma = 0.3929$$

The results in figure 16 show that residues have small resemblance to a Normal distribution, in fact it might even resemble a bi-modal distribution. The residues still have underlying patterns that could be extracted with further data or features. Observing these results there is clear evidence that further knowledge should be extracted to fully explain the output, but this does not negate the clear relationship between the sentiment of certain clusters and the likelihood of an M&A taking place.

3.6.3.2 Validating the model

Table 8: Model classification report - (Validation data)

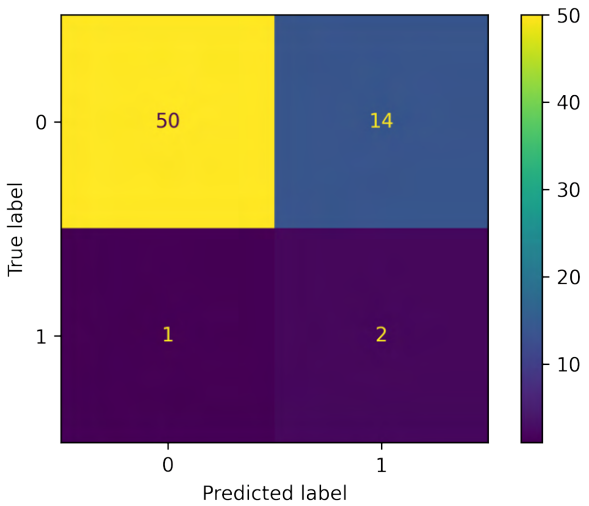
	precision	**recall**	**f1-score**	**support**
No M&A	0.98	0.77	0.86	64
M&A	0.12	0.67	0.20	3
accuracy			0.76	67
macro avg	0.55	0.72	0.53	67
weighted avg	0.94	0.76	0.83	67

Source: Own elaboration

The model presents an accuracy of 76%. Scoring a high weighted score when taking into account the small amount of mergers in the validation data (a total of 3). The model managed to predict 2 out of the 3 mergers correctly. Where we can see the model slightly fails is in false positives (15 non-merger reports where classified as mergers), bringing down the precision metric substantially.

The previous results can be further explored through the confusion matrix shown in Fig.17:

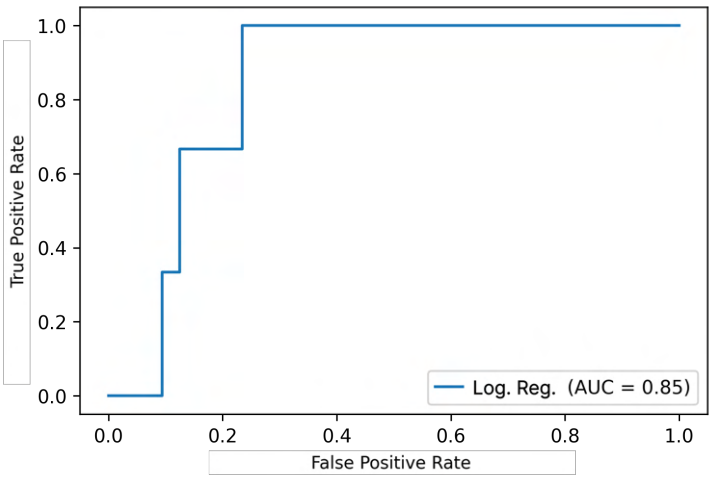
Figure 17: Confusion matrix of validation set



Source: Own elaboration

The predictor model also renders the following ROC Curve shown in Fig.18, with an area under curve (AUC) of 0.85.

Figure 18: ROC Curve of model on validation set



Source: Own elaboration

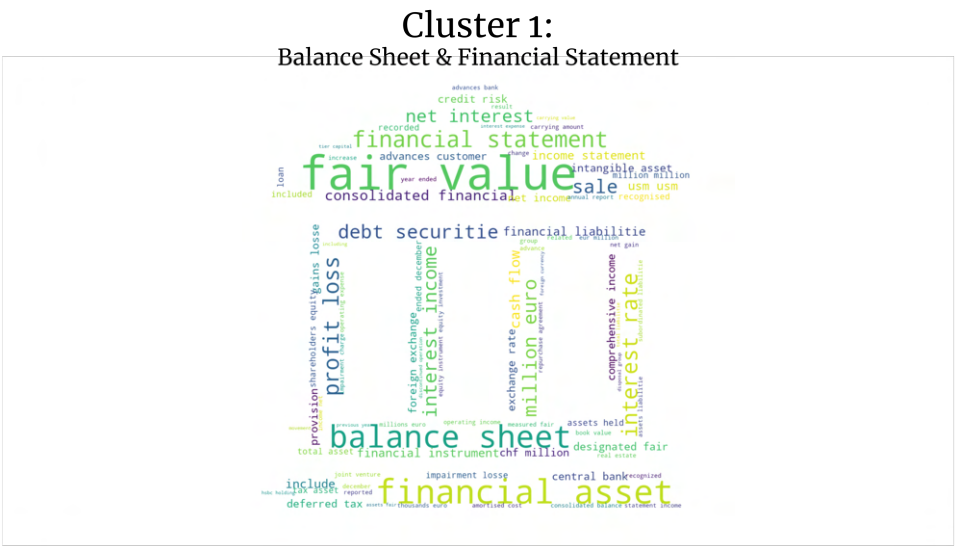
4 Results & Conclusions

The findings suggest that negative sentiment in annual reports is associated with the bank being acquired or merged. This suggests that banks with a higher percentage of negative phrases in their annual reports are more likely to be acquired by or merged with another bank. This may be due to the fact that banks that are more negative in their annual reports is directly related to the bank's performance, and is therefore perceived as being weaker and more vulnerable to takeover.

