SAPIENZA
UNIVERSITÀ DI ROMA

Authorship and Topic Analysis in Italian Central Bank Communication

Facoltà di Ingegneria dell'Informazione, Informatica e Statistica
Corso di Laurea Magistrale in Data Science

Iadisernia Giulia
ID number 2065450

Advisor
Prof. Barbarossa Sergio

Co-Advisors
Dr. Camassa Carolina
Dr. Coletta Andrea

Academic Year 2023/2024

Thesis defended on 28 October 2024
in front of a Board of Examiners composed by:

Prof. Brutti Pierpaolo (chairman)
Prof. Barbarossa Sergio
Prof. Galasso Fabio
Prof. Lembo Domenico
Prof. Maggino Filomena
Prof. Petti Manuela
Prof. Silvestri Fabrizio
Prof. Tieri Paolo

Authorship and Topic Analysis in Italian Central Bank Communication
Master's thesis. Sapienza University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: giulia.iadisernia1@gmail.com

I want to thank Prof. Barbarossa Sergio, my thesis Advisor, for kindly supporting me throughout the course of this thesis.
A heartfelt thanks to Dr. Camassa Carolina and Dr. Coletta Andrea, experts from the Applied Research Team (ART) in the IT directorate of the Bank of Italy, and my thesis Co-Advisors for their guidance and insightful suggestions that have strongly contributed to the realization of my thesis.
To Mom and Dad, for their moral support and invaluable lessons they've taught me. Without them, I wouldn't be the person I am today. Thank you.

Abstract

In this thesis, we apply machine learning and natural language processing (NLP) techniques to classify and analyze speeches delivered by the Governors of the Bank of Italy from 1990 to 2023. The study is structured around three main experiments. First, we classify the Governors based on the full text of their speeches, employing a combination of document embedding techniques such as TF-IDF and Doc2Vec, along with classification methods like Support Vector Machines (SVM), logistic regression, and fine-tuned Transformer-based models. The highest balanced accuracy of 91.01% is achieved by separately classifying English and Italian speeches, outperforming the models trained on the combined corpus. Second, we identify and analyze the underlying topics in the speeches using BERTopic, a state-of-the-art topic modeling technique. By running the algorithm on English and Italian documents separately, we detect a wide range of topics, including monetary policy, sustainable finance, cybersecurity, and the Covid-19 pandemic. We further concentrate on the evolution of the themes in the Governors' final consideration speeches by leveraging BERTopic's Dynamic Topic Modeling (DTM) functionality. Finally, we classify the Governors based on their distinctive writing styles by masking varying portions of content-related words identified through BERTopic. The highest balanced accuracy of 93.86% is again achieved by classifying English and Italian speeches separately, representing a 3% improvement over unmasked models. We find that masking content reduces the classifier's reliance on specific topics and encourages the models to focus on stylistic features, improving classification performance.

Keywords: Bank of Italy, BERTopic, classification, central bank communication, machine learning, natural language processing, topic analysis, writing style.

Contents

Chapter 1

Introduction and Related Work

Central banks need to communicate transparently with the general public. For this reason, text analysis of central bank communication has gained increasing attention in recent years, becoming a widely popular research subject. Numerous studies have investigated the effects of central bank communication on macroeconomic and financial variables. Specifically, much emphasis has been placed on understanding not only the topics central banks address (i.e., what they say) but also the tone and style of their communication (i.e., how they say it). For instance, Gebauer et al. (2024) show "how unexpected changes in communication influence growth and inflation" in the Eurosystem, and Bouscasse et al. (2024) state that "words matter as much as actions for central banks. Because changes in tone can presage shifts in monetary policy".

Today, one of the main goals of central banks is to inform and educate the general public to engage a broader audience in monetary policy decisions and related matters. For example, in 2022, the Bank of Italy launched a financial education portal called "L'economia per tutti"[1]. It aims to provide both young and adult audiences with basic knowledge of economics and finance, covering topics like investments, sustainable finance (ESG factors), inflation, cryptocurrencies, and artificial intelligence.

Several papers have applied natural language processing (NLP) methods to examine communications from central banks. For instance, Astuti et al. (2022) and Priola et al. (2021) employed *Latent Dirichlet Allocation* (LDA), as introduced by Blei et al. (2003), to uncover the latent topics in central bank speeches. These studies used LDA on *Bag-of-Words* (BoW) representations of the documents. While this embedding technique is straightforward to implement, it has several limitations.

In another study, Astuti et al. (2020) conducted a linguistic analysis of the final consideration speeches delivered by the Governors of the Bank of Italy from 1946 (starting with Luigi Einaudi) to 2018 (ending with Ignazio Visco). The objective was to observe the evolution of the topics and the Bank's communication style over the decades, exploring how the language reflects changes in the Italian and global economy. The authors visually presented the main topics of each decade using *wordclouds*[2], employing unigrams, bigrams, and trigrams after applying similar

[1] https://economiapertutti.bancaditalia.it.

[2] A wordcloud is a visual representation of a collection of words depicted in different sizes. The greater the word frequency, the bigger and bolder the word appears in the plot.

pre-processing steps to those used by Astuti et al. (2022) and Priola et al. (2021).

In a different approach, Baumgärtner and Zahner (2023) proposed using neural networks to generate high-quality textual representations specific to central bank communication. These representations were used to develop an index that monitors deviations in the Federal Reserve's inflation-targeting communication. Unlike the previous studies, which relied on BoW representations, Baumgärtner and Zahner employed more sophisticated techniques, including *Doc2Vec* (Le and Mikolov, 2014)—an extension of Word2Vec (Mikolov, 2013)—which generates semantically meaningful text embeddings of fixed-length.

In selecting the methods for our experiments, we were influenced by Wang et al. (2023)'s study, which investigated how to automatically separate an author's style from the content in their writing to isolate the stylist component. A. Wang et al. proposed various approaches to the problem. One of these approaches involved masking content words—such as nouns, adjectives, main verbs, and adverbs, which are assumed to convey topic signals—to isolate the author's style. Differently from Astuti et al. (2022), Gebauer et al. (2024), and Bouscasse et al. (2024), which focused on measuring and quantifying the sentiment or tone (hawkish/positive or dovish/negative)[3] of central bank communication, we adopted a different approach aimed to capture the writing style of the Governors in their speeches.

In this thesis, we analyze a collection of speeches delivered by the Governors of the Bank of Italy from 1990 to 2023, including speeches from Ciampi, Fazio, Draghi, and Visco. Unlike the studies mentioned above, which predominantly focused on linguistic and economic objectives, this thesis focuses more on data science and machine learning. Our goal is to create meaningful numerical representations of these documents, enabling us to classify[4] the Governors based on content and style and explore the underlying topics in their speeches. The thesis is divided into five chapters.

Following this introduction, Chapter 2 describes the methods and tools used to extract the dataset, along with a summary of its main characteristics.

Chapter 3 provides a theoretical overview of the algorithms and methodologies used in the experiments detailed in Chapter 4. This chapter is divided into three main sections. Section 3.1 focuses on document embedding techniques, specifically TF-IDF, Doc2Vec, and Sentence Transformers. All three methods address some of the limitations of the BoW approach used in Astuti et al. (2020), Astuti et al. (2022) and Priola et al. (2021). Section 3.2 covers the classification methods employed, i.e. Support Vector Machines (SVM) and Logistic Regression (applied to the TF-IDF and Doc2Vec embeddings), and fine-tuned Transformer-based models. Lastly, Section 3.3 offers a detailed explanation of the BERTopic topic modeling algorithm, which we used after LDA (following Astuti et al. 2022 and Priola et al. 2021) yielded unsatisfactory results. Instead of BoW, BERTopic uses semantically meaningful text representations created through Sentence Transformers, i.e. state-of-the-art deep

[3]Hawkish and dovish are two opposite monetary policy approaches. "Hawkish monetary policy focuses on low inflation and may involve raising interest rates, while dovish policy prioritizes low unemployment and may involve lowering rates" Franke-Folstad (2024).

[4]Our classification task coincides with *Authorship Attribution* (AA), which is "the process of attributing a text to the correct author among a closed set of potential writers" Fabien et al. (2020). In this case, the set includes Ciampi, Fazio, Draghi, and Visco.

learning embedding models.

Chapter 4 presents the practical application and findings of our experiments. This chapter is divided into three main sections. Section 4.1 focuses on classifying the speeches of the four Governors using the embedding and classification techniques outlined in Chapter 3. Section 4.2 delves into topic analysis, using BERTopic to identify the topics hidden in the documents. Subsection 4.2.2 further concentrates on the evolution of the topics addressed by the Governors in their final consideration speeches, drawing inspiration from Astuti et al. (2020). Finally, in Section 4.3, we attempt to isolate the Governors' writing styles by masking the content words, akin to Wang et al. (2023). However, we experiment with a different masking strategy that relies on masking topic-related words identified by the BERTopic algorithm. Then, the same embedding and classification techniques from Section 4.1 are applied, and the performances of these models are compared against the baseline models trained in Section 4.1.

Finally, in Chapter 5, we discuss the main findings of this work and address the following research questions:

- Can we classify the Governors based on the text of their speeches?
- Can we classify them according to their style?
- What are the most significant topics in the entire collection of documents?
- What are the prevalent themes the Governors addressed in their final consideration speeches, and do they change over time?

Chapter 2

Dataset

The dataset is composed of 504 documents containing the speeches of the Governors of the Bank of Italy from 1990 to 2023. The succession of Governors of the Bank of Italy was as follows:[1],

- Fabio Panetta (November 1, 2023 - Today)
- Ignazio Visco (November 1, 2011 - October 31, 2023)
- Mario Draghi (December 29, 2005 - October 31, 2011)
- Antonio Fazio (May 4, 1993 - December 20, 2005)
- Carlo Azeglio Ciampi (October 8, 1979 - April 29, 1993)

In Figure 2.1, we view the distribution of the documents at the time of extraction across the Governors. The majority of the speeches belong to Ignazio Visco while, on the contrary, only three documents are associated with the most recent Governor Fabio Panetta, whom we exclude from the rest of our work.

We scraped the files directly from the "`Speeches by the Governor`" section[2] of the Bank of Italy website. The documents were extracted using the `BeautifulSoup` library and divided into 33 folders, one per year.

Subsequently, we transformed the pdf files into text files. For most documents, we could extract the txt file directly from the corresponding pdf through the `pdfminer` module. Instead, for the more dated typewritten documents, i.e. the ones up to the year 1997, it was necessary to use more advanced technologies such as Optical Character Recognition (OCR)[3]. To apply OCR to our files, we first converted them from pdf format to images. Secondly, we exploited the `pytesseract.image_to_string` tool for `Python` that attempts to recognize and "read" the text embedded in images. Utilizing OCR technology, we have successfully extracted text from over 50 typewritten documents, as well as scans of foreign newspaper articles and other forms of photocopy documents.

[1] https://www.bancaditalia.it/chi-siamo/storia/governatori-direttori-generali/index.html.

[2] https://www.bancaditalia.it/pubblicazioni/interventi-governatore/index.html?com.dotmarketing.htmlpage.language=1.

[3] "[...] is the electronic or mechanical conversion of images of typed, handwritten or printed text into machine-encoded text [...]". Source: "Optical character recognition", Wikipedia, 7 May 2024.

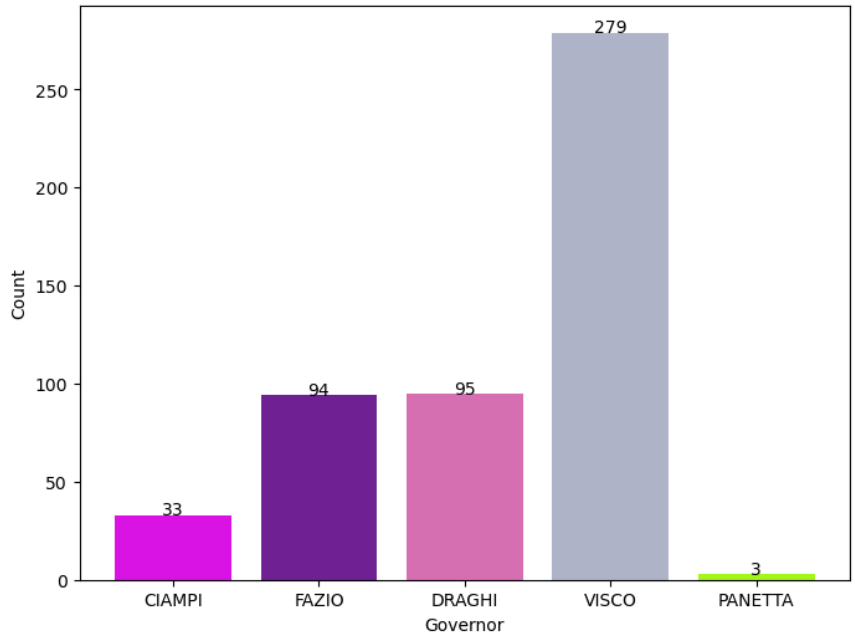


Figure 2.1. Plot of the distribution of the Governors of the Bank of Italy among the 502 documents.

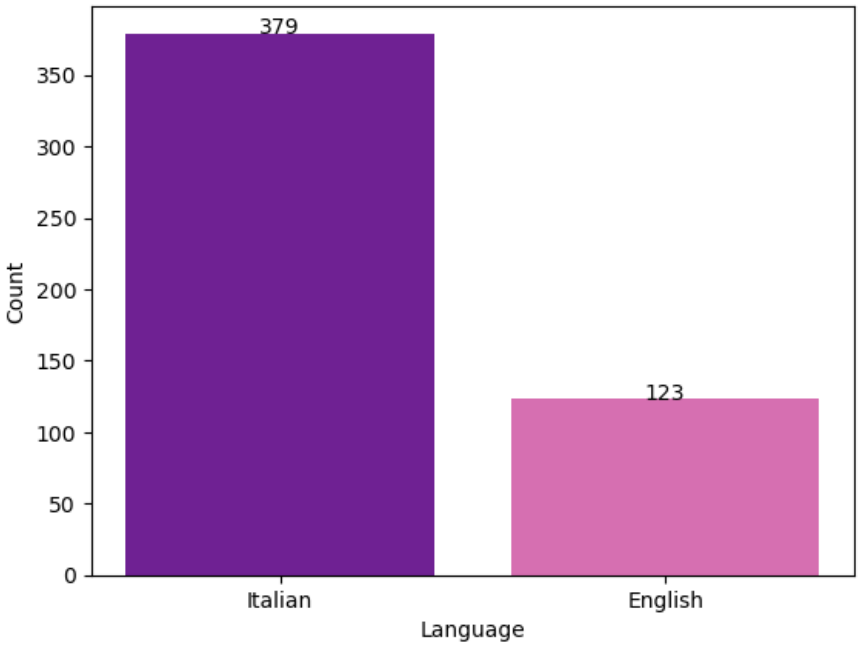


Figure 2.2. Plot of the distribution of the language of the speeches of the Governor of the Bank of Italy among the 502 documents.

Among all 504 documents, the `detect` function of the `langdetect` module labeled two documents as German and French and the remaining 502 files as Italian and English. We restrict our dataset solely to Italian and English documents that

are respectively 379 and 123, as shown in Figure 2.2. Excluding the three speeches of Governor Fabio Panetta, we are left with 499 documents, of which 377 are Italian and 122 are English.

The raw documents have an average length of 4717 words. The Italian documents seem to be dramatically longer than English-language ones, with 5240 words per document for the former and 3100 words per document for the latter. Due to the high variability of the dataset, which contains several exceptionally long documents, such as the annual reports, it's best to use a more robust statistical measure like the median. Specifically, the three medians are 4626 tokens for Italian, 2506 words for English, and 3838 words for mixed corpus.

Figure 2.3 illustrates the median word count of the speeches delivered by each Governor. It is evident that the documents of Antonio Fazio and Mario Draghi are longer and more detailed compared to those of Governors Carlo Azeglio Ciampi and Ignazio Visco, especially the former. It's important to note that most of Visco's speeches were conducted in English, and, as already discussed, English documents are, on average, shorter than Italian ones.

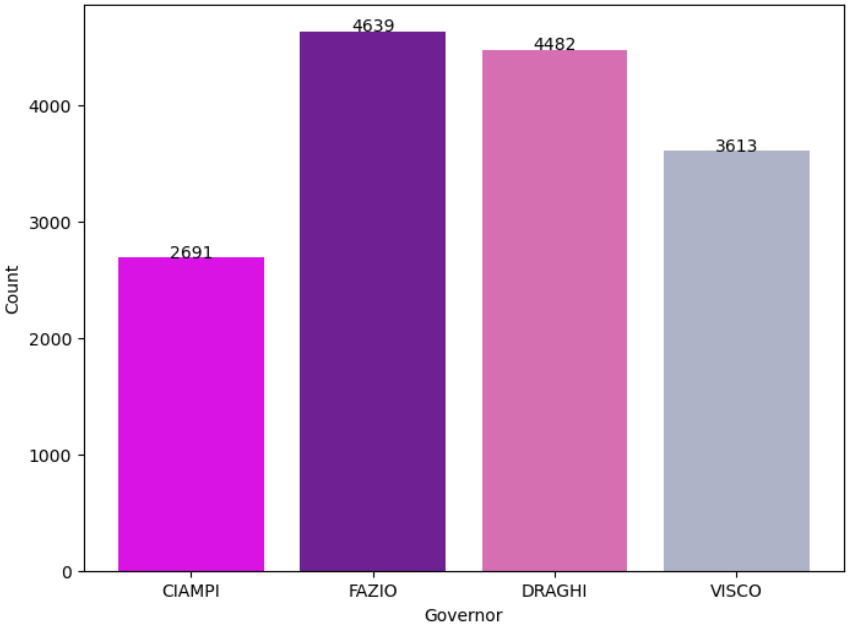


Figure 2.3. Plot of the median word count of the raw documents per Governor.