4.1 Classifying the segments of an Annual Report

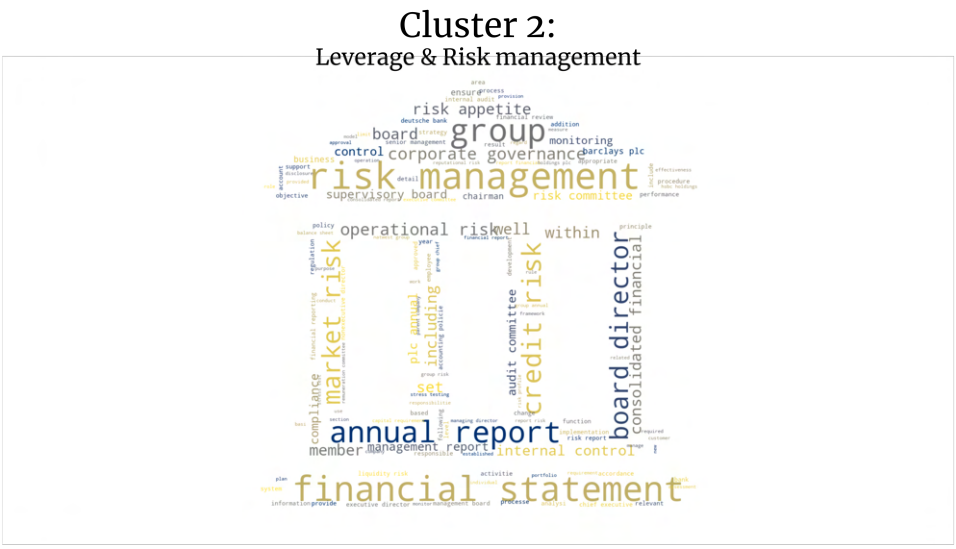
In order to investigate this relationship further, the data set of 1.2 million sentences was processed through a vectorizer (section 2.1.3.4) and clustering algorithm (section 3.4). This clustering separated sentences into 5 individual groups, which were inspected and cross referenced into different sections of an Annual Report, resulting in the following clusters:

Figure 19: Wordcloud of cluster number 1



Source: Own elaboration

Figure 20: Wordcloud of cluster number 2

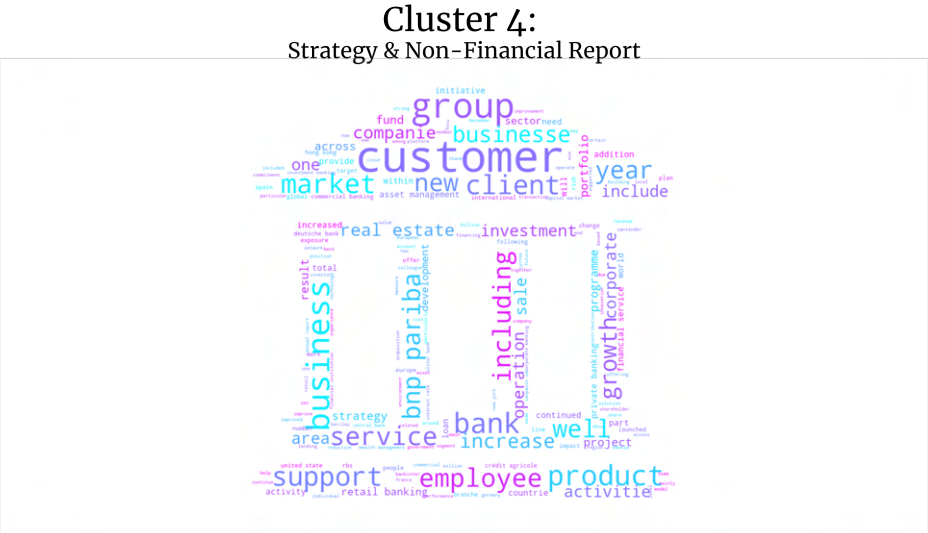


Source: Own elaboration

Figure 21: Wordcloud of cluster number 3

Source: Own elaboration

Figure 22: Wordcloud of cluster number 4



Source: Own elaboration

Figure 23: Wordcloud of cluster number 5

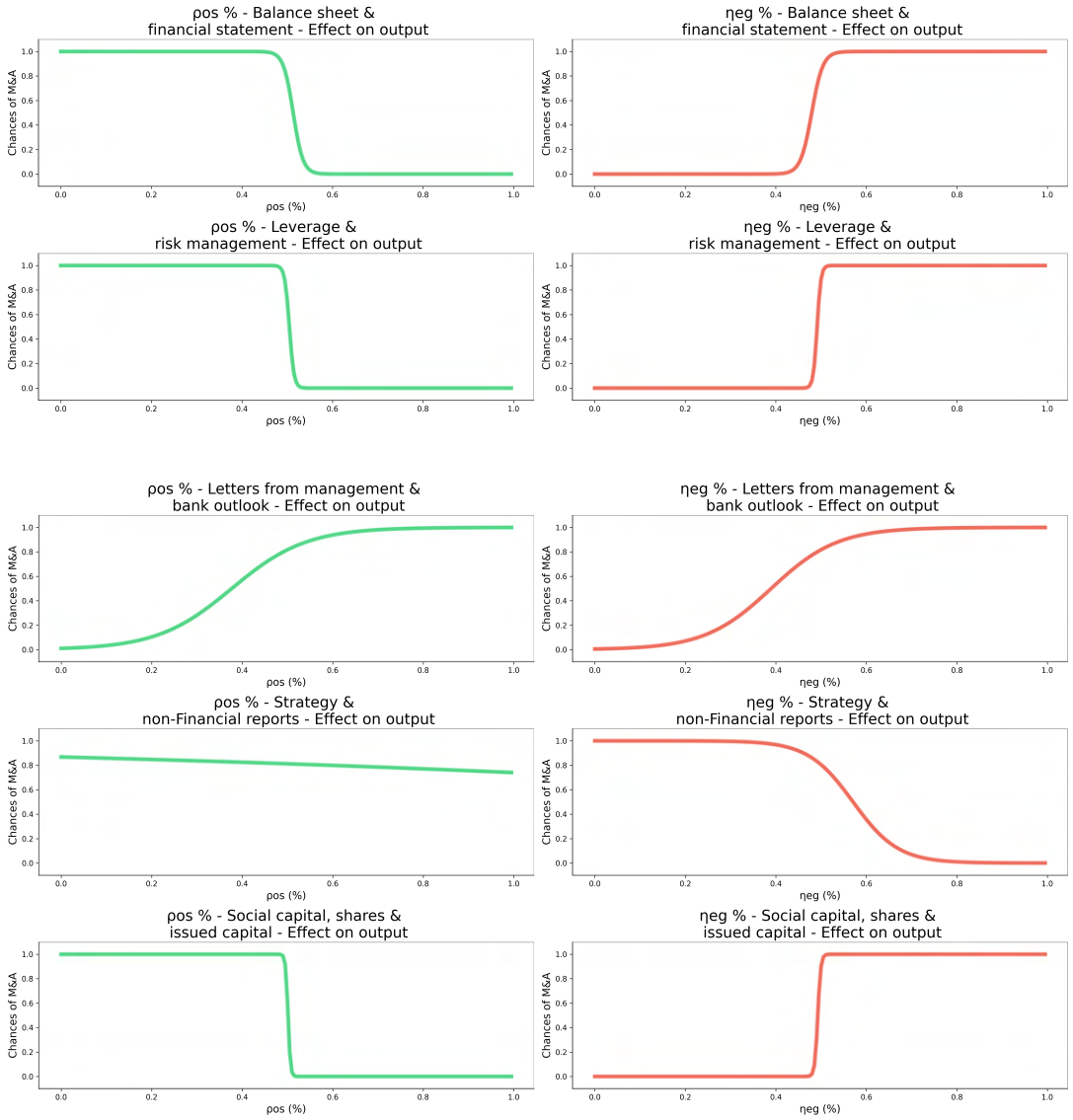
Source: Own elaboration

Once the clusters have been identified, sentiment can be extracted from each one of the individual segments of the annual report. It is to be expected that certain segments have a bigger impact on the bank's performance than others.

4.2 Correlation between sentiment and M&As

It is interesting to analyse the different effects each cluster has on the outcome. Figure 24 shows the effect each variable has on the output variable of the trained regression model, while all other variables are maintained at their average level.

Figure 24: Effect of each variable on the predicted M&A - Logistic regression



Source: Own elaboration

The most significant sections[†] of the annual report for predicting M&A likelihood are the financial statement, risk management and social capital sections, which show clear signs of correlation with the output. These sections all have a strong correlation with the output. An increase in the percentage of positive sentences (ρ) translates into a decrease in the likelihood of a merger or acquisition taking place. On the other hand, an increase in the percentage of negative sentences (η) translates into an increase in the likelihood of a merger or acquisition taking place.

It is also interesting to observe how the non-financial reports section has little correlation, in fact, it has an inverse correlation. This section is dedicated to reporting the bank's progress on social issues, climate change, and is very subjective. It seems that a very positive non-financial report may indicate that the bank's performance is below par and therefore has higher chances of being part of an M&A. A similar conclusion can be extracted from the letters from management section. It seems like there is a small opposite correlation, implying that a positive sentiment in the bank outlook section can increase the likelihood of an M&A occurring.

There could be alternative explanations to this phenomenon shown in both non-financial clusters. As explained in section 3.3, the model used to obtain these sentiments ("`distil roberta-finetuned-financial-news-sentiment-analysis`") is specifically trained on a financial dataset (Malo et al., 2014), so it may well be that the model is not able to obtain significant information from that sort of text, meaning that the sentiment obtained from these sections could be close to random. In fact, there is a very recently published algorithm specific for NLP related to climate change, named `ClimateBERT` (Webersinke et al., 2021), and a specific climate phrase bank for which to train models: `Climatext` (Varini et al., 2020), which would make sense to apply to those sections (especially the non-financial report) in order to ascertain if this section has any correlation with the output.

It must also be mentioned as a conclusion that the study has found evidence that the data with which the model has been trained is incomplete, and therefore future lines of work (section 4.3) should strive to acquire and process larger amounts of data to further

[†]These sections have traditionally harboured the most relevant information to measure a company's success. This is an assessment from a strictly financial perspective. The non-relevant sections for this model have invaluable information for investors and governments, who increasingly expect more from companies from an ESG (Environmental, Social, and Governance) perspective.

corroborate this relationship between sentiment and M&As.

4.3 Future lines of work

The future lines of work stemming from this project can be divided into the following different sections.

4.3.1 Expanding the dataset

The set of inputs in the study can be expanded in both vertically and horizontally depending on the scope of future studies. Vertical expansion of data would entail adding similar data to expand the model's capabilities without the need of synthetic oversampling (3.6.2), further corroborating the coefficients of improving on current findings.

A horizontal expansion of the dataset would entail looking for data with a different scope in mind, this could be a a new geographical scope (Katsafados et al., 2021 originally focused on the U.S. Banking sector), or expanding the set to other sectors, not only the banking sector.

In any case of dataset expansion, processing times should be taken into account. Processing more than 220 annual reports totalling over 1.4 million sentences is a labor intensive task that has taken over 6 hours (detailed in section 3.3) for every execution run. In order to improve this processing time, new studies should look into optimizing this process or using a more powerful machine.

4.3.2 Different machine learning algorithms

Further research into predictions of M&A likelihood could focus on applying models of different nature to this classification problem. Having established a relationship between the output variable and the financial sentiment of the texts, more complex models could unearth stronger relationships or non-linear correlations in the data that provide an improvement in accuracy over the model presented in this study.

The main objective of the regression model presented is to reveal the underlying relationships between the given inputs and the output variable, not to create a robust classifier. For this reason, trying more complex algorithms for the classification problem are likely

to improve in accuracy compared to the baseline logistic regression model.

In addition to this, as mentioned in the correlation findings in section 4.2, it is suspected that the sentiment analysis model applied is not effective against certain sections of the dataset that have been separated through clustering. Specific climate models have been developed and trained on climate phrases, such as `ClimateBERT` (Webersinke et al., 2021), and specific climate phrase banks have been developed, such as the `Climatext` phrase bank (Varini et al., 2020), used to train new models. Future lines of research could try to apply these climate-based models to these non-financial portions of each report, and compare the correlation findings to the results shown throughout our study.

References

Alshawi, Hiyan (1994). *The Core Language Engine*. The MIT Press.

Andreeva Desislava, et al. (2019). "Euro area bank profitability: where can consolidation help?" *Financial Stability Review,* November, pp. 107–118. URL: https://www.ecb.europa.eu/pub/pdf/fsr/ecb.fsr201911~facad0251f.en.pdf.

Bhargava, Rupal, Yashvardhan Sharma, and Shubham Sharma (2016). "Sentiment Analysis for Mixed Script Indic Sentences". In: *2016 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2016*. DOI: 10.1109/ICACCI.2016.7732099.

Bird, Steven, Ewan Klein, and Edward Loper (2019). *Natural Language Processing with Python, Analyzing Text with the Natural Language Toolkit*. Online Updated version of the original paper book published (Ed. O'Reilly, 2009). URL: https://www.nltk.org/book/.

Boer, Martin and Andrés Portilla (2020). "Consolidation of the European Banking Industry: Obstacles and Policies". In: Fernández, F. (Ed.): *The Euro in 2020. A Yearbook on the European Monetary Union,* pp. 263-282, FEF (Fundación de Estudios Financieros).

Briscoe, Ted et al. (1987). "A Formalism and Environment for the Development of a Large Grammar of English." In: *Proceedings of the 10th international joint conference on Artificial intelligence*, pp. 703–708.

Buckley, Ross, Douglas Arner, and Janos Barberis (2016). "The Evolution of Fintech: A New Post-Crisis Paradigm?" *Georgetown Journal of International Law* 47, pp. 1271–1319. DOI: 10.2139/ssrn.2676553.

Calipha, Rachel, Shlomo Tarba, and David Brock (2010). "Mergers and acquisitions: A review of phases, motives, and success factors" *Advances in Mergers & Acquisitions,* 19, pp. 1–24. DOI: 10.1108/S1479-361X(2010)0000009004.

Chawla, Nitesh V. et al. (2002). "SMOTE: Synthetic Minority Over-sampling Technique" *Journal of Artificial Intelligence Research,* 16, pp. 321–357. DOI: 10.1613/jair.953.

Crespo, Miguel (2021). "European bank consolidation prospects: using clustering and other machine learning techniques to identify merger participants". Final Business Analytics Degree Thesis. Universidad Pontificia Comillas. Madrid.

Datta, Deepak and George Puia (1995). "Cross-border acquisitions: An examination of the influence of relatedness and cultural fit on shareholder value creation in US acquiring firms" *Management International Review,* 35, pp. 337–359.

Devlin, Jacob et al. (2018). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: Burstein, J. et al. (Eds). *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics:*

Human Language Technologies, NAACL-HLT. Association for Computational Linguistics, 2019, Vol.1. DOI: 10.18653/v1/n19-1423. URL: https://aclanthology.org/N19-1423.

EBA - European Banking Authority (2021). Implementing Basel III in Europe. URL: https://www.eba.europa.eu/regulation-and-policy/implementing-basel-iii-europe.

European Central Bank (2000). *Mergers And Acquisitions Involving The EE Banking Industry - Facts And Implications.* URL: https://www.ecb.europa.eu/pub/pdf/other/eubkmergersen.pdf.

Figueiras, Isabel et al. (2021). "Bank mergers and acquisitions in the euro area: drivers and implications for bank performance" *Financial Stability Review,* November, pp. 115–125. URL: https://www.ecb.europa.eu/pub/financial-stability/fsr/special/html/ecb.fsrart202111_02~33910adb15.en.html.

Green Jr., Bert. F. et al. (1961). "Baseball: an automatic question-answerer". In: *Papers presented at the western joint IRE-AIEE-ACM computer conference*, pp. 219–224. DOI: 10.1145/1460690.1460714.

Hendrix, Gary G. et al. (1978). "Developing a Natural Language Interface to Complex Data". In: *ACM Transactions on Database Systems (TODS)* 3, pp. 105–147. DOI: 10.1145/320251.320253.

Hutchins, William. J. (1986). *Machine Translation: Past, Present, Future.* Ellis Horwood.

Hutto, C. J. and Eric Gilbert (2014). "VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text". In: *Proceedings of the 8th International AAAI Conference on Web and Social Media* 8, pp. 216–225. URL: https://ojs.aaai.org/index.php/ICWSM/article/view/14550.

Katsafados, Apostolos G. et al. (2021). "Using textual analysis to identify merger participants: Evidence from the U.S. banking industry" *Finance Research Letters,* 101949. DOI: 10.1016/j.frl.2021.101949.

Khurana, Diksha et al. (2017). "Natural Language Processing: State of The Art, Current Trends and Challenges". arXiv:1708.05148v1. URL: https://arxiv.org/abs/1708.05148v1.