Within the 377 Italian documents, 30 are annual reports, one per year except for the years 1993, 1994, 1999, and 2001. The final consideration is usually "published at the end of May and is the subject of the Governor's Concluding Remarks at a public meeting, not restricted to shareholders. It contains a comprehensive analysis of the key developments in the Italian and international economy in the previous year and the early months of the current year [...]."[4]. In Section 4.2.2 we deal specifically with the annual reports to seek for trends in topics treated over time.

[4] https://www.bancaditalia.it/pubblicazioni/relazione-annuale/index.html?com.dotmarketing.htmlpage.language=1.

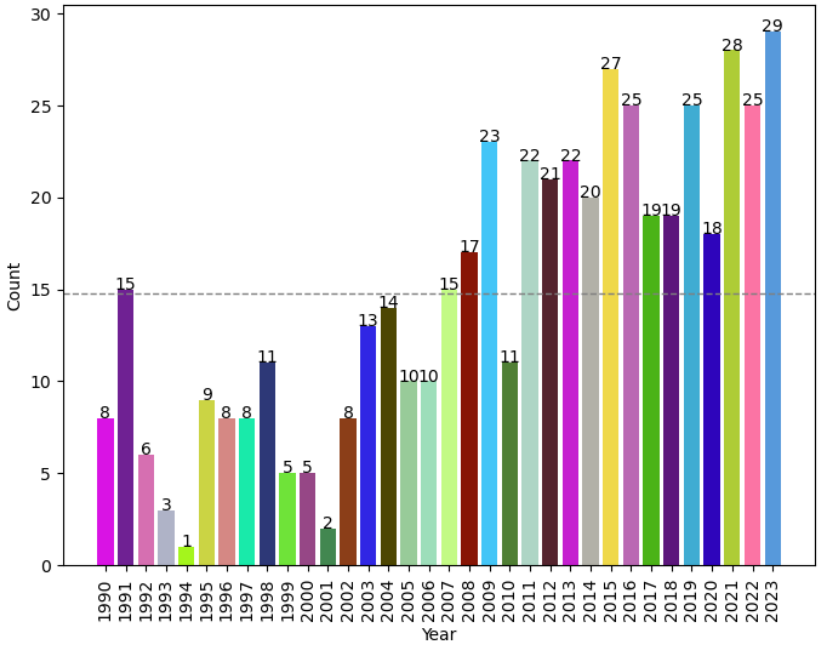


Figure 2.4. Plot of the distribution of the 502 documents across the years 1990-2023 with an average of 15 publications per year.

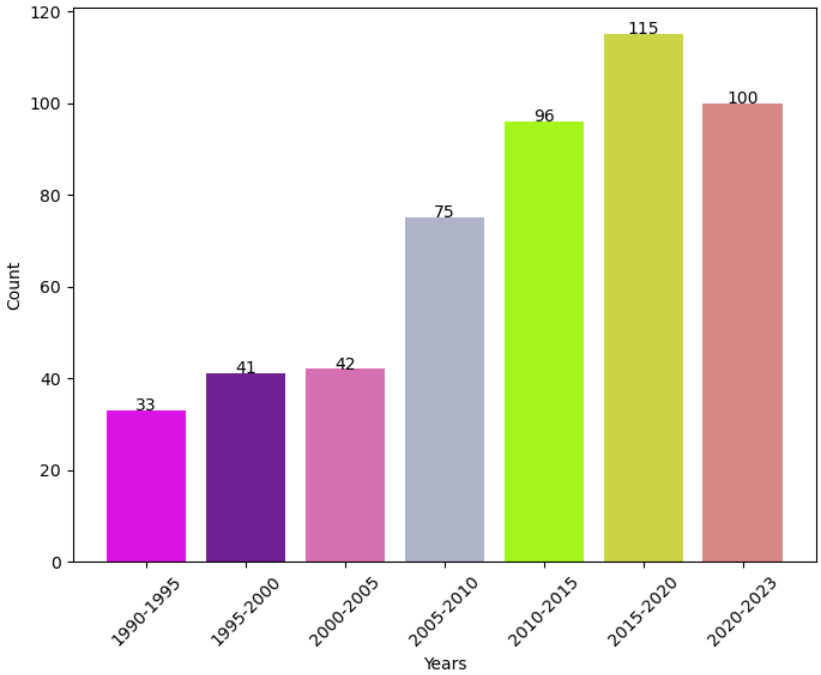


Figure 2.5. Plot of the distribution of the 502 documents over time, binned every five years. It is clear that our corpus contains more recent documents w.r.t. older ones.

Chapter 3

Theoretical Framework

3.1 Document Embedding Methods

Before proceeding with our tasks, we must transform our documents into an appropriate format, as machine learning algorithms cannot work with raw text. This "transformation" process is known as *embedding*. Document embeddings are numerical representations of strings of text. A prerequisite for embedding is a step called *tokenization*, i.e. the process of splitting a stream of text into a sequence of tokens. Typically, in natural language processing (NLP)[1], tokens are words, but they can also be subwords, characters, or n-grams[2], depending on the specific application.

Numerous ways exist to represent sets of tokens as n-dimensional numerical vectors, each having its own advantages. For instance, some embeddings can help reduce the dimensionality of the problem. Others can capture the position and context of words. In our work, we tested three very different types of document embeddings: TF-IDF, Doc2Vec, and Sentence Transformers.

3.1.1 TF-IDF

Term Frequency-Inverse Document Frequency (TF-IDF) is a traditional count-based document embedding technique introduced in the 1970s to address the limitations of the *Bag of Words* (BoW) approach. In the BoW model, each document is represented as a vector obtained by summing the one-hot encoded[3] representations of the tokens composing the document. This embedding strategy focuses solely on the presence or absence and frequency of tokens and ignores the order and semantics of the words[4].

Similarly, TF-IDF also ignores the order and semantics of the tokens. However, TF-IDF improves upon BoW by considering not only the frequency of a term inside a document but also how frequent the term is across the whole corpus of documents.

[1] Natural Language Processing (NLP) is a field that enables computers to understand, interpret, and generate human language. It combines linguistics, computer science, and statistics to process and analyze large amounts of natural language data.

[2] Ordered sequences of 2 or more words.

[3] In one-hot encoding each document is represented as a k-dimensional binary vector v, where k is the number of unique tokens contained in the whole corpus of documents. Each entry v_i equals 1 if the ith word is present in the document and 0 otherwise.

[4] The term "bag" in Bag of Words (BoW) signifies that each document is represented as a "bag" of words rather than as a sequence.

As Boykis (2023) states, "TF-IDF will tell you how important a single word is in a corpus by assigning it a weight and, at the same time, down-weight common words like, 'a', 'and', and 'the'[5]. This calculated weight gives us a feature for a single word in TF-IDF, and also the relevance of the features across the vocabulary".

Formally, we introduce the TF-IDF score for token t of document d as

$$(\text{tf-idf})_{td} = \text{tf}_{td} \times \text{idf}_t, \tag{3.1}$$

where tf_{td} is the term frequency of word t belonging to document d, and idf_t is the inverse document frequency of word t in relation to the whole corpus of documents D. We define the term frequency (TF) as "the number of times a term t appears in a document d relative to the other terms in the document" Boykis (2023), i.e.

$$\text{tf}_{td} = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} = \frac{\text{Frequency of term } t \text{ in document } d}{\text{Total number of terms in document } d}. \tag{3.2}$$

Term frequency is biased towards longer documents: if a document is lengthy, the denominator of tf_{td} is very large, hence also frequent terms lose importance. We might consider computing *relative term frequency* $\text{Rel-tf}_{t,d}$ instead of *absolute term frequency* tf_{td} to solve this problem.

$$\text{Rtf}_{t,d} = \frac{\text{Frequency of term } t \text{ in document } d}{\text{Frequency of the most common term in document } d}. \tag{3.3}$$

Term specificity (Sparck Jones, 1972), later known as *Inverse document frequency* (IDF), is defined as

$$\begin{aligned} \text{idf}_t &= \log_{10} \frac{|\text{D}|}{|\{d \in \text{D} : t \in d\}| + 1} \\ &= \log_{10} \frac{\text{Total number of documents in the corpus}}{\text{Number of documents containing term } t}. \end{aligned} \tag{3.4}$$

The inverse document frequency score reduces the importance of words that occur frequently across the entire corpus, while boosting the score of infrequent words, as they might be significant in identifying a particular document. Certain terms, such as stopwords, might appear across the entire document base; hence, the inverse document frequency will heavily down-rank them.

TF-IDF is easy to implement. However, it has several issues. The resulting embedding matrix tends to be extremely large, especially when dealing with large corpora. These matrices are sparse, meaning that the majority of their entries are zero. Additionally, the TF-IDF embedding technique disregards the order of the tokens, and consequently, the context in which they appear, which can be crucial for capturing the semantic meaning of the text.

3.1.2 Doc2Vec and Word2Vec

Doc2Vec, short for "document to vector", is an extension of *Word2Vec*. Both Word2Vec and Doc2Vec are algorithms for learning vector representations of text.

[5] Also known as *stopwords*.

However, while Word2Vec encodes individual words, Doc2Vec learns vector representations for entire paragraphs or documents. Since Word2Vec is the building block of Doc2Vec, we first need to understand the functioning of the Word2vec model to comprehend the Doc2Vec embedding technique.

Word2Vec was introduced in 2013 over two papers by a team of researchers led by Mikolov at Google (Mikolov, 2013; Le and Mikolov, 2014). This algorithm addresses some of the limitations of TF-IDF. Unlike the latter, Word2Vec allows us to choose the length of the output embeddings independently from the vocabulary size, as seen in Section 3.1.1. These embeddings are also able to capture semantic meaning by considering word context and order in sentences. The goal is to map words into an N-dimensional embedding space where semantically similar words are "close" in terms of measured cosine similarity. Conversely, dissimilar words are far apart. Since we know, thanks to linguistics, the semantic meaning of words, we can test the effectiveness of our model using analogies. One very famous example is the "Woman is to queen as man is to king" analogy. Performing simple algebraic operations on the word vectors we get

$$\mathbf{b} = \mathbf{v}_{\text{queen}} - \mathbf{v}_{\text{woman}} + \mathbf{v}_{\text{man}} \approx \mathbf{v}_{\text{king}},$$

where $\mathbf{v}_{\text{queen}}$, $\mathbf{v}_{\text{woman}}$, $\mathbf{v}_{\text{man}}$ and $\mathbf{v}_{\text{king}}$ are the word vectors for *queen*, *woman*, *man* and *king* respectively, and meaning that vector $\mathbf{b}$ is closest to the vector representation of the word *king*.

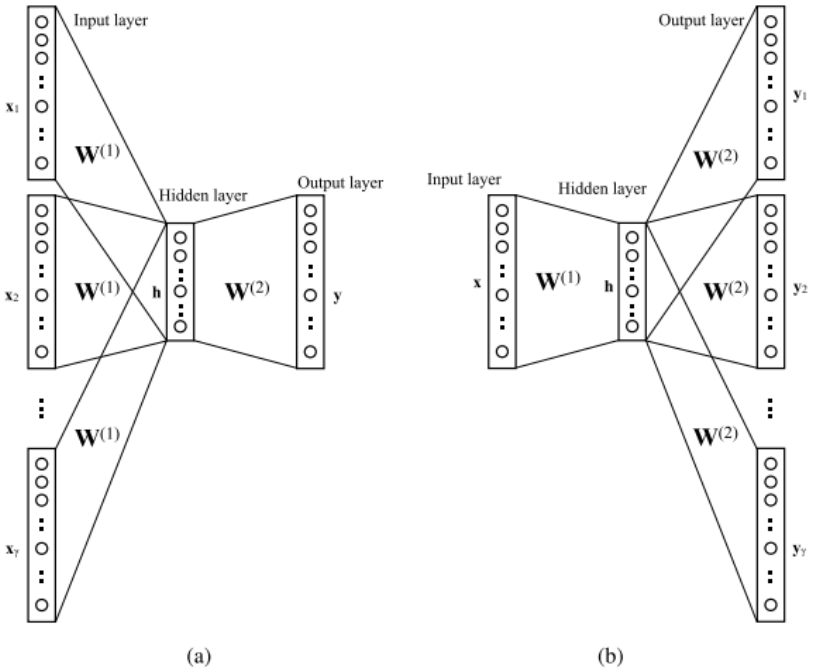


Figure 3.1. CBOW (a) versus Skipgram (b) diagram. Source: Park et al. (2019).

Word2Vec models are shallow neural networks that consist of one input layer, one hidden layer, and one output layer. There are two main Word2Vec models, namely the *skip-gram* model and the *continuous bag-of-words* (CBOW) model. In these

two models, the input and output layers are reversed, like in a mirror (see Figure 3.1). Since the `gensim.models.doc2vec` module we used in this work exploits the CBOW model[6], we can focus on the latter rather than the skip-gram model.

Figure 3.2 displays the structure of the CBOW model. The input layer consists of the one-hot encoded representations of the input context words $\{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x}_C\}$ w.r.t. the center word $\mathbf{y}$, where C is the length of the *word context window*. The input context words are projected into a binary V-dimensional space, where V is the vocabulary size. The linear hidden layer is an N-dimensional vector $\mathbf{h}$ containing a set of neurons (hidden units). Lastly, the output layer is the one-hot encoded representation of the output word $\mathbf{y}$, also of length V, whose context is given in the input layer. "The one-hot encoded input vectors are connected to the hidden layer via a $V \times N$ weight matrix $\mathbf{W}$ and the hidden layer is connected to the output layer via a $N \times V$ weight matrix $\mathbf{W'}$" Minnaar (2015). The CBOW model aims to predict the next word (the output word $\mathbf{y}$) in a sentence given the other words in the context.

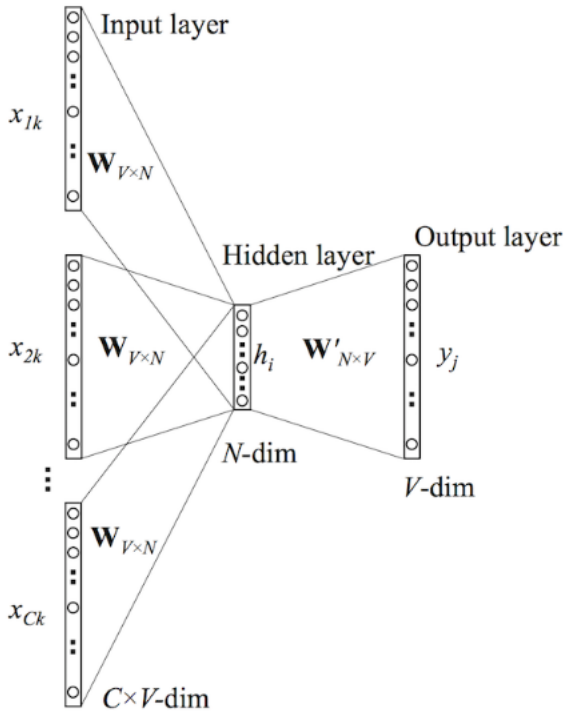


Figure 3.2. Continuous Bag of Words diagram. Source: Minnaar (2015).

Just like for any artificial neural network, we aim at optimizing the weight matrices iteratively by alternating forward and backward propagation steps, within a stochastic gradient descent optimization procedure. First, we must understand how the output $\mathbf{y}$ is computed from the input context vectors $\{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x}_C\}$, also

[6]The distributed memory `dm` parameter is set to `1`, hence the PV-DM model is used. This algorithm is further explained in paragraph 2.1.2.

known as *forward propagation*. We can compute the output of the hidden layer $\mathbf{h}$ as

$$\mathbf{h} = \frac{1}{\mathrm{C}}\mathbf{W}^{\intercal} \cdot \left(\sum_{i=1}^{\mathrm{C}} \mathbf{x_i}\right) = \mathbf{W}^{\intercal} \cdot \bar{\mathbf{x}} \in \mathbb{R}^{\mathrm{N}\times 1}, \tag{3.5}$$

which is the average $\bar{\mathbf{x}}$ of the input context vectors $\mathbf{x_i} \in \mathbb{R}^{\mathrm{V}\times 1}$ weighted by the input matrix $\mathbf{W} \in \mathbb{R}^{\mathrm{V}\times \mathrm{N}}$. We can use $\mathbf{h}$ to compute the inputs to each node in the output layer

$$u_j = \mathbf{v'_{W_j}}^{\intercal} \cdot \mathbf{h} \in \mathbb{R}, \tag{3.6}$$

where $\mathbf{v'_{W_j}} \in \mathbb{R}^{\mathrm{N}\times 1}$ is the jth column of the output matrix $\mathbf{W'}$. Equivalently, we can write

$$\mathbf{u} = \mathbf{W'}^{\intercal} \cdot \mathbf{h} = \mathbf{W'}^{\intercal} \cdot \mathbf{W}^{\intercal} \cdot \bar{\mathbf{x}}. \tag{3.7}$$

Finally, we compute the output of the output layer $\mathbf{y}$ by applying the *soft-max activation function* to u_j.

$$y_j = p(w_{y_j}|w_1, \ldots, w_{\mathrm{C}}) = \mathrm{Softmax}\left(u_j\right) = \frac{e^{u_j}}{\sum_{z=1}^{\mathrm{V}} e^{u_z}} \tag{3.8}$$

where y_j is the probability of word w_{y_j} in the vocabulary of being the true output word, given the input context words $\{w_1, \ldots, w_{\mathrm{C}}\}$. Equivalently,

$$\mathbf{y} = \mathrm{Softmax}\left(\mathbf{u}\right) = \mathrm{Softmax}\left(\mathbf{W'}^{\intercal} \cdot \mathbf{W}^{\intercal} \cdot \bar{\mathbf{x}}\right). \tag{3.9}$$

The soft-max function converts the raw output scores $\mathbf{u}$ into a vector $\mathbf{y}$ of real variables y_i, with $0 \leq y_j \leq 1,\ \forall j \in [1, \mathrm{V}]$ and $\sum_{j=1}^{\mathrm{V}} y_j = 1$, that can be interpreted as a probability distribution. The exponentiation step ensures that all entries are positive, and the normalization step guarantees that all values of $\mathbf{y}$ sum up to 1. We select as the output word the word with the highest probability in $\mathbf{y}$.

So far, we assumed that the weight matrices $\mathbf{W}$ and $\mathbf{W'}$ are known. However, the optimization process aims to learn the input and output weight matrices[7] by exploiting a procedure called *backpropagation*. The first step is to define the *loss function*, which quantifies the errors made by our model by "comparing" the predicted output word $\mathbf{y}$ with the true output word. We aim at minimizing the loss function E or equivalently maximizing the conditional probability of the predicted output word given the input context. The loss function is the *negative log-likelihood loss*

$$\mathrm{E} = -\log\, p\left(w_{\mathrm{O}}|w_{\mathrm{I}}\right), \tag{3.10}$$

where $p\left(w_{\mathrm{O}}|w_{\mathrm{I}}\right)$ is the predicted probability of the true output word w_{O} given the

[7]The input and output weight matrices are randomly initialized at the beginning of the learning process.

input context words w_{I}. Equivalently, we can write

$$\begin{aligned}
\mathrm{E} &= -\log\left(y_{j^*}\right) \\
&= -\log\frac{e^{u_{j^*}}}{\sum_{z=1}^{V} e^{u_z}} \\
&= -\log\left(e^{u_{j^*}}\right) + \log\left(\sum_{z=1}^{\mathrm{V}} e^{u_z}\right) \\
&= -u_{j^*} + \log\left(\sum_{z=1}^{\mathrm{V}} e^{u_z}\right) \\
&= -\mathbf{v'}_{w_{\mathrm{O}}}^{\intercal} \cdot \mathbf{h} + \log\left(\sum_{z=1}^{\mathrm{V}} e^{\left(\mathbf{v'}_{w_z}^{\intercal} \cdot \mathbf{h}\right)}\right)
\end{aligned} \quad , \tag{3.11}$$

where j^* is the index of the true output word in the vocabulary, and $\mathbf{v'}_{w_{\mathrm{O}}}$ is the column of weight matrix $\mathrm{W'}$ correspondent to the output word w_{O}.

Once the loss is defined, we can compute the gradient of this loss w.r.t. the elements of both weight matrices via backpropagation. We exploit the *chain rule* to compute the derivative of E w.r.t. an arbitrary output weight $w'_{i,j}$, i.e. the element in the ith row and jth column of the hidden-output weight matrix

$$\frac{\partial \mathrm{E}}{\partial w'_{i,j}} = \frac{\partial \mathrm{E}}{\partial u_j} \cdot \frac{\partial u_j}{\partial w'_{i,j}}. \tag{3.12}$$

Firstly, we define the partial derivative of the loss w.r.t. the input to the jth node in the output layer as

$$\frac{\partial \mathrm{E}}{\partial u_j} = \frac{\frac{\partial}{\partial u_j}\left(\sum_{z=1}^{\mathrm{V}} e^{u_z}\right)}{\sum_{z=1}^{\mathrm{V}} e^{u_z}} - \delta_j = \frac{\frac{\partial}{\partial u_j} e^{u_j}}{\sum_{z=1}^{\mathrm{V}} e^{u_z}} - \delta_j = \frac{e^{u_j}}{\sum_{z=1}^{\mathrm{V}} e^{u_z}} - \delta_j = y_j - \delta_j, \tag{3.13}$$

where δ_j is a Kronecker delta: $\delta_j = 1$ when $j = j^*$ and $\delta_j = 0$ otherwise. $y_j - \delta_j$ is the prediction error of the jth node of the output word. Secondly, we define the partial derivative of u_j w.r.t. $w'_{i,j}$ as

$$\begin{aligned}
\frac{\partial u_j}{\partial w'_{i,j}} &= \frac{\partial}{\partial w'_{i,j}}\ \mathbf{v'}_j^{\intercal} \cdot \mathbf{h} \\
&= \frac{\partial}{\partial w'_{i,j}}\ \sum_i w'_{i,j} \cdot h_i = h_i.
\end{aligned} \tag{3.14}$$

We can substitute the two partial derivatives in equation (3.12) and obtain

$$\frac{\partial \mathrm{E}}{\partial w'_{i,j}} = (y_j - \delta_j) \cdot h_i. \tag{3.15}$$

Finally, we can update an arbitrary entry $w'_{i,j}$ of the hidden-output weight matrix through a *stochastic gradient descent* step, which directs the weight matrices in the negative direction of the gradient in the following way

$$\begin{aligned}
w'^{\,(\mathrm{old})}_{i,j} &= w'^{\,(\mathrm{new})}_{i,j} - \eta \cdot \frac{\partial \mathrm{E}}{\partial w'_{i,j}} \\
&= w'^{\,(\mathrm{new})}_{i,j} - \eta \cdot (y_j - \delta_j) \cdot h_i,
\end{aligned} \tag{3.16}$$

where $\eta > 0$ is the learning rate.

Again, we can use the chain rule to compute the derivative of E w.r.t. an arbitrary input weight $w_{k,i}$, i.e. the element of the kth row and ith column of the input-hidden weight matrix

$$\frac{\partial \mathrm{E}}{\partial w_{k,i}} = \frac{\partial \mathrm{E}}{\partial h_i} \cdot \frac{\partial h_i}{\partial w_{k,i}}. \tag{3.17}$$

"The first step is to compute the derivative of E with respect to an arbitrary hidden node h_i

$$\begin{aligned} \frac{\partial \mathrm{E}}{\partial h_i} &= \sum_{j=1}^{\mathrm{V}} \frac{\partial \mathrm{E}}{\partial u_j} \cdot \frac{\partial u_j}{\partial h_i} \\ &= \sum_{j=1}^{\mathrm{V}} (y_j - \delta_j) \cdot w'_{i,j}, \end{aligned} \tag{3.18}$$

where the sum is due to the fact that the hidden layer node h_i is connected to each node of the output layer" Minnaar (2015), $\frac{\partial \mathrm{E}}{\partial u_j}$ was computed in equation (3.13), and $\frac{\partial u_j}{\partial h_i}$ can be derived from equation (3.14). Secondly, we can compute the derivative of h_i w.r.t. an arbitrary input weight $w_{k,i}$

$$\frac{\partial h_i}{\partial w_{k,i}} = \frac{\partial}{\partial w_{k,i}} \left(\sum_{k=1}^{\mathrm{V}} w_{k,i} \cdot (\bar{\mathbf{x}})_k \right) = \bar{x}_k, \tag{3.19}$$

where $\bar{x}_k$ is the value at the k-th node of the average vector $\bar{\mathbf{x}}$ of the input context words. We can substitute the two partial derivatives in equation (3.17) and get

$$\frac{\partial \mathrm{E}}{\partial w_{k,i}} = \sum_{j=1}^{\mathrm{V}} (y_j - \delta_j) \cdot w'_{i,j} \cdot \bar{x}_k. \tag{3.20}$$

Ultimately, we can update an arbitrary entry $w_{k,i}$ of the input-hidden weight matrix by performing the following stochastic gradient descent operation

$$\begin{aligned} w_{k,i}^{(\mathrm{old})} &= w_{k,i}^{(\mathrm{new})} - \eta \cdot \frac{\partial \mathrm{E}}{\partial w_{k,i}} \\ &= w_{k,i}^{(\mathrm{new})} - \eta \cdot \sum_{j=1}^{\mathrm{V}} (y_j - \delta_j) \cdot w'_{i,j} \cdot \bar{x}_k. \end{aligned} \tag{3.21}$$

Once the training converges[8], we can discard the hidden-output weight matrix $\mathbf{W}'$ and retain the input-hidden weight matrix $\mathbf{W}$. The rows of $\mathbf{W}$ are the N-dimensional vector representations of all the words in the vocabulary, such that semantically similar words are "close" in the vector space. For instance, synonyms like *strong* and *powerful* have similar positions in space, but are distant w.r.t. a dissimilar word like *Paris*.

For our purposes, the idea is to extend the CBOW Word2Vec model "to go beyond word-level to achieve phrase-level or sentence-level representations" Mikolov

[8]Meaning that the training process has reached a maximum number of iterations set a priori, or the gradient of the loss approaches zero, i.e. $|\nabla \mathrm{E}| \leq c$ where c is a very small positive constant.

(2013). One simple approach that exploits the CBOW model consists in computing the weighted average of the vector representations of all the words in the document. However, this approach loses the order of the words within the document, hence behaving in the same way as a standard BOW model. On the other hand, Doc2vec was proposed to potentially overcome the weaknesses of BOW models (including TF-IDF). This algorithm, also known as *Paragraph Vector*, was introduced by a team of researchers led by Mikolov at Google as "an unsupervised[9] algorithm that learns fixed-length feature representations from variable-length pieces of texts, such as sentences, paragraphs, and documents" Le and Mikolov (2014). This model represents documents as *dense vectors*, unlike the sparse representations produced with the TF-IDF approach, which capture the semantic meaning of the entire document. Moreover, compared to TF-IDF, we can choose the output size independently of the vocabulary size V.

In Doc2Vec there are two main approaches: *Distributed Memory Model of Paragraph Vectors* (PV-DM) and *Distributed Bag of Words* (DBOW). The first approach extends the idea of CBOW, while DBOW takes inspiration from the skip-gram model. As the `dm`[10] parameter of the `gensim.models.doc2vec` module is set to `1` (`default`), we can focus on the PV-DM model rather than the DBOW model.

PV-DM consists in modifying the structure of the CBOW model by introducing *paragraph tokens*, i.e. one-hot encoded representations of the documents. Supposing we have P documents in our corpus, each input paragraph token $\mathbf{d}$ has size P. Each document is mapped to a unique, lower-dimensional vector of size M contained in a $(\mathrm{P} \times \mathrm{M})$ paragraph input-hidden weight matrix $\mathbf{D}$, in the same way the rows of the word input-hidden weight matrix $\mathbf{W}$ represent the words within the vocabulary. As displayed in Figure 3.3, the paragraph token $\mathbf{d}$ contributes to the prediction task in the same manner the input context words do. Formally, the only change in comparison to the CBOW model regards the computation of the hidden layer $\mathbf{h}$, seen in equation (3.5). As a matter of fact, vector $\mathbf{h}$ is no longer computed as the weighted average of the input context word vectors, but rather as the *concatenation* of the *paragraph vector*, i.e. the paragraph token $\mathbf{d}$ weighed by matrix $\mathbf{D}$, and several word vectors, i.e. the one-hot encoded context words $\mathbf{x_i}$ weighted by matrix $\mathbf{W}$. Mathematically, we can write

$$\mathbf{h} = \begin{bmatrix} \mathbf{D}^\intercal \cdot \mathbf{d} & \mathbf{W}^\intercal \cdot \mathbf{x_1} & \mathbf{W}^\intercal \cdot \mathbf{x_2} & \dots & \mathbf{W}^\intercal \cdot \mathbf{x}_\mathrm{C} \end{bmatrix}^\intercal \in \mathbb{R}^{(\mathrm{M}+\mathrm{CN})\times 1}, \tag{3.22}$$

where M is the size of the embedded paragraph vectors, i.e. the rows of $\mathbf{D}$, N is the size of the embedded word vectors, i.e. the rows of $\mathbf{W}$, and C is the size of the context

[9] Why is Word2Vec defined as an unsupervised algorithm? The final product of Word2Vec is the input-hidden weight matrix W, i.e. a matrix of compressed lower-dimensional representations of the original input data. Like any other dimensionality reduction technique, e.g. Principal Component Analysis or Singular Value Decomposition, Word2Vec is seen as an unsupervised algorithm, even though the training process requires a supervised approach. Moreover, the true labels of the training data are highly dependent on the input context words and the assumptions made at the beginning of the learning process, such as the size of the context sliding window C. On the contrary, in traditional supervised learning, the class labels are given as input and will not change according to the model hyperparameters.

[10] If `dm=1`, "distributed memory" PV-DM is used. Otherwise, PV-DBOW is employed.

sliding window, i.e. the number of input context words sampled from the paragraph or document. The output of the hidden layer **h** is an (M + CN)-dimensional vector used to predict the following word in the context. "The paragraph token can be thought of as another word. It acts as a memory that remembers what is missing from the current context or the topic of the paragraph" Le and Mikolov (2014). This explains why the Doc2Vec model is often called Distributed Memory Model of Paragraph Vectors (PV-DM).

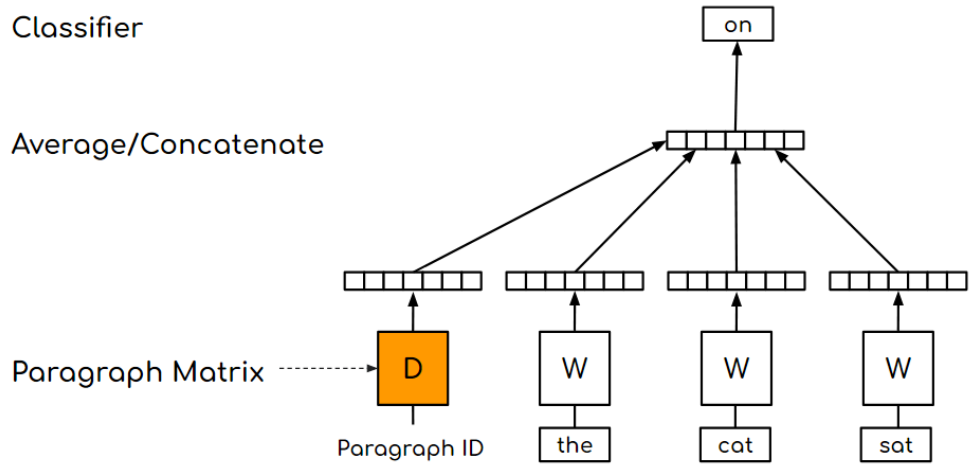


Figure 3.3. Paragraph Vectors. Source: Le and Mikolov (2014).

The first phase of the Paragraph Vector algorithm is the "training stage", which involves sampling a C-length context from a *random paragraph* **d**. During this phase, the entries of weight matrices[11] **W**, **W′** and **D** are updated using stochastic gradient descent, where the gradient of the loss E[12] w.r.t. all model parameters is obtained via backpropagation. This process is repeated at every step of stochastic gradient descent. "The contexts are fixed-length and sampled from a sliding window over the paragraph. The paragraph vector is shared across all contexts generated from the same paragraph, but not across paragraphs[13]. The word vector matrix **W**, however, is shared across paragraphs" Le and Mikolov (2014).

The second phase, called the "inference stage", involves computing the paragraph vectors for new, unseen paragraphs. Again, gradient descent is used, but in this phase, only the paragraph vectors are updated while keeping the word vectors fixed.

The model has a total of (P × M + V × N) learnable parameters. Although the number of documents P and vocabulary size V can be very large, leading to a high number of parameters, the updates during training are typically sparse. This sparsity makes the Doc2Vec model efficient regardless of the number of parameters.

[11]The weight matrices are initialized with random values.

[12]The loss in equation (3.10) can be adapted to the Doc2Vec model by writing

$$\mathrm{E} = -\log p\left(w_{\mathrm{O}} | \mathbf{d}, w_{\mathrm{I}}\right)$$

where w_{O} is the output word, and the paragraph token **d** and input context words w_{I} compose the input layer of the Doc2Vec model.

[13]As each document is *uniquely* mapped to a row in the paragraph weight matrix **D**.

3.1.3 Sentence Transformers

Sentence Transformers[14] are specialized models built on top of the Transformer architecture, designed to generate high-quality sentence embeddings. These embeddings are fixed-size vectors that capture the semantic meaning of sentences or chunks of text. Hence, the term "sentence" in "Sentence Transformer" refers to the model's focus on producing sentence embeddings, while "transformer" indicates that these models leverage pre-trained *Transformer* models.