Liu, Yinhan et al. (2019). "RoBERTa: A Robustly Optimized BERT Pretraining Approach". arXiv:abs/1907.11692.

Maas, Andrew L. et al. (2011). "Learning Word Vectors for Sentiment Analysis". In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 142–150. URL: https://aclanthology.org/P11-1015.

Malo, Pekka et al. (2014). "Good debt or bad debt: Detecting semantic orientations in economic texts". *Journal of the Association for Information Science and Technology,* 65, pp. 782–796. DOI: 10.1002/asi.23062.

Mani, Inderjeet and Mark T. Maybury (Eds.) (2000). *Advances in Automatic Text Summarization*. The MIT Press.

Manning, Christopher D., Hinrich Schütze, and Gerhard Weikurn (1999). *Foundations of Statistical Natural Language Processing*. The MIT Press.

McCallum, Andres and Kamal Nigam (1998). "A Comparison of Event Models for Naive Bayes Text Classification". AAAI Technical Report WS-98-05.

McFadden, Daniel (1974). "Conditional logit analysis of qualitative choice behavior". In: Zarembka, P. (Ed.), *Frontiers in Econometrics.* Academic Press, pp. 105–142.

McKinney and the Pandas Development Team (2022). *Pandas: powerful Python data analysis toolkit*. URL: https://pandas.pydata.org/docs/pandas.pdf.

Mikolov, Tomas et al. (2013). "Efficient Estimation of Word Representations in Vector Space". In: *Proceedings of Workshop at ICLR - 1st International Conference on Learning Representations*.

Norrestad, F. (2021). *Leading fintech unicorns in Europe as of August 2021, by market valuation*. URL: https://www.statista.com/statistics/1262336/leading-european-fintech-unicorns/.

Řehůřek, Radim et al. (2022). *Gensim: Topic modelling for humans*. URL: https://radimrehurek.com/gensim/auto_examples/index.html#documentation.

Ritter, Alan et al. (2011). "Named Entity Recognition in Tweets: An Experimental Study". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing.* Association for Computational Linguistics, pp. 1524–1534.

Routledge, Bryan R., Stefano Sacchetto, and Noah A. Smith (2013). "Predicting Merger Targets and Acquirers from Text". Working Paper, Carnegie Mellon University. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.691.160.

Sanh, Victor et al. (2019). "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter". arXiv:1910.01108.

Schueffel, Patrick (2016). "Taming the Beast: A Scientific Definition of Fintech" *Journal of Innovation Management,* 4, pp. 32–54. DOI: 10.2139/ssrn.3097312.

Scikit-learn developers (2022). *scikit-learn: Machine Learning in python*. URL: https://scikit-learn.org/stable/modules/classes.html.

Sun, Chi et al. (2019). "How to Fine-Tune BERT for Text Classification?" In: Sun, M. (Eds). *China National Conference on Chinese Computational Linguistics (CCL).* Springer International Publishing, pp. 194–206.

The NumPy community (2022). *Numpy Library Documentation.* URL: https://numpy.org/doc/stable/.

Van der Maaten, Laurens and Geoffrey Hinton (2008). "Viualizing data using t-SNE" *Journal of Machine Learning Research,* 9, pp. 2579–2605.

Varini, Francesco S. et al. (2020). "ClimaText: A Dataset for Climate Change Topic Detection". In: Tackling Climate Change with Machine Learning workshop at NeurIPS 2020. arXiv:2012.00483. DOI: 10.48550/arXiv.2012.00483.

Venuti, Keenan (2021). "Predicting Mergers and Acquisitions using Graph-based Deep Learning". In: *CoRR.* URL: https://arxiv.org/abs/2104.01757.

Wasserman, Larry (2004). "All of Statistics: A Concise Course in Statistical Inference". In: *The American Statistician.* ISSN: 0003-1305. DOI: 10.1198/tas.2005.s30.

Weber, Yaakov, Shlomo Tarba, and Arie Reichel (2011). "A Model of the Influence of Culture on Integration Approaches and International Mergers and Acquisitions Performance". *International Studies of Management & Organization,* 41, pp. 9–24. DOI: 10.2307/23032393.

Webersinke, Nicolas et al. (2021). "ClimateBert: A Pretrained Language Model for Climate-Related Text". *arXiv:2110.12010.* DOI: 10.48550/arXiv.2110.12010.

Woods, W.A. (1978). *Semantics and Quantification in Natural Language Question Answering.* Academic Press. DOI: 10.1016/S0065-2458(08)60390-3.

Yi, J. et al. (2003). "Sentiment analyzer: extracting sentiments about a given topic using natural language processing techniques". In: Proceedings - IEEE International Conference on Data Mining, ICDM, pp. 427–434. DOI: 10.1109/ICDM.2003.1250949.

Attachments

Table 9: Python 3.8 Libraries used during development

Library	**Version**	**Homepage**
fastparquet	0.8.1	fastparquet.readthedocs.io
gensim	4.1.2	radimrehurek.com/gensim
googletrans	4.0.0rc1	github.com/ssut/py-googletrans
h5py	3.6.0	h5py.org
joblib	1.1.0	joblib.readthedocs.io
nltk	3.7	nltk.org
numpy	1.22.3	numpy.org
pandas	1.4.1	pandas.org
pypdf2	1.26.0	pypdf2.readthedocs.io
scikit-learn	1.0.2	scikit-learn.org
spacy	3.2.3	spacy.io
tensorflow	2.8.0	tensorflow.org
torch	1.11.0	pytorch.org
transformers	4.17.0	github.com/huggingface/transformers