The Transformer

Let's first focus on a brief explanation of the Transformer architecture, as described in Vaswani (2017). A Transformer-based model can have an *encoder-decoder*, *encoder-only* or *decoder-only* structure. The model proposed in the aforementioned publication explains the canonical encoder-decoder model architecture, displayed in Figure 3.4. The block on the left is the *encoder*, whose outputs are fed to the *decoder*, the block on the right. As all the Sentence Transformers encountered in this work are based on encoder-only Transformers, we can focus explaining the latter.

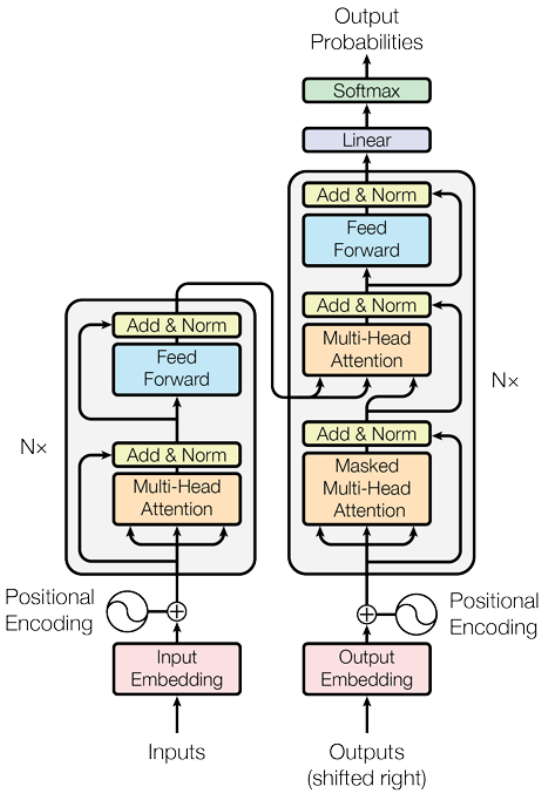


Figure 3.4. The Transformer - model architecture. Source: Vaswani (2017).

An encoder block is composed of a stack of L identical layers, 12 for the `BERT-base` model and 24 for the `BERT-large` model. "Each layer has two sub-layers. The first is a *Multi-Head self-Attention mechanism* (MHA), and the second is a

[14] `sentence_transformers.SentenceTransformer`.

simple, position-wise *fully connected feed-forward network* $g(\cdot)$. We employ a *residual connection*[15] around each of the two sub-layers, followed by *layer normalization*[16]" Vaswani (2017), as shown in the equations below

$$\begin{aligned} \mathbf{H} &= \text{MHA}(\mathbf{X}) \\ \mathbf{H} &= \text{LayerNorm}(\mathbf{H} + \mathbf{X}) \\ \mathbf{F} &= g(\mathbf{H}) \\ \mathbf{H} &= \text{LayerNorm}(\mathbf{F} + \mathbf{H}). \end{aligned} \tag{3.23}$$

The core of the Transformer model is the *self-attention mechanism*, which allows the model to focus on different parts of the input sequence, mimicking the way human attention works. This mechanism works by computing *attention scores* between pairs of tokens $(\mathbf{x_i}, \mathbf{x_j})$ in the sequence. Specifically, the *scaled dot-product attention* function $\alpha(\cdot, \cdot)$ is used

$$\alpha(\mathbf{x_i}, \mathbf{x_j}) = \frac{1}{\sqrt{d}} \mathbf{x_i}^\intercal \mathbf{x_j}, \tag{3.24}$$

where $\mathbf{x_i}$ and $\mathbf{x_j}$ are two d-dimensional vectors. Now, we can define the *self-attention layer* (for a single token i) as

$$\mathbf{h_i} = \sum_{j=1}^{n} \text{Softmax}_j \left(\frac{1}{\sqrt{d}} \mathbf{x_i}^\intercal \mathbf{x_j} \right) \mathbf{x_j}, \tag{3.25}$$

where the Softmax function ensures that the attention scores are normalized, converting them into probabilities, also called *attention weights*. This formulation of the self-attention layer lacks trainable parameters. Hence, we must introduce 3 *trainable matrices*, namely $\mathbf{W}_q \in \mathbb{R}^{d \times d_k}$, $\mathbf{W}_k \in \mathbb{R}^{d \times d_k}$, $\mathbf{W}_v \in \mathbb{R}^{d \times d_v}$, such that we can define the following vectors

$$\mathbf{q_i} = \mathbf{W}_q^\intercal \cdot \mathbf{x_i}, \ \mathbf{k_j} = \mathbf{W}_k^\intercal \cdot \mathbf{x_j}, \ \mathbf{v_j} = \mathbf{W}_v^\intercal \cdot \mathbf{x_j}, \tag{3.26}$$

which we call *query*, *key* and *value*. We substitute vectors $\mathbf{q_i}, \mathbf{k_i}$ and $\mathbf{v_i}$ in equation (3.25) and obtain

$$\mathbf{h_i} = \sum_{j=1}^{n} \text{Softmax}_j \left(\frac{1}{\sqrt{d_k}} \mathbf{q_i}^\intercal \mathbf{k_j} \right) \mathbf{v_j}, \tag{3.27}$$

[15] A *residual or skip connection* connects the output of a previous layer with the output of a future layer, skipping any transformation in between. In the context of a Transformer encoder layer, with the first residual connection, the input to the first sub-layer X connects to the output of the first sub-layer MHA(X), and with the second residual connection, the output of the first sub-layer, after applying layer normalization, connects to the output of the second sub-layer g(H).

[16] *Layer normalization* is a variant of batch normalization that normalizes the inputs across the features for each data sample rather than across the batch. First, we normalize the input features

$$\hat{x}_i = \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}}$$

where μ and σ^2 are respectively the mean and variance across the feature dimension, and ϵ is a small constant for numerical stability. Secondly, we apply the learned scale α and shift β parameters to $\hat{x}_i$ and obtain

$$\text{output}_i = \alpha \cdot \hat{x}_i + \beta$$

where the attention scores are calculated based on the dot product of query and key vectors, scaled by the square root of their dimension. Finally, we can rewrite the self-attention layer in vectorized form by stacking the n input vectors $\mathbf{x_i}$ into a matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$.

$$\text{Attention Scores} = \frac{\mathbf{Q}\mathbf{K}^\intercal}{\sqrt{d_k}} \in \mathbb{R}^{n \times n} \tag{3.28}$$

$$\text{Attention Weights} = \text{Softmax}\left(\text{Attention Scores}\right) \in \mathbb{R}^{n \times n} \tag{3.29}$$

$$\mathbf{H} = \text{Attention Weights} \cdot \mathbf{V} \in \mathbb{R}^{n \times d_v}, \tag{3.30}$$

where $\mathbf{H}$ is the output of the self-attention layer, $\mathbf{Q} = \mathbf{X}\mathbf{W}_q \in \mathbb{R}^{n \times d_k}$, $\mathbf{K} = \mathbf{X}\mathbf{W}_k \in \mathbb{R}^{n \times d_k}$ and $\mathbf{V} = \mathbf{X}\mathbf{W}_v \in \mathbb{R}^{n \times d_v}$.

To enhance the model's capability, Transformers employ *Multi-Head Attention* (MHA), which allows the model to capture various aspects of the input sequence simultaneously, e.g. different meanings of a word in different contexts. The MHA layer works by computing $t = 1, \ldots, h$ separate sets of keys, queries and values, with h being the number of *attention heads*

$$\mathbf{Q}_t = \mathbf{X}\mathbf{W}_{q,t},\ \mathbf{K}_t = \mathbf{X}\mathbf{W}_{k,t},\ \mathbf{V}_t = \mathbf{X}\mathbf{W}_{v,t} \tag{3.31}$$

$$\mathbf{H}_t = \text{Softmax}\left(\frac{\mathbf{Q}_t\mathbf{K}_t^\intercal}{\sqrt{d_k}}\right)\mathbf{V}_t. \tag{3.32}$$

The outputs from all attention heads $\mathbf{H}_t$ are concatenated over the embedding dimension and projected again using the *trainable output matrix* $\mathbf{W}_o \in \mathbb{R}^{hd_v \times d_o}$, where d_o is the output size

$$\text{MHA}(\mathbf{X}) = [\mathbf{H}_1, \ldots, \mathbf{H}_h] \cdot \mathbf{W}_o \in \mathbb{R}^{n \times d_o}. \tag{3.33}$$

Figure 3.4 highlights another key component of Transformer models known as *positional encodings or embeddings*. These positional encodings, represented by the matrix $\mathbf{E}$, are either concatenated or summed with the token embeddings $\mathbf{Y}$ before being fed into the first Multi-Head Attention (MHA) layer. The purpose of positional encodings is to provide information about the position of each token within the input sequence, which allows the model to capture the order of tokens. The Transformer model uses sinusoidal functions to generate positional embeddings. The idea is to alternate sine and cosine functions of the same frequency w_j

$$w_j = \frac{1}{10000^{\frac{2j}{d}}}, \tag{3.34}$$

where d is the dimension of both the positional encoding and token embedding vectors. For position i and dimension j we get

$$[\mathbf{E}]_{i,2j} = sin\left(w_j\right),\ [\mathbf{E}]_{i,2j+1} = cos\left(w_j\right). \tag{3.35}$$

The positional embeddings are then added to the token embeddings to obtain the input matrix $\mathbf{X}$

$$[\mathbf{X}]_{i,:} = [\mathbf{Y}]_{i,:} + [\mathbf{E}]_{i,:}, \tag{3.36}$$

where each row $[\mathbf{E}]_{i,:}$ uniquely encodes the position of every element in the sequence. This combined representation incorporates the content of the token and its position in the sequence.

Sentence-BERT (SBERT)

The concept of Sentence Transformers was introduced in 2019 by Nils Reimers and Iryna Gurevych in their paper titled "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks". This initial model, referred to as SBERT, leverages the well-known Transformer architecture called *Bidirectional Encoder Representations from Transformers* (BERT).

BERT is a multi-layer bidirectional Transformer encoder "designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers" Devlin (2018). The usage of BERT involves two main steps: *pre-training* and *fine-tuning*. As described in Section 3.2.3, several of our experiments focused on using pre-trained Transformer-based models with BERT architecture. We adapted these models to our specific task through fine-tuning. In this Section, however, we will explain the pre-training procedure for BERT, which enables the model to learn general language representations from extensive text corpora. BERT is pre-trained via two unsupervised tasks: *Masked Language Modeling* (MLM) and *Next Sentence Prediction* (NSP).

MLM enables BERT to transition from unidirectional to bidirectional representations. This method involves *masking* certain tokens in the input text and then training the model to predict these masked tokens based on the context provided by the remaining words in the sentence. Unlike traditional language models that predict the next word in a sequence (which is unidirectional), Masked LM allows BERT to utilize context from both directions (left and right) around a masked token. In the BERT paper, approximately 15% of the tokens in each input sequence are masked. "If the i-th token is chosen, we replace the i-th token with (1) the `[MASK]` token 80% of the time (2) a random token 10% of the time (3) the unchanged i-th token 10% of the time[17]. Then, T_i[18] will be used to predict the original token with cross-entropy loss" Devlin (2018). Let's suppose that the unlabeled sentence is "`i like this book`", and during the random masking procedure, we choose the 4th token. The masking procedure can be illustrated with the following example:

- 80% of the time we replace token `book` with `[MASK]`.

  `i like this book` $\longrightarrow$ `i like this [MASK]`

- 10% of the time we replace token `book` with a random word, e.g. `apple`.

  `i like this book` $\longrightarrow$ `i like this apple`

- 10% of the time we keep the 4th token unchanged.

  `i like this book` $\longrightarrow$ `i like this book`

On the other hand, NSP helps BERT learn the relationships between sentences, a capability not captured by language modeling alone but essential for tasks such as

[17] To bias the representation towards the actual observed word.
[18] The final hidden representation for the ith input token.

Question Answering (QA) and Natural Language Inference (NLI). In NSP, the two sentences are separated by a special token `[SEP]` and preceded by the `[CLS]` token used for classification tasks

Input: `[CLS] Sentence A [SEP] Sentence B [SEP].`

For each sentence pair `(Sentence A, Sentence B)` (1) 50% of the time `Sentence B` is the actual sentence that follows `Sentence A` in the text (labeled as `IsNext`), (2) the remaining 50% of the time `Sentence B` is chosen at random from the corpus (labeled ad `NotNext`). For instance,

(1) `Sentence B:` "`he bought a gallon [MASK] milk`" is the true sentence that follows `Sentence A:` "`the man went to [MASK] store`". The `label` is `IsNext`.

(2) `Sentence B:` "`my dog [MASK] barking`" is a random sentence from the corpus. The `label` is `NotNext`.

Sentence-BERT was developed to overcome some of the limitations of BERT in generating fixed-length sentence embeddings and efficiently comparing sentences. BERT was primarily designed for token-level embeddings. "Before Sentence Transformers, the approach to calculating accurate sentence similarity with BERT was to use a *cross-encoder approach*" Pinecone (n.d.b). With this method, both sentences are fed into BERT simultaneously, separated by a `[SEP]` token, and the model outputs a similarity score. Although effective, this method lacks *scalability* as each pair of sentences requires a separate forward pass through the model. Supposing we have a collection of 10,000 sentences, BERT requires approximately 50 million comparisons to find the most similar sentence pair in the corpus ($\sim$ 65 h with modern GPU).

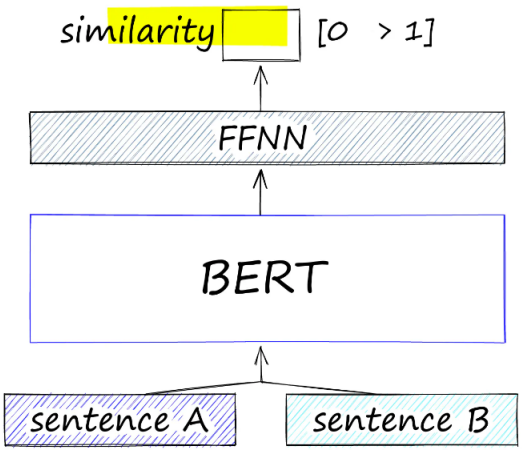


Figure 3.5. BERT cross-encoder architecture. Source: Pinecone (n.d.b).

The development of SBERT by Nils Reimers and Iryna Gurevych addresses the need for efficient generation of high-quality, dense sentence embeddings. SBERT introduces three key innovations. Firstly, unlike traditional BERT, which processes

pairs of sentences at a time, SBERT employs a *Siamese network structure*, where two sentences are passed *independently* through the same BERT model (same weights). Secondly, BERT is followed by a *pooling layer* that transforms token-level embeddings into sentence-level embeddings. By default, SBERT uses *mean pooling*, which averages the token embeddings to generate a sentence embedding. Specifically, the token embeddings matrix M is averaged across the embedding dimension, producing a 768-dimensional vector, where 768 corresponds to the hidden dimension H of BERT models. The jth element of the resulting vector $\mathbf{u}$ is obtained as follows:

$$u_j = \frac{1}{512} \sum_{i=1}^{512} \mathrm{M}_{ij}, \tag{3.37}$$

where 512 is the number N of token-level embeddings outputted by BERT. With the Siamese architecture processing two sentences independently, SBERT produces two distinct sentence embeddings, denoted as $\mathbf{u}$ and $\mathbf{v}$ in Figure 3.6.

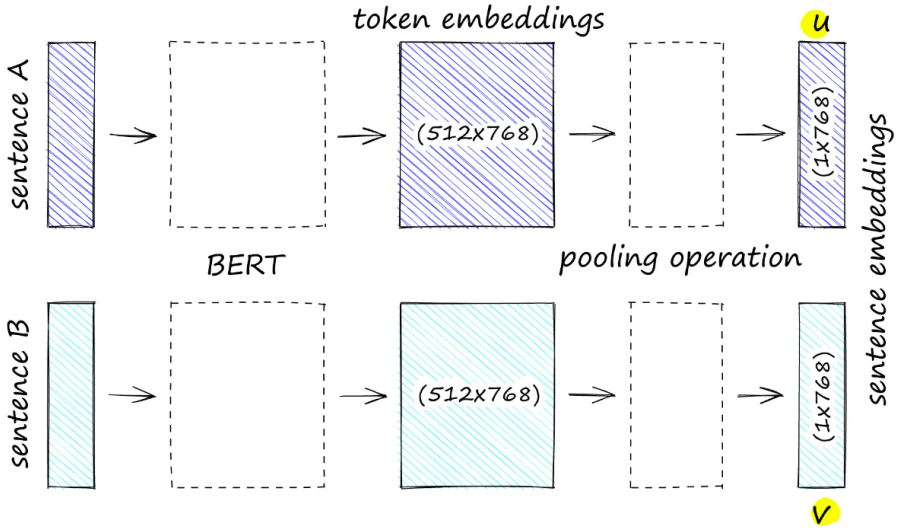


Figure 3.6. SBERT siamese architecture. Source: Pinecone (n.d.b).