Source: Own elaboration

Listing 1: Preprocessing functions script

```
# --- Preprocessing functions ---
# Intake PDFs and produce the resulting dataset

import os
import time
import numpy as np
import pandas as pd
import PyPDF2
import torch
import googletrans as gt
from tqdm import tqdm

def read_raw_pdf(path, verbose=0):
    # Get raw PDF file path and extract text using the PyPDF2 library
    import PyPDF2
    try:
        pdfReader = PyPDF2.PdfFileReader(path)
        num_pages = pdfReader.numPages
    except:
        if verbose > 1:
            print(f'Error de lectura de PDF en: {path}')
        raise
    count = 0
    pages = []
    if verbose > 1: print(f'\n{path.split("/")[-1]}')
    while count < num_pages:
        if verbose > 1: print(f'Page {count+1}/{num_pages}', end="\r")
        pageObj = pdfReader.getPage(count)
        pages.append(pageObj.extractText())
        count += 1

    pdf_dict = {
        'path':path,
        'pages':pages,
    }
    return pdf_dict

def prepare_PDF_dataframe(raw_texts):
    # Process and transform the dataframe containing the RAW PDF
    import pandas as pd
    data = pd.DataFrame(raw_texts)
    data['bank'] = data['path'].apply(lambda x: x.split('/')[3])
    data['year'] = data['path'].apply(lambda x: x[-8:-4])
    data = data.explode('pages').reset_index().reset_index()
    data['page'] = data.groupby('index').rank('max').rename(columns={'
    level_0':'page'})['page']
    data['page'] = data['page'].astype('int')
    data = (data.drop(columns=['level_0','index','path'])
                .set_index(['bank','year','page'])
                .rename(columns={'pages':'text'})
           )
    return data

def clean_text(text, esp=False):
```

```
    # Clean text through regex queries
    import re
    text = re.sub(r"\n\n", ".", text)
    text = re.sub(r"\n", " ", text)
    text = re.sub(":", " ", text)
    text = re.sub(r"[^a-zA-Z.\s:ñÑáéíóúÁÉÍÓÚ]", "", text)
    text = re.sub(r' +', ' ', text)
    text = text.strip()
    if text == '':
        return None
    if esp:
        try:
            text=translate_ES(text)
        except:
            return None
    # Tokenize input text
    return list(filter(None, text.split('.')))

def tokenize_phrases(df_data, esp=False):
    # Get PDF data, clean and separate into sentences.
    tqdm.pandas()
    if esp: print('--- Realizando traducción al inglés...')
    df_data['sentences'] = df_data['text'].progress_apply(lambda x:
    clean_text(x, esp=esp))
    df_data = df_data.dropna()
    df_sentence = df_data.explode('sentences').dropna()
    df_sentence['sentence'] = df_sentence.groupby(df_sentence.index).
    rank('max')['sentences']
    df_sentence['sentence'] = df_sentence['sentence'].astype('int')
    df_sentence.set_index('sentence', append=True, inplace=True)
    df_sentence.drop(columns=['text'], inplace=True)
    return df_sentence

translator = gt.Translator()

def translate_ES(sentence):
    # Translates sentences into english
    return translator.translate(sentence, dest='en', src='es').text

def process_PDFs(list_of_paths,
                 timeit=True,
                 verbose=0,
                 esp=False,
                 save_path=None,
                 save_raw_text=None):
    # Process RAW PDFs
    import time
    total_times = dict()
    if timeit:
        print('Cronometrando lectura de PDFs...')
        start_time = time.time()

    raw_texts = []
    for path in list_of_paths:
        try:
```

```
107             pdf_dict = read_raw_pdf(path, verbose=verbose)
108         except:
109             continue
110         raw_texts.append(pdf_dict)
111
112     if timeit:
113         end_time = time.time()
114         total_times["lectura"] = end_time - start_time
115         if verbose: print(f'--- Tiempo de lectura (s): {end_time -
    start_time}')
116
117     # Textos a DataFrame:
118     if verbose: print(f'Preparando dataframe...')
119     raw_text_data = prepare_PDF_dataframe(raw_texts)
120
121     # Guardado de texto RAW:
122     if save_raw_text:
123         raw_text_data.to_parquet(save_raw_text)
124
125     if verbose: print(f'Tokenizando frases...')
126     df_phrases = tokenize_phrases(raw_text_data, esp=esp)
127
128     # Guardando dataset:
129     if save_path:
130         print('Guardando dataset...')
131         df_phrases.to_parquet(save_path)
132         print('Dataset guardado!')
133
134     return df_phrases, total_times
135
136 def filter_phrases_size(df_sentence, min_words=10, tensor_limit=512):
137     # Filters phrases to have a maximum tensor lenght of 512
138     df_sentence['length'] = df_sentence['sentences'].apply(lambda x:
    len(x.strip().split(' ')))
139     return df_sentence[(df_sentence.length >= min_words) & (df_sentence
    .length <= tensor_limit)]
140
141 def softmax(x):
142     # Compute softmax values for each sets of scores in x
143     e_x = np.exp(x - np.max(x, axis=1)[:, None])
144     return e_x / np.sum(e_x, axis=1)[:, None]
145
146 def sentimentAnalysis_full(text_payload, tokenizer, model, tensor_limit
    =512):
147     # Computes the sentiment results for a given phrase
148     inputs = tokenizer(text_payload, return_tensors="pt")["input_ids"
    ][:,:tensor_limit]
149     logits = model(inputs).logits
150     return softmax(np.array(logits.detach()))[0]
151
152 def get_phrase_sentiment(dataset, min_words=10, tensor_limit=512,
    save_sentiment_path=None, timeit=True):
153     # Returns sentiment for every phrase in the dataset
154     import time
155     import pandas as pd
```

```
156     from tqdm import tqdm
157     from transformers import AutoTokenizer,
    AutoModelForSequenceClassification
158     MODEL_PATH = "mrm8488/distilroberta-finetuned-financial-news-
    sentiment-analysis"
159     tqdm.pandas()
160     total_times = dict()
161 
162     tokenizer = AutoTokenizer.from_pretrained(MODEL_PATH)
163     model = AutoModelForSequenceClassification.from_pretrained(
    MODEL_PATH)
164 
165     if timeit:
166         print('Cronometrando procesado de sentiment de frases...')
167         start_sentiment_time = time.time()
168 
169     dataset = filter_phrases_size(dataset, min_words=min_words,
    tensor_limit=tensor_limit)
170     dataset['sentiment'] = dataset.sentences.progress_apply(
171         lambda x: sentimentAnalysis_full(x, tokenizer, model,
    tensor_limit=tensor_limit)
172     )
173 
174     if timeit:
175         end_sentiment_time = time.time()
176         total_times['sentiment'] = end_sentiment_time -
    start_sentiment_time
177 
178     dataset['negative'] = dataset['sentiment'].progress_apply(lambda x:
     x[0])
179     dataset['neutral']  = dataset['sentiment'].progress_apply(lambda x:
     x[1])
180     dataset['positive'] = dataset['sentiment'].progress_apply(lambda x:
     x[2])
181     df_export = dataset.drop(columns='sentiment')
182 
183     if save_sentiment_path:
184         df_export.to_parquet(save_sentiment_path)
185 
186     return df_export, total_times
187 
188 def main():
189     datos              = '../Data/RAW/'
190     raw_pages          = '../Data/PREP/raw_pages.parquet'
191     raw_sentences      = '../Data/PREP/sentences.parquet'
192     data_sentiment     = '../Data/PREP/sentiment.parquet'
193 
194     pdf_paths = []
195 
196     for b in bancos:
197         pdfs = os.listdir(datos+b)
198 
199         for pdf in pdfs:
200             pdf_paths.append(datos+b+'/'+pdf)
201 
```

```
202     # Processes PDFs and saves the resulting DataFrame:
203     data_en, total_times_en = process_PDFs(random.sample(pdf_paths,12),
        esp=False, verbose=2, save_path=raw_en_sentences)
204     # Returns sentiment data for each phrase in dataset and saves it to
        the corresponding path:
205     df_sentiment, times = get_phrase_sentiment(data_en, min_words=5,
      save_sentiment_path=data_sentiment)
206
207 if __name__ == "__main__":
208     main()
```