The third key innovation is *fine-tuning* the model on sentence similarity tasks to derive semantically meaningful sentence embeddings that can be compared using cosine similarity (Reimers, 2019). SBERT is fine-tuned on a combination of the *SNLI* and *Multi-Genre NLI* sentence pairs (1M), annotated with the labels *contradiction*, *entailment*, and *neutral*[19]. SBERT is fine-tuned as illustrated in Figure 3.7: we concatenate the two sentence embeddings, $\mathbf{u}$ and $\mathbf{v}$, with their element-wise difference $|\mathbf{u}-\mathbf{v}|$, and feed this 3H-dimensional vector into a feedforward neural network (FFNN) with three output nodes. The loss is then computed using a Softmax function. This procedure is described by the following equation:

$$o = \mathrm{Softmax}\left(\mathbf{W}_t \cdot (\mathbf{u}, \mathbf{v}, |\mathbf{u} - \mathbf{v}|)\right), \tag{3.38}$$

where $\mathbf{W}_t$ is a trainable weight matrix of dimensions $(3\mathrm{H} \times k)$, with k being the number of labels (3 in this case).

[19]Natural Language Inference (NLI) is a task that determines whether a *hypothesis* is true

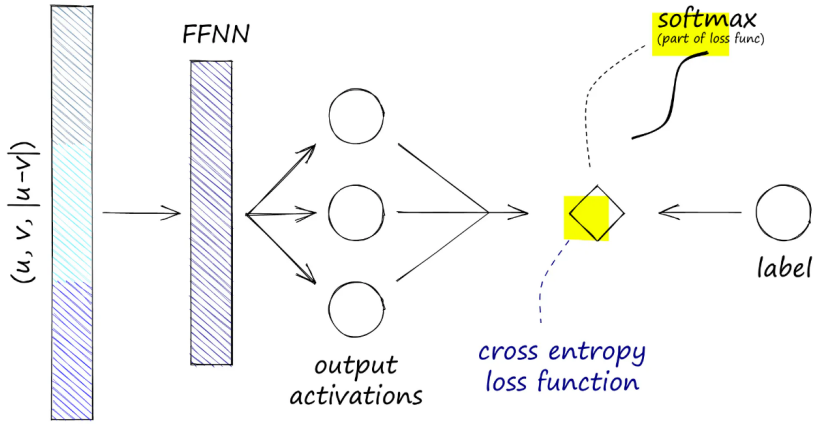


Figure 3.7. SBERT NLI fine-tuning. Source: Pinecone (n.d.b).

Sentence-Transformers

In Section 3.3, we detail the *BERTopic* algorithm designed to identify the main topics in our documents. The initial step consists in representing portioned documents using sentence embeddings generated by pre-trained Sentence Transformers. The original `bert-base-nli-mean-token` model is now deprecated, as newer, more efficient models have since been developed, following the general pipeline outlined in the original paper cited above. For the *English documents* in our corpus, we selected the `all-MiniLM-L6-v2` Sentence Transformer model, which is the default choice in the `bertopic.BERTopic` module for monolingual English texts.

As described by the model card on Hugging Face, the `all-MiniLM-L6-v2` model is obtained by fine-tuning the `nreimers/MiniLM-L6-H384-uncased` model on a 1 billion sentence pairs dataset using a *contrastive learning objective*. Given one of the sentences, e.g. `Sentence A`, belonging to the sentence pair (`Sentence A, Sentence B`), the model aims at predicting `Sentence B` by choosing among a set of sentences that are randomly sampled from the dataset. The cross-entropy loss is used. The model is trained using a batch size of 1024 and an `AdamW` optimizer with a 2e-5 learning rate. Moreover, the sequence length was limited to 128 tokens. As shown in Table 3.1, the final `all-MiniLM-L6-v2` model is composed of 6 encoder layers and has an hidden dimension of 384, for a total of approximately 22 million parameters, like its base model `MiniLM-L6-H384-uncased`. The latter is a monolingual *MiniLM* model, a compressed version of a larger pre-trained Transformer model, obtained through the *Deep Self-Attention Distillation* procedure introduced by Wang W. et al. in the 2019 paper "MINILM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers". This *knowledge distillation* approach aims at compressing a large model (the *teacher* model) into a smaller model (the *student* model), having many fewer parameters but still achieving competitive results on downstream tasks. Deep Self-Attention Distillation involves three main steps: (1) *Self-Attention Distribution Transfer*, (2) *Self-Attention Value-Relation Transfer*, and (3) *Teacher Assistant*. Firstly, the student model is trained to deeply mimic the

(entailment), undetermined (neutral), or false (contradiction) given a *premise*.

self-attention modules of the teacher's last layer by minimizing the KL-divergence[20] $\mathcal{L}_{\text{AT}}$ between the self-attention distributions of the teacher and student. Secondly, the authors proposed "using the relation between the values in the self-attention module to guide the training of the student. The value relation is computed via the multi-head scaled dot-product between values. The KL-divergence $\mathcal{L}_{\text{VR}}$ between the value relation of the teacher and student is used as the training objective" Wang et al. (2020). The final training loss is given by the sum of the self-attention distribution transfer loss and the value-relation transfer loss

$$\mathcal{L} = \mathcal{L}_{\text{AT}} + \mathcal{L}_{\text{VR}}. \tag{3.39}$$

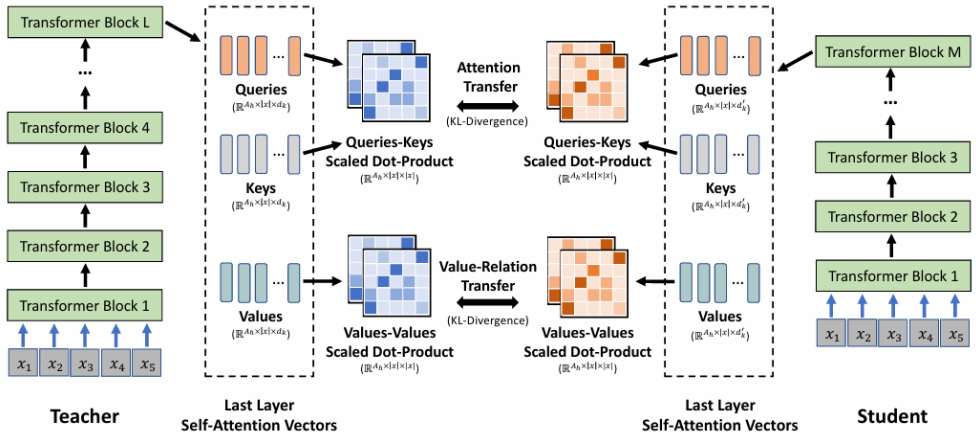


Figure 3.8. Overview of Deep Self-Attention Distillation. Source: Wang et al. (2020).

Lastly, we can use an additional model, acting as a teacher assistant, to further improve the performance of smaller student models. "Assuming the teacher model consists of L-layer Transformer with d_h hidden size, the student model has M-layer Transformer with d'_h hidden size. For smaller students ($\text{M} \leq \frac{1}{2}\text{L}$, $d_h \leq \frac{1}{2}d'_h$), we first distill the teacher into a teacher assistant with L-layer Transformer and d'_h hidden size. The assistant model is then used as the teacher to guide the training of the final student. Introducing a teacher assistant bridges the size gap between teacher and smaller student models, helping to distill Transformer-based pre-trained LMs" Wang et al. (2020). In this context, the `MiniLM-L6-H384-uncased` (22M parameters) model is the student model, while the `bert-base-uncased` model (110M parameters) acts as the teacher model. Given that $\text{L} = 6 \leq \frac{1}{2}\text{M}$ (where $\text{M} = 12$) and $d'_h = 384 \leq \frac{1}{2}d_h$ (where $d_h = 768$), the `MiniLM-L12-H384-uncased` model is used as teacher assistant. This model is a distilled version of the teacher model, with the same number of layers as the teacher and the same hidden dimension as the student.

For the *Italian documents* in the corpus, we selected a multilingual Sentence Transformer called `paraphrase-multilingual-MiniLM-L12-v2`, which is the default choice in `BERTopic` for any language other than English[21]. This model, along with any

[20] The Kullback–Leibler divergence measures the difference between two different probability distributions.

[21] ar, bg, ca, cs, da, de, el, es, et, fa, fi, fr, fr-ca, gl, gu, he, hi, hr, hu, hy, id, it, ja, ka, ko, ku, lt, lv, mk, mn, mr, ms, my, nb, nl, pl, pt, pt, pt-br, ro, ru, sk, sl, sq, sr, sv, th, tr, uk, ur, vi, zh-cn, zh-tw.

other *Multilingual or Cross-lingual Sentence Transformer*, is obtained by adapting a monolingual English model to new languages. The training is based on the idea that semantically similar sentences should be mapped closely in the embedding vector space, regardless of their language. Figure 3.9 displays an example. The Italian sentence "`amo le piante`" should be mapped to the same location in the vector space as the original English sentence "`I love plants`", as the Italian sentence is the exact translation of the English one. The phrase "`mi piacciono le piante`" should be mapped to a close location with respect to the previous two sentences, as they are semantically very similar. On the other hand, the Italian sentence "`ho un cane arancione`" should not be close to "`amo le piante`" or "`mi piacciono le piante`" just because they are written in the same language, as they have different meanings. Hence, "`mi piacciono le piante`" should be closer to "`I love plants`" than to "`ho un cane arancione`".

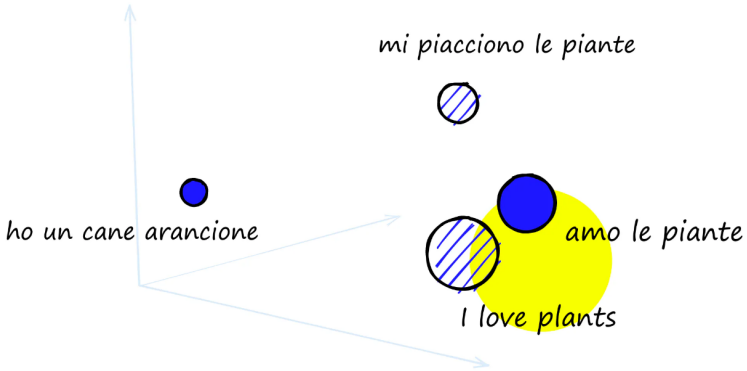


Figure 3.9. A multilingual model will map sentences from different languages into the same vector space. Source: Pinecone (n.d.a).

We can train a multilingual Sentence Transformer via *Multilingual Knowledge Distillation*, an algorithm introduced by Nils Reimers and Iryna Gurevych in their 2020 paper titled "Making Monolingual Sentence Embeddings Multilingual using Knowledge Distillation". We require a monolingual Sentence Transformer, i.e. the *teacher* model M, and a cross-lingual Transformer model, i.e. the *student* model $\hat{\mathrm{M}}$. Suppose we are given a dataset $\mathcal{D}$ of the type

$$\mathcal{D} = \Big\{(s_1, t_1), (s_2, t_2), \ldots, (s_n, t_n)\Big\},$$

where s_i is a sentence in one of the *source languages* (English in our case) and t_i is the translation of s_i in one of the *target languages*. We train model $\hat{\mathrm{M}}$ such that:

(1) $\hat{\mathrm{M}}(t_i) \approx \mathrm{M}(s_i)$ meaning that "vector spaces are aligned across languages, i.e. identical sentences in different languages are close" Reimers and Gurevych (2020).

(2) $\hat{\mathrm{M}}(s_i) \approx \mathrm{M}(s_i)$ meaning that "vector space properties in the original source language from the teacher model M are adopted and transferred to other languages" Reimers and Gurevych (2020).

For a given mini-batch $\mathcal{B}$, we minimize the mean-squared loss in equation (3.40). This procedure is illustrated in Figure 3.10.

$$\frac{1}{|\mathcal{B}|}\sum_{j\in\mathcal{B}}\left[\left(\hat{\mathrm{M}}(s_j)-\mathrm{M}(s_j)\right)^2+\left(\hat{\mathrm{M}}(t_j)-\mathrm{M}(s_j)\right)^2\right] \tag{3.40}$$

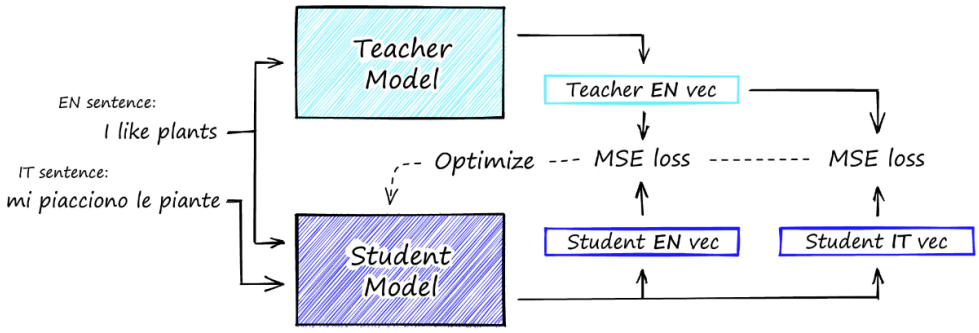


Figure 3.10. Overview of Multilingual Knowledge Distillation. Source: Pinecone (n.d.a).

Concerning the `paraphrase-multilingual-MiniLM-L12-v2` Sentence Transformer, the teacher is `paraphrase-MiniLM-L12-v2` (118M parameters), a SBERT-paraphrase model, and the student is `Multilingual-MiniLM-L12-H384` (117M parameters), a MiniLM multilingual model. The latter is a compact version of the `xlm-roberta-base` cross-lingual model (280M parameters), obtained through a Deep Self-Attention Distillation procedure as explained above. The XLM-roBERTa model is further explained in Section 3.2.3, where we tackle document classification methods. The multilingual Sentence Transformer inherited the architecture of the teacher. In fact, as shown in Table 3.1, `paraphrase-MiniLM-L12-v2` has 12 layers, an embedding dimension of 384 and a maximum input sequence length equal to 128.

All models referenced in this section are summarized in table 3.1. Specifically, the table details the following: L represents the number of layers, H is the hidden dimension, N indicates the maximum input sequence length, and #Params refers to the number of the embedding and Transformer model parameters. The two Sentence Transformers used in `BERTopic` are highlighted in bold.

Model	L	H	N	#Params
`bert-base-nli-mean-tokens (SBERT)`	12	768	128	110M
`all-MiniLM-L6-v2`	**6**	**384**	**256**	**23M**
`MiniLM-L6-H384-uncased`	6	384	512	22M
`bert-base-uncased`	12	768	512	110M
`paraphrase-multilingual-MiniLM-L12-v2`	**12**	**384**	**128**	**118M**
`Multilingual-MiniLM-L12-H384`	12	384	512	117M
`paraphrase-MiniLM-L12-v2`	12	384	128	33M
`xlm-roberta-base`	12	768	512	280M

Table 3.1. Transformer and Sentence Transformer Model Architectures: L is the number of layers, H is the hidden dimension, N is the maximum input sequence length, and #Params is the number of the embedding and Transformer model parameters.

3.2 Classification Methods

After embedding our documents, we move on to one of the central objectives of this work: *authorship classification*. The goal is to categorize each speech into one of four classes associated with the Governors introduced in Chapter 2: Ciampi, Fazio, Draghi, and Visco.

Classification is a Machine Learning problem, specifically belonging to a key branch called *Supervised Machine Learning*. In supervised learning, models are trained using *labeled data* of the type

$$\mathcal{D} = \Big\{(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), \ldots, (\mathbf{x_n}, y_n)\Big\},$$

where each data point $\mathbf{x_i} \in \mathbb{R}^p$ is paired with a corresponding label $y_i \in \mathbb{R}$. During prediction, the model assigns a class $\hat{y}_i$ to each document $\mathbf{x_i}$, and these predicted labels are compared to the true labels y_i to evaluate performance.

For our classification task, we used three techniques: Support Vector Machines (SVM), Multiple Logistic Regression, and Transformer-based models. These methods differ in terms of complexity and scalability. Notably, Transformer-based models stand out from the other two approaches as they do not require pre-computed text embeddings via TF-IDF or Doc2Vec.

3.2.1 SVM

Support Vector Machines (SVM) are supervised machine learning algorithms designed to classify data points by finding the optimal hyperplane that best separates the classes in a dataset. Although SVM was originally intended for binary classification problems, it can be extended to handle multi-class problems as well. Let's first focus on binary classification, specifically on the simplest case: linearly separable data.

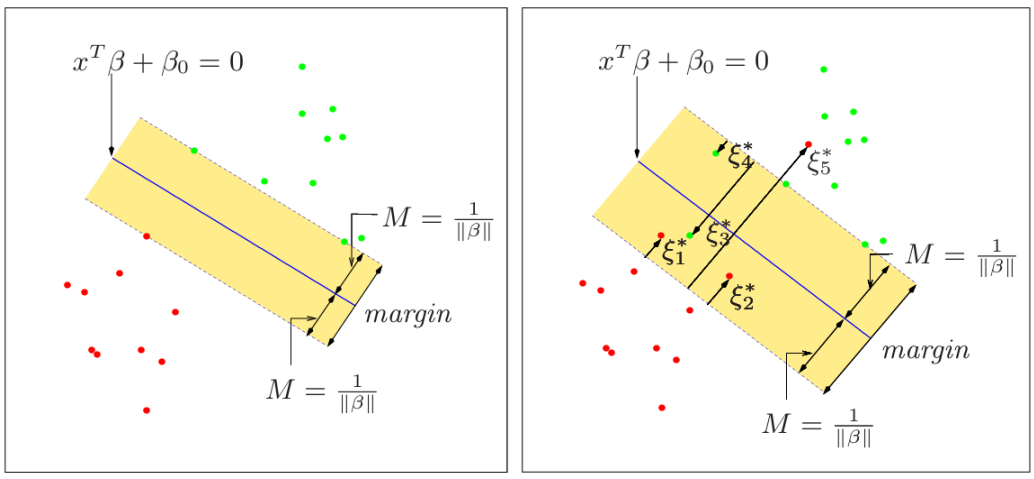


Figure 3.11. Support Vector Classifiers. The left panel shows the separable case. The right panel shows the non-separable (overlap) case. Source: Hastie et al. (2009).

The Separable Case

Suppose we have a labeled dataset $\mathcal{D} = \{(\mathbf{x_i}, y_i)\}_{i=1}^n$ where $\mathbf{x_i} \in \mathbb{R}^p$ represents the

feature vector and $y_i \in \{-1, 1\}$ is the class label associated with $\mathbf{x_i}$. The data is linearly separable, meaning that there exists a hyperplane

$$\{\mathbf{x} : f(\mathbf{x}) = \mathbf{w}^\intercal \mathbf{x} + b = 0\} \tag{3.41}$$

that shatters[22] the input data points $\mathbf{x_i}$, achieving zero classification error. The goal is to find the "best" weights $\hat{w}$ and bias $\hat{b}$ such that

$$y_i \cdot \left(\hat{\mathbf{w}}^\intercal \mathbf{x_i} + \hat{b}\right) \geq 1, \quad i = 1, \ldots, n \tag{3.42}$$

meaning that if the i-th data point belongs to class 1, it should lie on the positive side of the separating hyperplane, and if it belongs to class -1, it should lie on the negative side of the hyperplane. The aim is to maximize the *margin*, highlighted in yellow in Figure 3.11, between the hyperplane and the nearest data points from each class, known as *support vectors*. Maximizing this margin improves the model's generalization capability by increasing its robustness to small perturbations in the data. The margin is defined as $\frac{2}{\|\mathbf{w}\|}$, and the corresponding *hard-margin* optimization problem is formulated as

$$\min_{\mathbf{w},b} \frac{1}{2}\|w\|^2 \tag{3.43}$$

subject to

$$y_i \cdot \left(\hat{\mathbf{w}}^\intercal \mathbf{x_i} + \hat{b}\right) \geq 1, \quad i = 1, \ldots, n. \tag{3.44}$$

Here, minimizing the squared norm $\|\mathbf{w}\|^2$ directly maximizes the margin. The constraint ensures that all data points are correctly classified and do not violate the margin constraint, i.e. they lie on the correct side of the hyperplane and outside the margin. Starting from the primal problem, we derive the dual problem, which often results in a sparse solution, as many Lagrange multipliers α_i are zero (only a small number of data points have non-zero α_i, i.e. the support vectors that define the decision boundary). The first step involves constructing the Lagrangian function[23] for this problem, defined as

$$\mathcal{L}(\mathbf{w}, b, \alpha) = \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{n} \alpha_i \left(1 - y_i \left(\mathbf{w}^\intercal \mathbf{x_i} + b\right)\right), \tag{3.45}$$

where $\alpha_i \geq 0, i = 1, \ldots, n$ are the Lagrange multipliers associated with the inequality constraints. To find the dual function, we need to minimize the Lagrangian with

[22] A set of data points is said to be shattered, when there exists a learning function $f(\cdot)$ that perfectly separates the two classes in the p-dimensional space defined by the data points.

[23] Suppose we want to minimize $f(\mathbf{x})$ w.r.t. $\mathbf{x}$, subject to equality constraints of type $g_i(\mathbf{x}) = 0, i = 1, \ldots, m$ and inequality constraints of type $h_j(\mathbf{x}) \leq 0, j = 1, \ldots, p$. For each equality constraint, we introduce a Lagrange multiplier λ_i, and for each inequality constraint, we introduce a Lagrange multiplier $\mu_j \geq 0$. The Lagrangian is given by

$$\mathcal{L}(\mathbf{x}, \lambda, \mu) = f(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i g_i(\mathbf{x}) + \sum_{j=1}^{p} \mu_j h_j(\mathbf{x})$$

respect to $\mathbf{w}$ and b. First, we compute the partial derivative of $\mathcal{L}(\mathbf{w}, b, \alpha)$ w.r.t. $\mathbf{w}$ and set it to zero

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{n} \alpha_i y_i \mathbf{x_i} = 0 \implies \mathbf{w}^* = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x_i}. \tag{3.46}$$

Secondly, we compute the partial derivative of $\mathcal{L}(\mathbf{w}, b, \alpha)$ w.r.t. b and set it to zero

$$\frac{\partial \mathcal{L}}{\partial b} = \sum_{i=1}^{n} \alpha_i y_i = 0. \tag{3.47}$$

We substitute $\mathbf{w}^*$ back into $\mathcal{L}$ and obtain

$$\begin{aligned} \mathcal{L}(\alpha) &= \frac{1}{2} \left(\sum_{i=1}^{n} \alpha_i y_i \mathbf{x_i} \right)^{\mathsf{T}} \left(\sum_{i=1}^{n} \alpha_i y_i \mathbf{x_i} \right) + \sum_{i=1}^{n} \alpha_i \left(1 - y_i \left(\sum_{i=1}^{n} \alpha_i y_i \mathbf{x_i}^{\mathsf{T}} \mathbf{x_i} \right) \right) \\ &= \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x_i}^{\mathsf{T}} \mathbf{x_j}. \end{aligned} \tag{3.48}$$

Finally, we can write the dual problem:

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x_i}^{\mathsf{T}} \mathbf{x_j} \tag{3.49}$$

subject to

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{3.50}$$

$$\alpha_i \geq 0, \ i = 1, \ldots, n. \tag{3.51}$$

The dual problem is a convex optimization problem, as the objective function, which we aim to maximize, is concave[24], and the feasible region defined by the constraints forms a convex set. Solving the dual problem (e.g. by exploiting solvers for convex optimization problems) yields the optimal values $\{\hat{\alpha}_i\}_{i=1}^{n}$. These Lagrange multipliers can then be used to compute the optimal weights $\hat{\mathbf{w}}$ and bias $\hat{b}$ in the following way

$$\hat{\mathbf{w}} = \sum_{i=1}^{n} \hat{\alpha}_i y_i \mathbf{x_i} \tag{3.52}$$

$$\hat{b}_s = y_s - \sum_{i=1}^{n} \alpha_i y_i \mathbf{x_i}^{\mathsf{T}} \mathbf{x_s}, \tag{3.53}$$

where $\hat{b}_s$ is the bias associated with the s-th support vector. We can average b_s across all support vectors for more stability

$$\hat{b} = \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \hat{b}_s. \tag{3.54}$$

[24]This function is concave because it is a quadratic function in terms of α with a negative definite Hessian matrix.

Once the optimal separating hyperplane is found, we can classify new data points by using the *decision function*

$$\hat{y}_i = \text{sign}\left(\hat{\mathbf{w}}^\intercal \mathbf{x_i} + \hat{b}\right), \tag{3.55}$$

such that $\hat{y}_i = 1$ if the point is classified as positive and $\hat{y}_i = -1$ if it is classified as negative.

The Non-separable Case

In practice, data is rarely perfectly separable, and there might be some *overlap* between classes. To address this, SVM introduces a *soft-margin* approach that allows some misclassifications. "In this case, one may want to separate the training set with a minimal number of errors. To express this formally, let us introduce some non-negative variables $\xi_i \geq 0, i = 1, \ldots, n$" Cortes (1995). As shown in the right panel of Figure 3.11, slack variables measure the extent to which each data point violates the margin:

- If $\xi_i = 0$, the i-th data point is correctly classified and lies outside (or exactly on) the margin boundary. Hence, $y_i \cdot (\mathbf{w}^\intercal \mathbf{x_i} + b) \geq 1$.
- if $0 < \xi_i \leq 1$, the i-th data point is correctly classified but falls within the margin area, hence violating the ideal separation (e.g. look at points $1, 2$ and 4 in the right panel in Figure 3.11).
- if $\xi_i > 1$, the i-th data point is incorrectly classified, i.e. it lies on the wrong side of the decision boundary (e.g. look at points 3 and 5 in the right panel in Figure 3.11).

We then need to find a way to maximize the margin while allowing as few misclassifications as possible. This can be expressed as the following constrained optimization problem

$$\min_{\mathbf{w},b,\xi} \frac{1}{2}\|\mathbf{w}\|^2 + \mathrm{C}\sum_{i=1}^{n} \xi_i \tag{3.56}$$

subject to

$$y_i \cdot (\mathbf{w}^\intercal \mathbf{x_i} + b) \geq 1 - \xi_i, \ i = 1, \ldots, n \tag{3.57}$$

$$\xi_i \geq 0, \ i = 1, \ldots, n, \tag{3.58}$$

with C being the regularization parameter that controls the trade-off between maximizing the margin and allowing misclassifications. A higher C gives a smaller margin with fewer misclassifications (risk of overfitting), while a lower C allows a larger margin with more misclassifications. This formulation is also a convex optimization problem. As for the hard-margin case, we can derive the dual problem from the primal problem. The first step consists in building the Lagrangian function, defined as

$$\mathcal{L}(\mathbf{w}, b, \alpha, \beta, \xi) = \frac{1}{2}\|\mathbf{w}\|^2 + \mathrm{C}\sum_{i=1}^{n} \xi_i + \sum_{i=1}^{n} \alpha_i \left(1 - \xi_i - y_i \left(\mathbf{w}^\intercal \mathbf{x_i} + b\right)\right) - \sum_{i=1}^{n} \beta_i \xi_i, \tag{3.59}$$

where $\alpha_i \geq 0, \beta_i \geq 0, i = 1, \ldots, n$ are the Lagrange multipliers associated with the inequality constraints. Then, we compute the partial derivatives of $\mathcal{L}(\mathbf{w}, b, \alpha, \beta, \xi)$

w.r.t. $\mathbf{w}, b, \xi$. The partial derivatives of $\mathcal{L}$ w.r.t. $\mathbf{w}$ and b are are already given in equations (3.46) and (3.47). Then we calculate the partial derivative w.r.t. ξ_i and set it to zero

$$\frac{\partial \mathcal{L}}{\partial \xi_i} = \mathrm{C} - \alpha_i - \beta_i = 0 \implies \alpha_i = \mathrm{C} - \beta_i. \tag{3.60}$$

By substituting $\mathbf{w}^*$ and α_i back into $\mathcal{L}$ we get the following dual problem:

$$\mathcal{L}(\alpha, \beta, \xi) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x_i}^\intercal \mathbf{x_j} \tag{3.61}$$

subject to

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{3.62}$$

$$0 \leq \alpha_i \leq \mathrm{C}, \ \beta_i \geq 0, i = 1, \ldots, n. \tag{3.63}$$

Solving this convex optimization problem yields the optimal values for the Lagrange multipliers $\{\hat{\alpha}_i\}_{i=1}^n, \{\hat{\beta}_i\}_{i=1}^n$, and, in turn, the optimal weights $\hat{\mathbf{w}}$ and bias $\hat{b}$.

Non-linear SVM: the Kernel Trick

In most real-world problems, where the data is not linearly separable, using a linear SVM model can be limiting. Hence, we can introduce a nonlinear version of SVM. The fundamental aim of nonlinear SVM is to project the input data into a higher-dimensional feature space and then establish a linear classifier within this newly defined space. The input training vectors $\mathbf{x_i}$ are transformed via a kernel function $\phi(\mathbf{x_i})$ with $i = 1, \ldots, n$. The kernel function performs a mapping $\phi(\mathbf{x_i}) : \mathbb{R}^n \to \mathbb{R}^m$, where $m \gg n$. In this higher-dimensional space, the decision boundary is a hyperplane, but when we project it back to the original space, the decision surface is no longer linear (see Figure 3.12). In the `sklearn.svm.SVC` module, the `kernel` parameter can be set to values other than `linear`, such as `poly` and `rbf`:

- The *polynomial kernel* maps the data into a higher-dimensional space using polynomial functions. It is defined as

  $$k(\mathbf{x_i}, \mathbf{x_j}) = (\mathbf{x_i}^\intercal \mathbf{x_j} + c)^d,$$

  where c is a constant and d is the degree of the polynomial.

- The Radial-Basis-Function (RBF) kernel maps the data into an infinite-dimensional space. It is defined as

  $$k(\mathbf{x_i}, \mathbf{x_j}) = \exp\left(-\frac{\|\mathbf{x_i} - \mathbf{x_j}\|^2}{2\sigma^2}\right),$$

  where σ is a hyperparameter that controls the width of the Gaussian function.

We can solve the same dual problem of linear SVM except we replace $\mathbf{x_i}^\intercal \mathbf{x_j}$ with $\mathrm{K}_{ij} = \phi(\mathbf{x_i})^\intercal \phi(\mathbf{x_j}) = k(\mathbf{x_i}, \mathbf{x_j})$, where $\mathrm{K}(\mathbf{x_1}, \ldots, \mathbf{x_n})$ is the kernel matrix.

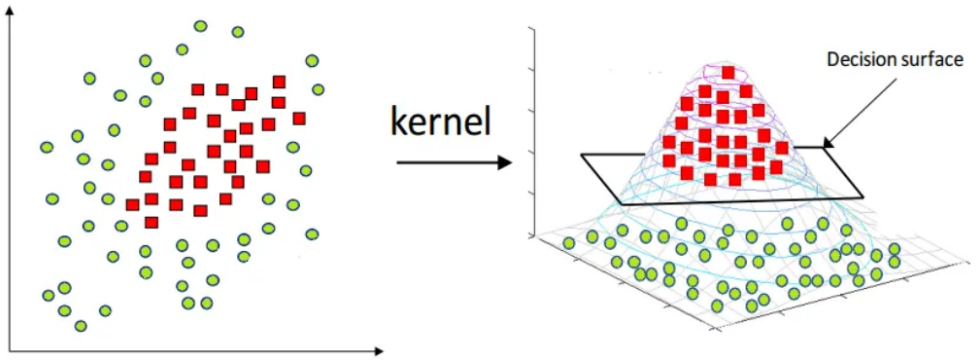


Figure 3.12. Kernel Trick. Source: Zhang (2018).

Multiclass SVM: One-Versus-Rest

So far, we focused on binary SVMs. However, in our work, we also need to address multiclass classification problems. We can extend the binary SVM formulation for multiclass tasks through strategies like *One-Versus-One* (`ovo`) and *One-Versus-Rest* (`ovr`). Given that the `decision_function_shape` parameter of the `sklearn.svm.SVC` class is set to `ovr`, we will explain how this approach works in steps:

- We create a separate binary SVM classifier for each class $i = 1, \ldots, k$.
- Each classifier is trained on a dataset where the instances belonging to class i are labeled as the positive class (label 1), while all other instances are labeled as the negative class (label -1).
- To classify a new instance, each classifier computes a score $f_i(\mathbf{x})$ indicating how strongly it belongs to class i versus all other classes.
- The class assigned to data point $\mathbf{x}$ is the one that maximizes the score:

$$\hat{y} = \text{argmax}_i \; f_i(\mathbf{x})$$

3.2.2 Multiple Logistic Regression

Multiple logistic regression extends multiple linear regression to handle categorical outcome variables, particularly when the outcome is classified into distinct classes. Instead of directly modeling Y, which is categorical, we model the probability that Y belongs to a specific class, given a feature vector $\mathbf{x} = (x_1, x_2, \ldots, x_p) \in \mathbb{R}^p$.

For the *binary classification* case, the goal is to model the probability that Y, taking values in $\{0, 1\}$, equals 1 given feature vector $\mathbf{x}$, which we write as

$$\Pr(\mathrm{Y} = 1 \mid \mathbf{x}) = p(\mathbf{x}) = \frac{e^{\beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p}}{1 + e^{\beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p}}, \tag{3.64}$$

with β_0 being the intercept coefficient. This expression is obtained by applying the sigmoid function to $z = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p$, which results in an S-shaped curve. This curve represents the *decision boundary* between classes. The logistic function

compresses the input z into the range $(0, 1)$, providing a probability estimate. The decision rule for classifying an observation is based on a threshold, often set at 0.5.

$$\hat{y} = \begin{cases} 1 & \text{if } p(\mathbf{x}) > 0.5, \\ 0 & \text{otherwise.} \end{cases} \tag{3.65}$$

To better understand the relationship between z and the probability, we express the model using the *logit transformation*.

$$\log\left(\frac{p(\mathbf{x})}{1-p(\mathbf{x})}\right) = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p \tag{3.66}$$

where $\frac{p(\mathbf{x})}{1-p(\mathbf{x})}$ is the *odds* that $\mathrm{Y} = 1$, and taking the logarithm linearizes the relationship. Once this logistic regression model is defined, the next step is to estimate the *coefficients* $\beta_0, \beta_1, \ldots, \beta_p$, to find the best fitting line. These coefficients are estimated by minimizing the *cross-entropy* (log-loss) function:

$$\mathcal{L}(\beta_0, \beta_1, \ldots, \beta_p) = -\sum_{i=1}^{n} \Big[y_i \cdot \log(p(\mathbf{x}_i)) + (1 - y_i) \cdot \log(1 - p(\mathbf{x}_i)) \Big]. \tag{3.67}$$

To prevent overfitting, we can introduce ℓ_2*-regularization* (Ridge) by adding a penalty term that penalizes large β_j coefficients.

$$\mathcal{L}_{\mathrm{REG}}(\beta_0, \beta_1, \ldots, \beta_p) = \mathcal{L}(\beta_0, \beta_1, \ldots, \beta_p) + \lambda \sum_{j=1}^{p} \beta_j^2, \tag{3.68}$$

where λ controls the regularization strength. The optimization of $\mathcal{L}_{\mathrm{REG}}(\beta_0, \beta_1, \ldots, \beta_p)$ with respect to the beta coefficients can be done using gradient descent. Hence, we compute the partial derivative of the loss function w.r.t. each coefficient β_j and iteratively update the coefficients until convergence.

$$\beta_j^{(\text{new})} \leftarrow \beta_j^{(\text{old})} - \eta \cdot \frac{\partial \mathcal{L}_{\mathrm{REG}}(\beta_0, \beta_1, \ldots, \beta_p)}{\partial \beta_j}, \tag{3.69}$$

with $\eta > 0$ being the learning rate. The optimization procedure stops when a stopping criterion is met, e.g. when the gradient's magnitude $|\nabla \mathcal{L}_{\mathrm{REG}}(\cdot)|$ falls below a predefined tolerance[25] or a maximum number of iterations is reached.