Listing 2: Logistic regression training script

```
# --- Training functions ---
# Intake dataset and train a logistic regression model
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
from tqdm import tqdm
from gensim.models import Word2Vec
from joblib import load, dump

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from imblearn.over_sampling import SMOTE

def refactor_data(data):
    # Prepares data into the correct shape for input
    return (data
            .reset_index()
            .rename(columns={'sentences':'phrase', 'page':'n_page', '
    sentence':'n_phrase'},)
            .set_index(['bank','year', 'n_page', 'n_phrase']))

def get_top_positive_phrases(data, count=20):
    # Returns a list of the top ranked positive phrases
    data = refactor_data(data)
    preview = data.sort_values(by='positive', ascending=False).head(
    count)[['phrase']]
    return preview

def get_top_negative_phrases(data, count=20):
    # Returns a list of the top ranked negative phrases
    data = refactor_data(data)
    preview = data.sort_values(by='negative', ascending=False).head(
    count)[['phrase']]
    return preview

def palabra_en_modelo(palabra,w2v):
    # Checks if word is in Word2Vec vocabulary:
    try:
        w2v.wv.get_vector(palabra)
        return True
    except:
        return False

def vectorize_text(texto_vector, w2v, SIZE_VECTORS=100):
    # Uses trained Word2Vec to create a numeric text representation:
    vectors = [w2v.wv.get_vector(i) for i in texto_vector if
    palabra_en_modelo(i,w2v)]
    return np.mean(vectors, axis=0) if len(vectors) else np.zeros(
    SIZE_VECTORS)