For multi-class problems where Y can take more than two values, e.g. $\mathrm{Y} \in \{1, 2, \ldots, \mathrm{K}\}$, we generalize multiple logistic regression to *Multinomial Logistic Regression*. In this case, the model estimates the probability that Y belongs to each class k given the feature vector $\mathbf{x}$. We can rewrite equation (3.64) as

$$\Pr(\mathrm{Y} = k \mid \mathbf{x}) = \frac{e^{\beta_{0k} + \beta_{1k} x_1 + \cdots + \beta_{pk} x_p}}{\sum_{t=1}^{K} e^{\beta_{0t} + \beta_{1t} x_1 + \cdots + \beta_{pt} x_p}}, \tag{3.70}$$

where the denominator is a normalization term ensuring that the probabilities sum to 1. In Section 4.1 we use the `sklearn.linear_model.LogisticRegression` class to fit logistic regression models, with the `multi_class` parameter set to "`multinomial`".

[25] Generally set between 10^{-3} and 10^{-6}.

This configuration allows the model to directly minimize the multinomial loss rather than using the One-vs-Rest (OvR) approach, where a binary classifier is trained for each class, treating one random class as the reference class. The multinomial loss being optimized is given by

$$\mathcal{L}_{\text{REG}}(\beta_{00}, \beta_{10}, \ldots, \beta_{p\text{K}}) = -\sum_{i=1}^{n}\sum_{k=1}^{\text{K}} y_{ik}\log(p(\mathbf{x}_i)) + \lambda\sum_{j=1}^{p}\sum_{k=1}^{\text{K}}\beta_{jk}^2. \tag{3.71}$$

To optimize the multinomial logistic regression model, gradient descent is employed. The update rule for the coefficients is given by

$$\beta_{jk}^{(\text{new})} \leftarrow \beta_{jk}^{(\text{old})} - \eta \cdot \frac{\partial \mathcal{L}_{\text{REG}}(\beta_{00}, \beta_{10}, \ldots, \beta_{p\text{K}})}{\partial \beta_{jk}}. \tag{3.72}$$

3.2.3 Fine-Tuning Pre-trained Transformer-based Models

Training a deep neural network typically requires a large amount of data. When such extensive datasets are unavailable, we can leverage pre-trained models (e.g. Transformers) and adapt them for our specific task. This approach is known as *Transfer Learning*. As the name suggests, rather than starting from scratch, Transfer Learning involves transferring the knowledge acquired by a large, pre-trained model and applying it to a different but related problem. As shown in Figure 3.13, the process begins with a pre-trained model $\text{M}_{\mathcal{D}}$ (e.g. BERT), which has already been trained on a large general-purpose dataset $\mathcal{D}$ (e.g. as described in Section 3.1.3). We can then adapt this model to a new task t using a task-specific dataset $\mathcal{D}_t$, creating a new model $\text{M}_{\mathcal{D}_t}$. This process can be repeated multiple times across different tasks and datasets.

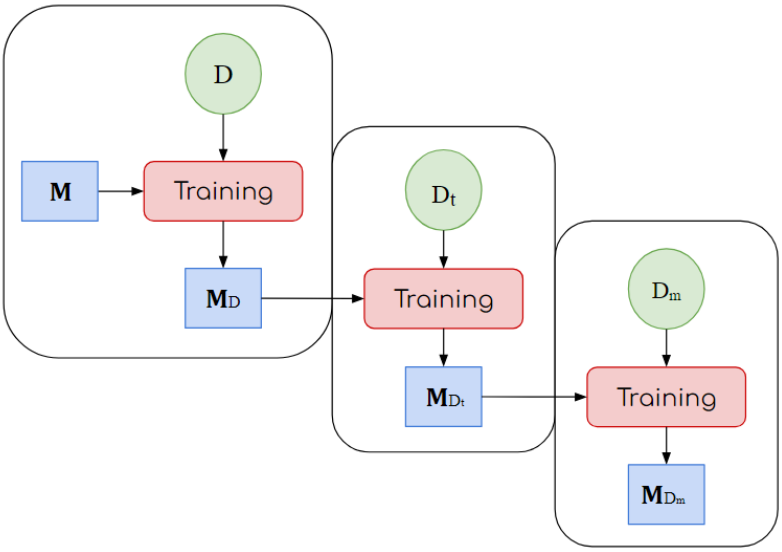


Figure 3.13. Transfer Learning.

The simplest and most common way to perform Transfer Learning is through *Fine-Tuning*. Fine-tuning adapts a pre-trained model to a new task by *re-training some of its weights* using a task-specific dataset—in our case, the corpus of Governor speeches—to improve its performance on that specific task. Typically, the final layer of the model is replaced with a new layer that suits the new task. For example, we might add a classification head with randomly initialized weights that generate probabilities for the target classes.

A Transformer model is composed of multiple layers, each capturing different levels of information. The earlier layers focus on extracting general, low-level features, while the later layers extract more task-specific, high-level features. Thus, during fine-tuning, it is common to "freeze" the earlier layers, leaving only the later layers trainable (as shown in Figure 3.14). This allows the model to adapt to the new task without losing the general knowledge it gained during pre-training. Fine-tuning is usually done with a low learning rate to avoid wiping out all the pre-trained knowledge.

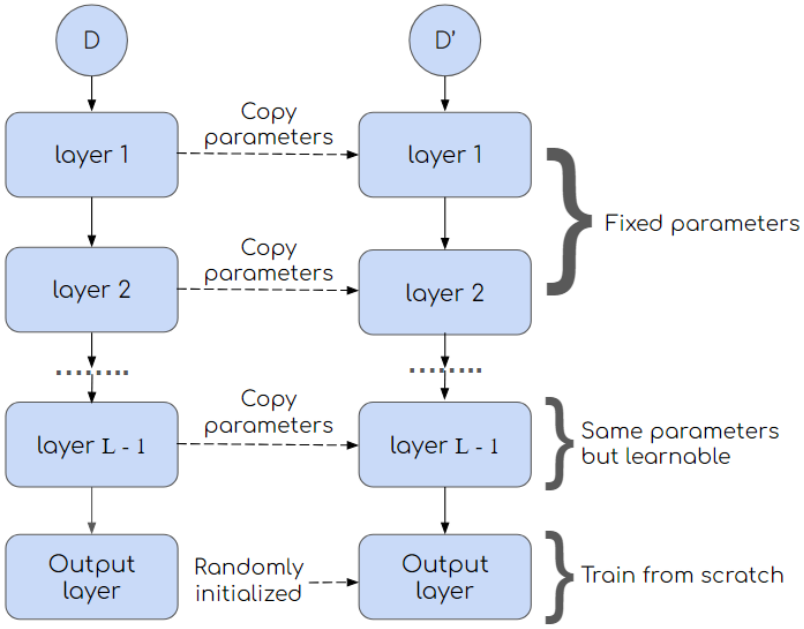


Figure 3.14. Fine Tuning.

For this work, we fine-tuned three different pre-trained models on three different datasets. For the English dataset, we used the RoBERTa-base model (`FacebookAI/roberta-base`). For the Italian dataset, we fine-tuned a RoBERTa model specifically designed for the Italian language (`osiria/roberta-base-italian`). Finally, for the combined English-Italian dataset, we used the XLM-RoBERTa model (`XLM-RoBERTa`), which is multilingual. These models are summarized in Table 3.2.

RoBERTa (*Robustly optimized BERT approach*) is an improved version of BERT, sharing the same architecture but trained with several improvements:

(1) Extended training: RoBERTa was trained for significantly more steps (125M or 31M steps vs. BERT's 1M steps), using larger batch sizes (2000 or 8000 vs.

256) and over more data (160GB, about 10 times more than for BERT).

(2) Removal of the Next Sentence Prediction (NSP) objective: RoBERTa was trained exclusively on the Masked Language Modeling (MLM) objective, as its authors found that eliminating NSP either matched or slightly improved performance on downstream tasks.

(3) Training on longer sequences: RoBERTa was trained with sequences of up to 512 tokens, while BERT was trained mostly on shorter sequences of 128 tokens.

(4) Dynamic masking: Unlike BERT's static masking (where mask tokens are fixed during data preprocessing), RoBERTa uses dynamic masking, meaning that sequences are masked differently throughout the training process. Each training sequence is masked in 10 different ways over the 40 epochs of training.

XLM-RoBERTa is a multilingual extension of RoBERTa that has been proven to outperform other multilingual models, like mBERT, on various cross-lingual tasks. It was trained using the MLM objective over massive monolingual datasets spanning 100 languages, with a vocabulary size of 250K tokens. Unlike earlier XLM models (Conneau and Lample, 2019), which incorporated a Translation Language Modeling (TLM) objective (an extension of BERT's MLM to multiple language inputs), XLM-RoBERTa is trained purely on monolingual data. Due to its multilingual nature, XLM-RoBERTa has over twice as many parameters as RoBERTa-base (270M vs. 125M).

As described in its GitHub model card[26], `osiria/roberta-base-italian` is a variant of RoBERTa fine-tuned for the Italian language. It was developed by specializing XLM-RoBERTa in Italian by adapting the embedding layer.

Model	L	H	N	#Params	Vocab
`FacebookAI/roberta-base`	12	768	512	125M	50K
`osiria/roberta-base-italian`	12	768	512	125M	51K
`FacebookAI/xlm-roberta-base`	12	768	512	270M	250K

Table 3.2. Transformer Model Architectures: L is the number of layers, H is the hidden dimension, N is the maximum input sequence length, #Params is the number of the embedding and Transformer model parameters, and Vocab is the vocabulary size of the model.

[26] https://huggingface.co/osiria/roberta-base-italian.

3.3 BERTopic

BERTopic is a *topic modeling* technique proposed by Maarten Grootendorst in 2022. This algorithm is designed to extract the main topics within our document corpus and analyze how these topics evolve over time. BERTopic generates topic representations through four main steps.

First, documents are converted into numerical representations. BERTopic recommends using Sentence Transformers to obtain semantically meaningful *document embeddings*. Since Sentence Transformers have a maximum token limit, longer documents must be split into shorter chunks.

Then, these chunks are clustered, with each cluster representing a topic. However, clustering high-dimensional data is not ideal due to the curse of dimensionality[27]. To address this, we can reduce the dimensionality of the document embeddings using *dimensionality-reduction* techniques. Specifically, we employed UMAP, a graph-based algorithm that can keep some of the local and global structure of the corpus when reducing its dimensionality.

After dimensionality reduction, the document chunks are clustered using HDBSCAN, a state-of-the-art density-based *clustering algorithm*. HDBSCAN offers several advantages over traditional clustering methods such as K-means. Firstly, the number of clusters does not need to be specified beforehand. Secondly, it can identify clusters of varying shapes, sizes, and densities. Thirdly, it allows some data points to be treated as *noise*. This means that if a particular document chunk does not belong to any cluster (topic), it can be excluded, reducing the risk of creating confusing topics.

Finally, *topic representations* are extracted from the clusters using a custom class-based variation of TF-IDF, resulting in easily interpretable descriptions of the identified topics.

3.3.1 UMAP

Dimensionality reduction algorithms generally fall into one of two main categories: matrix factorization methods like PCA or graph-based approaches like t-SNE. UMAP (*Uniform Manifold Approximation and Projection*) belongs to the latter category and is a graph-layout algorithm that "can be described in two phases. In the first phase, a particular weighted k-neighbour graph is constructed. In the second phase, a low dimensional layout of this graph is computed" McInnes et al. (2018). This process relies on the assumption that the *Manifold Hypothesis* holds, saying that high-dimensional data often lies on a low-dimensional manifold embedded in a higher-dimensional space.

The first step involves constructing a *weighted k-nearest neighbor graph*[28] $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ of the input data $\mathcal{X} = \{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_p}\}$, where each feature vector $\mathbf{x_i}$ is

[27] In high-dimensional spaces, distances can become less informative because the relative difference between the minimum and maximum distances tends to shrink, making it difficult to distinguish between close and distant points.

[28] A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a mathematical structure used to model relationships between objects. It consists of a set of vertices $\mathcal{V}$ connected by edges contained in a set $\mathcal{E}$. A graph is said to be directed if the edges have a direction otherwise it is undirected. A graph can also be weighted when the edges carry different weights.

embedded in a high-dimensional space $\mathbb{R}^n$. Given a distance metric (or dissimilarity measure) d, we start by computing the k-nearest neighbors $\{\mathbf{x_{i1}}, \ldots, \mathbf{x_{ik}}\}$ for each data point $\mathbf{x_i}$. Then, for each $\mathbf{x_i}$, we compute ρ_i, defined as the distance to the closest nearest neighbor

$$\rho_i = \min \left\{ d(\mathbf{x_i}, \mathbf{x_j}) \mid 1 \leq j \leq k, d(\mathbf{x_i}, \mathbf{x_j}) > 0 \right\}. \tag{3.73}$$

Next, we calculate the normalization factor σ_i as the value satisfying

$$\sum_{j=1}^{k} \exp\left(\frac{-\max(0, d(\mathbf{x_i}, \mathbf{x_j}) - \rho_i)}{\sigma_i} \right) = \log_2(k). \tag{3.74}$$

This allows us to define a weighted, directed graph $\bar{\mathcal{G}} = \left(\bar{\mathcal{V}}, \bar{\mathcal{E}}, w\right)$. The vertices $\bar{\mathcal{V}}$ of $\bar{\mathcal{G}}$ correspond to the set $\mathcal{X}$. The set of directed edges is given by $\bar{\mathcal{E}} = \{(\mathbf{x_i}, \mathbf{x_j}) \mid 1 \leq j \leq k, 1 \leq i \leq p\}$, and the weight function w is defined as

$$w((\mathbf{x_i}, \mathbf{x_j})) = \exp\left(\frac{-\max\left(0, d(\mathbf{x_i}, \mathbf{x_j}) - \rho_i\right)}{\sigma_i} \right). \tag{3.75}$$

Given this collection of *local graphs* (represented as a single directed graph $\bar{\mathcal{G}}$), we then require a method to combine them into a *unified topological representation*. Let $\mathbf{A}$ be the adjacency matrix of $\bar{\mathcal{G}}$, we can derive a symmetric matrix $\mathbf{B}$

$$\mathbf{B} = \mathbf{A} + \mathbf{A}^\intercal - \mathbf{A} \circ \mathbf{A}^\intercal, \tag{3.76}$$

where $\circ$ denotes the Hadamard product. Matrix $\mathbf{B}$ is the adjecency matrix of the equivalent *undirected weighted graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. "If one interprets the value of $\mathbf{A}_{ij}$ as the probability that the directed edge from $\mathbf{x_i}$ to $\mathbf{x_j}$ exists, then $\mathbf{B}_{ij}$ is the probability that at least one of the two directed edges (from $\mathbf{x_i}$ to $\mathbf{x_j}$ and $\mathbf{x_j}$ to $\mathbf{x_i}$) exists" McInnes et al. (2018).