```

```
50 def prepare_dataset(data, path_word2vec, path_clustering, SIZE_VECTORS
   =100):
51     # Runs through all the preparation functions and divides phrases
   into clusters:
52     import numpy as np
53     import pandas as pd
54     from tqdm import tqdm
55     tqdm.pandas()
56     data = refactor_data(data)
57 
58     word2vec = Word2Vec.load(path_modelo_w2v)
59     print('Tokenizing...')
60     data['texto_vector'] = data.phrase.progress_apply(lambda x: [w.
   lower() for w in x.split(' ') if w])
61     print('Aplicando Word2Vec...')
62     vectors = data.texto_vector.progress_apply(lambda x: vectorize_text
   (x, word2vec, SIZE_VECTORS)).to_numpy()
63 
64     dfVectors = pd.DataFrame(np.concatenate(vectors).reshape(vectors.
   shape[0],SIZE_VECTORS,), index=data.index,
65                              columns=[f'WV{i}' for i in range(SIZE_VECTORS)
   ])
66     print('Aplicando Clustering...')
67     clustering = load(path_modelo_clustering)
68     clusters = clustering.predict(dfVectors)
69     data['cluster'] = clusters
70     data = data.dropna()
71     print('Preparando Dataset...')
72     ml_df = data.reset_index().groupby(['bank','year','cluster']).mean
   ()[['positive','negative']].unstack('cluster')
73     ml_df.columns = [sent[:3]+'_cluster_'+str(cluster)  for sent,
   cluster in ml_df.columns.values]
74     ml_df = ml_df.fillna(0.0)
75 
76     return ml_df
77 
78 def get_mergers(ml_df, path_merger_data):
79     # Reads mergers data for each bank each year from the CSV in the
   defined path
80     df_mergers = pd.read_csv(path_merger_data).dropna()
81     df_mergers['YEAR'] = df_mergers['YEAR'].astype('int')
82     df_mergers = df_mergers.set_index('YEAR')
83 
84     merger_target = []
85     for bank, year in ml_df.index:
86         merger_target.append(int(df_mergers[bank][int(year)]))
87 
88     return merger_target
89 
90 def predict_merger(data,
91                    path_modelo_word2vec,
92                    path_modelo_clustering,
93                    path_modelo_merger,
94                    timeit=True
95                   ):
```

```
    # Used to make predictions once the model has been trained
    import time
    from joblib import load
    total_time = 0
    if timeit:
        start = time.time()
    X = prepare_dataset(data, path_modelo_word2vec,
path_modelo_clustering)
    # Load del modelo:
    print('Cargando Modelo...')
    model = load(path_modelo_merger)
    print('Haciendo Predicción...')
    merger_pred = model.predict_proba(X)[:,1]
    if timeit:
        end = time.time()
        total_time = end - start

    return merger_pred, total_time

def main():
    path_sentiment_data    = "../Data/PREP/sentiment_full.parquet"
    path_merger_data       = "../Data/MERGERS/mergers.csv"
    path_modelo_w2v        = '../Data/MODELS/modelo_word2vec_100.model'
    path_modelo_clustering = '../Data/MODELS/clustering_pipeline.joblib
'

    X = prepare_dataset(data, path_modelo_w2v, path_modelo_clustering)
    y = get_mergers(X, path_merger_data)

    # Train-test split:
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
=0.3, random_state=40)
    # Synthetic minority oversampling (ONLY IN TRAINING):
    sm = SMOTE(random_state=42)
    X_train, y_train = sm.fit_resample(X, y)
    # Standard scaler:
    scaler = Pipeline(steps=[
        ('scaler', StandardScaler()),
    ])
    X_train_st = pd.DataFrame(scaler.fit_transform(X_train), columns=
X_train.columns)
    # Statsmodels logit model:
    logit_model=sm.Logit(y_train,X_train_st)
    result=logit_model.fit()
    print(result.summary())

    # Validation:
    X_test_st = scaler.transform(X_test)
    y_pred = np.where(result.predict(X_test_st)>0.5,1,0)
    y_true = np.array(y_test)

    # Classification report:
    from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score, classification_report
    print(classification_report(y_true, y_pred))
```

```
146
147     # Confusion matrix and AUC
148     from sklearn.metrics import ConfusionMatrixDisplay, RocCurveDisplay
149     fig, ax = plt.subplots(dpi=300)
150     ConfusionMatrixDisplay.from_predictions(y_true, y_pred, ax=ax)
151     plt.show()
152
153     fig, ax = plt.subplots(dpi=300)
154     RocCurveDisplay.from_predictions(y_true, y_pred, ax=ax)
155     plt.show()
156
157 if __name__ == "__main__":
158     main()
```

Listing 3: Word2Vec and K-Means training script

```
# --- Word2Vec and K-Means training ---
# Intake dataset and:
# -> Train a Word2Vec model using the Gensim API
# -> Train a K-Means clustering model using the Scikit-Learn API

import numpy as np
import pandas as pd

from tqdm import tqdm
from gensim.models import Word2Vec

from sklearn.pipeline import Pipeline
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

def main():
    data_sentiment         = '../Data/PREP/sentiment.parquet'
    path_modelo_w2v        = '../Data/MODELS/modelo_word2vec_100.model'
    path_modelo_clustering = '../Data/MODELS/clustering_pipeline.joblib'
    tqdm.pandas()

    data = pd.read_parquet(data_sent_full)
    data = (
        data
        .reset_index()
        .rename(columns={'sentences':'phrase', 'page':'n_page', 'sentence':'n_phrase'},)
        .set_index(['bank','year', 'n_page', 'n_phrase'])
    )

    data['texto_vector'] = data.phrase.progress_apply(lambda x: [w.lower() for w in x.split(' ') if w])

    # Word2vec parameters
    SIZE_VECTORS = 100
    WINDOW = 8
    EPOCHS = 5
    MIN_COUNT = 5

    w2v = Word2Vec(data.texto_vector,
                vector_size=SIZE_VECTORS,
                window=WINDOW,
                epochs=EPOCHS,
                min_count=MIN_COUNT, callbacks=[callback_w2v()])

    # Prepare dataset for clustering:
    dataset = pd.DataFrame(np.concatenate(vectors).reshape(vectors.shape[0],SIZE_VECTORS,), index=data.index,
                           columns=[f'WV{i}' for i in range(SIZE_VECTORS)])

```

```
    # Iterate over many n_clusters and get the measures
    # for silhouette score and elbow score
    sil_scores = []
    elbow_scores = []

    for i in n_clusters:
        print(f'Clustering with {i} clusters...')
        train_data = dataset.sample(10000, random_state=10)
        model = Pipeline(steps=[
            ('scaler', StandardScaler()),
            ('cluster', KMeans(n_clusters=i))
        ])
        cluster = model.fit_predict(train_data)
        sil_scores.append(silhouette_score(train_data,cluster))
        elbow_scores.append(model['cluster'].inertia_)

    # The plots for these measures are shown in the report,
    # but have not been included in this summary code

    # The optimum number of clusters is 5:
    model = Pipeline(steps=[
        ('scaler', StandardScaler()),
        ('cluster', KMeans(n_clusters=5))
    ])
    cluster = model.fit_predict(dataset)
    print(pd.Series(cluster).value_counts())

    # SAVE the models:
    w2v.save(path_modelo_w2v) # Word2Vec model
    dump(model, path_modelo_clustering) # K-Means model
```