The second step consists in optimizing a low-dimensional representation $\mathcal{Y} = \{\mathbf{y_1}, \ldots, \mathbf{y_p}\}$ of $\mathcal{X}$, where $\mathbf{y_i} \in \mathbb{R}^m$ and $m \ll n$. The goal is to preserve the fuzzy set structure of the high-dimensional graph. This is achieved by minimizing the cross-entropy loss between the weighted graph $\mathcal{G}$ and an equivalent weighted graph $\mathcal{H}$, constructed from the points in $\mathcal{Y}$, with respect to the lower-dimensional representations. The loss function $\mathcal{C}$ is given by

$$\begin{aligned} \mathcal{C} &= \sum_{(i,j) \in \mathcal{E}} p_{ij} \cdot \log\left(\frac{p_{ij}}{q_{ij}}\right) + (1 - p_{ij}) \cdot \log\left(\frac{1 - p_{ij}}{1 - q_{ij}}\right) \\ &= c - \sum_{(i,j) \in \mathcal{E}} p_{ij} \cdot \log(q_{ij}) + (1 - p_{ij}) \cdot \log(1 - q_{ij}). \end{aligned} \tag{3.77}$$

Here c is a constant depending only on p_{ij}, where $p_{ij} = \mathbf{B}_{ij}$ represents the probability of existence of a connection between nodes $\mathbf{x_i}$ and $\mathbf{x_j}$. This value, called a *fuzzy membership value*, measures the strength of the connection between two nodes. The term "fuzzy" reflects the fact that these are not strict binary memberships but

rather continuous values between 0 and 1. The quantity q_{ij} is the corresponding low-dimensional fuzzy membership, computed as

$$q_{ij} = \left(1 + a\left(\|\mathbf{y_i} - \mathbf{y_j}\|_2^2\right)^b\right)^{-1}, \tag{3.78}$$

where a and b are hyperparameters automatically set during the algorithm's initialization. The cross-entropy loss $\mathcal{C}$ is minimized via stochastic gradient descent until a maximum number of epochs is reached. It's important to note that the low-dimensional embeddings $\mathcal{Y}$ are not initialized randomly. Instead, they are initialized using *spectral embedding* to obtain topologically meaningful coordinates, which accelerates convergence. Given the adjacency matrix $\mathbf{B}$, we first compute the *Degree Matrix* $\mathbf{D}$[29]. The *Normalized Laplacian Matrix* is then defined as $\mathbf{L} = \mathbf{D}^{\frac{1}{2}}(\mathbf{D} - \mathbf{A})\mathbf{D}^{\frac{1}{2}}$. Next, we perform the *eigen-decomposition* of $\mathbf{L}$[30]. The eigenvalues of $\mathbf{L}$ are $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_p$ with corresponding eigenvectors $\mathbf{u_1}, \ldots, \mathbf{u_p}$[31]. The first m eigenvectors $\mathbf{u_1}, \ldots, \mathbf{u_m}$ are used to initialize the low-dimensional representations, where each point $\mathbf{y_i} \in \mathcal{Y}$ is given by $\mathbf{y_i} = (\mathbf{u_1}_i, \ldots, \mathbf{u_m}_i)$. Equivalently, if $\mathbf{U} \in \mathbb{R}^{p \times m}$ is the matrix of eigenvectors, then each point $\mathbf{y_i}$ corresponds to the i-th row of $\mathbf{U}$.

3.3.2 HDBSCAN

HDBSCAN (*Hierarchical Density-Based Spatial Clustering of Applications with Noise*) is a density-based clustering algorithm proposed in Campello et al. (2013). This method improves upon the popular DBSCAN algorithm by eliminating the requirement for a global density threshold, which assumes uniform density across all clusters. In practice, clusters often vary in density, shape, and size, making a single global density threshold overly restrictive.

Given the dimensionally-reduced dataset $\mathcal{Y} = \{\mathbf{y_1}, \ldots, \mathbf{y_p}\}$, the first step in HDBSCAN is to determine the space of transformed distances (i.e. mutual reachability distances). To do so, it is essential to define some key concepts. Let $d(\cdot, \cdot)$ be a distance metric (e.g., Euclidean distance) that measures the distance between any two points in $\mathcal{Y}$. Next, the *Core Distance* $d_{core}(\cdot)$ of an object $\mathbf{y_i} \in \mathcal{Y}$ w.r.t. m_{pts} is defined as the distance from $\mathbf{y_i}$ to its m_{pts}-nearest neighbor. Hence, points in denser regions will have smaller core distances compared to points in sparser regions. Finally, we define the *Mutual Reachability Distance* between two objects $\mathbf{y_i}$ and $\mathbf{y_j}$ w.r.t. m_{pts} as

$$d_{mrd}(\mathbf{y_i}, \mathbf{y_j}) = \max\left\{d_{core}(\mathbf{y_i}), d_{core}(\mathbf{y_j}), d(\mathbf{y_i}, \mathbf{y_j})\right\}. \tag{3.79}$$

"Under this metric, dense points (with low core distance) remain the same distance from each other, but sparser points are pushed away to be at least their core distance away from any other point"[32]. Figure 3.15 illustrates these distance concepts using

[29] Where each diagonal entry is $\mathbf{D}_{ii} = \sum_{j=1}^{p} \mathbf{A}_{ij}$.

[30] The eigen-decomposition of $\mathbf{L}$ is well-defined since it is a $p \times p$ square matrix. Moreover, since $\mathcal{G}$ is undirected, $\mathbf{L}$ is symmetric. Hence, the left and right eigenvectors associated with the same eigenvalue are identical.

[31] Given the properties of the Laplacian matrix, $\lambda_1 = 0$ and $\mathbf{u_1} = \mathbf{1}$, as all rows sum up to zero.

[32] https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html.

an example where $m_{pts} = 5$. The core distances are the distances of $\mathbf{y}_{blue}, \mathbf{y}_{red}, \mathbf{y}_{green}$ from their respective 5-nearest neighbors. For instance, it can be observed that $d_{core}(\mathbf{y}_{blue}) < d(\mathbf{y}_{blue}, \mathbf{y}_{green}) < d_{core}(\mathbf{y}_{green})$. Consequently, applying the definition in equation (3.79) we get $d_{mrd}(\mathbf{y}_{blue}, \mathbf{y}_{green}) = d_{core}(\mathbf{y}_{green})$.

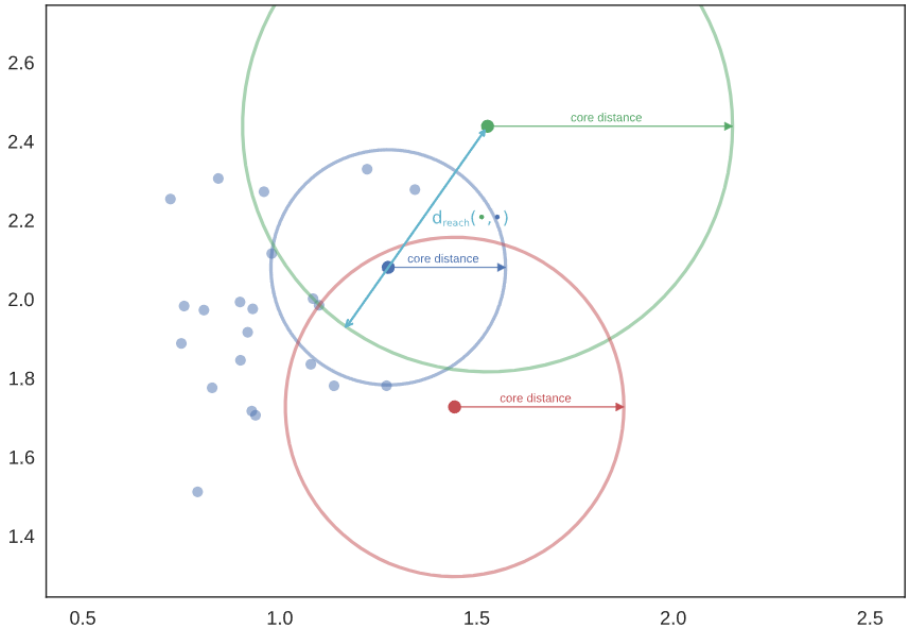


Figure 3.15. Mutual Reachability Distance. Source: McInnes et al. (2016).

The next step requires computing the *Mutual Reachability Graph*, i.e. a complete graph $\mathcal{G}_{mrd}(\mathcal{Y}, \mathcal{E}_{\mathcal{Y}})$ having as nodes the elements of $\mathcal{Y}$ connected by edges $(\mathbf{y_i}, \mathbf{y_j})$ weighted by their mutual reachability distance $d_{mrd}(\mathbf{y_i}, \mathbf{y_j})$. Now we can build the *Minimum Spanning Tree* (MST) of $\mathcal{G}_{mrd}$, i.e. a tree-like subgraph of $\mathcal{G}_{mrd}$ that connects all vertices of $\mathcal{Y}$ while minimizing the total edge weight of MST. Specifically, in the aforementioned paper, the authors suggest using an implementation of Prim's algorithm based on an ordinary list search (instead of a heap). The MST is further extended by adding for each vertex $\mathbf{y_i}$ a self-loop $(\mathbf{y_i}, \mathbf{y_i})$ having weight $d_{core}(\mathbf{y_i})$. We call MST_{ext} the extended version of MST.

The third main step of HDBSCAN involves converting MST_{ext} into a *hierarchy* of connected components represented by a *dendrogram*, as shown in Figure 3.16. We start by considering every object in $\mathcal{Y}$ as part of the same root cluster C_{root}. Then, we iteratively remove all edges (self-loops included) from MST_{ext} in descending order of weights[33] (mutual reachability distance). "After each removal, we must process one at a time each cluster that contained the edge(s) just removed by relabeling its resulting connected subcomponent(s)" Campello et al. (2013). If the size of a subcomponent is smaller than the minimum cluster size $m_{clSize} \geq 1$, we label the subcomponent as *spurious* and the objects contained in the subcomponent as *noise*. When all the subcomponents arising from the edge(s) removal are *spurious*, the cluster disappears. If a single subcomponent of the cluster is *not spurious*, we keep the original cluster label (the cluster has shrunk in size). Instead, if two or more

[33] In case of ties, edges must be removed simultaneously.

subcomponents are *not-spurious*, then we assign new cluster labels to each of them. We call this a *"true" cluster split*. For simplicity, we can set $m_{clSize} = m_{pts}$[34].

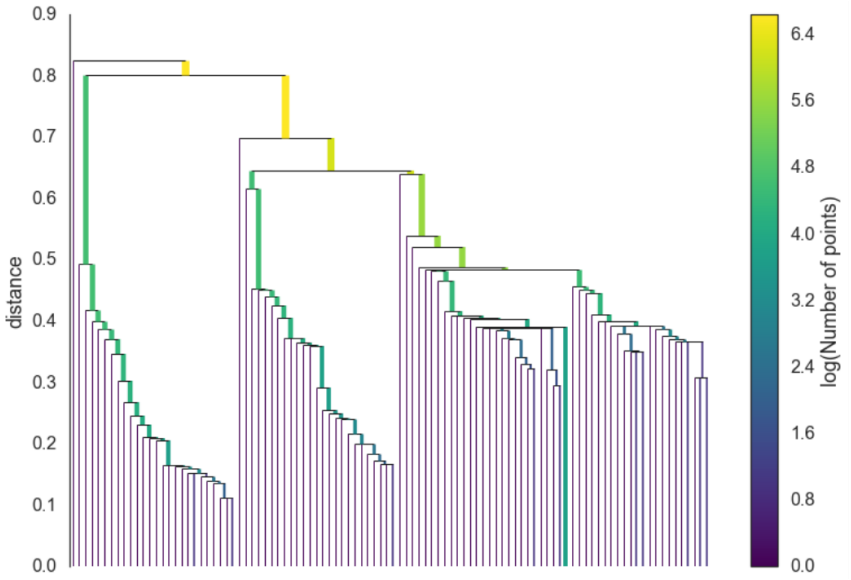


Figure 3.16. HDBSCAN Cluster Hierarchy. Source: McInnes et al. (2016).

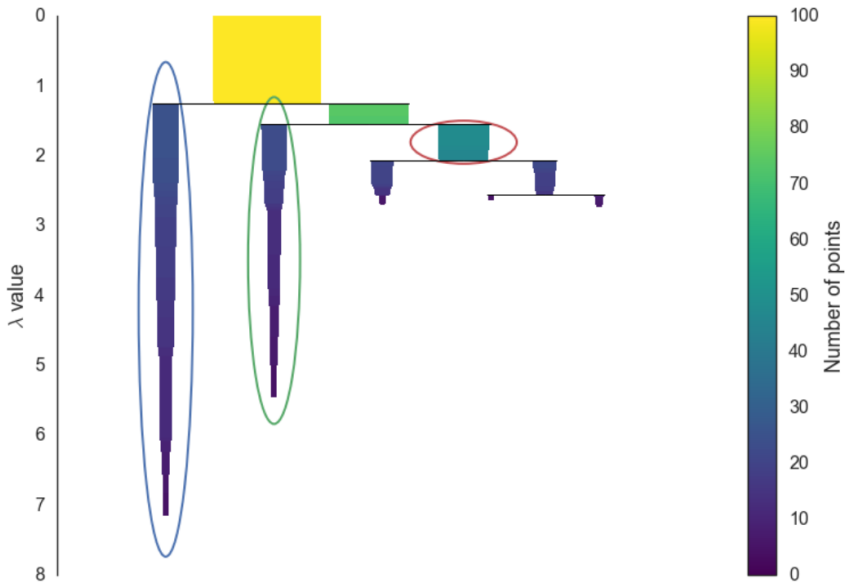


Figure 3.17. HDBSCAN Cluster Selection. Source: McInnes et al. (2016).

Given the cluster hierarchy, the final clusters must be extracted. Unlike DBSCAN, which assumes uniform density across clusters and uses a single global density threshold, HDBSCAN accounts for varying densities between clusters. Therefore, a single global density threshold cannot be used to cut the condensed dendrogram to

[34]The `hdbscan.HDBSCAN` class automatically sets `min_samples` (m_{pts}) to `min_cluster_size` (m_{clSize}), if unspecified.

determine the number of clusters. Instead, the authors of HDBSCAN propose to identify the optimal clusters by solving an optimization problem aimed at maximizing the total *cluster stability* J. The stability $S(c_i)$ of cluster C_i is defined as

$$S(C_i) = \sum_{\mathbf{x_j} \in C_i} (\lambda_{death} - \lambda_{birth}), \tag{3.80}$$

and it measures how stable cluster C_i is in the hierarchy. "This is dependent on the density-range in which the cluster exists, i.e. minimum density value at which the cluster starts to exist and maximum density level at which the cluster is either split or disappears" Strobl et al. (2021). To solve this problem "we process every node except the root, starting from the leaves (bottom-up), deciding at each node C_i whether C_i or the best-so-far selection of clusters in C_i's subtrees should be selected. To be able to make this decision locally at C_i, we propagate and update the total stability $\hat{S}(C_i)$ of clusters selected in the subtree rooted at C_i in the following, recursive way:

$$\hat{S}(C_i) = \begin{cases} S(C_i) & \text{if } C_i \text{ is a leaf node} \\ \max\left\{S(C_i), \hat{S}(C_{i_l}) + \hat{S}(C_{i_r})\right\} & \text{if } C_i \text{ is an internal node} \end{cases} \tag{3.81}$$

where C_{i_l} and C_{i_r} are the left and right children of C_i (for the sake of simplicity, we discuss the case of binary trees; the generalization to n-ary trees is trivial)" Campello et al. (2013). Let's make an example by looking at Figure 3.18. Here, clusters C_{10} and C_{11} together are preferred over cluster C_8 alone because $S(C_8) < S(C_{10}) + S(C_{11})$. However, when comparing the set $\{C_{11}, C_{10}, C_9\}$ to cluster C_5, we choose C_5 as it has better stability. Then, clusters C_4 and C_5 together are better than C_2, and C_3 alone is more stable than C_6, C_7. The final clusters extracted are C_3, C_4 and C_5, as this selection maximizes the total cluster stability $J = S(C_3) + S(C_4) + S(C_5) = 6 + 6 + 5 = 17$.

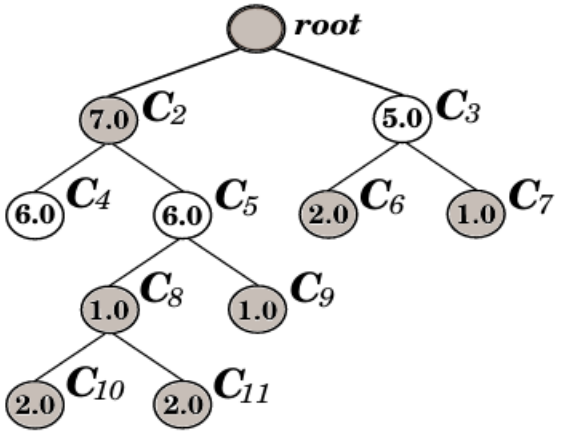


Figure 3.18. Illustration of the optimal selection of clusters from a given cluster tree. Source: Campello et al. (2013).

3.3.3 c-TF-IDF

The clusters identified by HDBSCAN correspond to the topics found within our corpus of documents. The goal is to extract topic representations, meaning that we want to identify the words or n-grams that best describe each cluster or topic. The process begins by treating each cluster as a single document. This is done by concatenating all document chunks within a cluster into one long document. Next, we apply TF-IDF to this new corpus of documents. This approach is known as *Class TF-IDF* (c-TF-IDF) and is computed as

$$(\text{c-tf-idf})_{t,c} = \text{tf}_{t,c} \times \log\left(1 + \frac{\text{A}}{\text{tf}_t}\right), \tag{3.82}$$

where $\text{tf}_{t,c}$ is the frequency of term t in cluster c, and the second term is the *inverse class frequency* computed as the logarithm of the average number of terms A in c divided by the term frequency of term t across all clusters. The $(\text{c-tf-idf})_{t,c}$ score represents the importance of term t in class c. Then, we select the most significant words or n-grams to describe each cluster. The number of words selected is user-defined (e.g. 10).

Furthermore, given cluster representations of length V, where V is the vocabulary size of the document corpus, we can easily compute the similarity between clusters. If HDBSCAN identifies more clusters than the user's defined maximum, we can iteratively merge the most similar clusters until we reach the desired number of clusters.

Chapter 4

Analysis and Results

4.1 Authorship Classification

One of the primary objectives of this work is to classify the Governors based on their documents. We decided to test three different scenarios: using only English documents, only Italian documents, and a combination of both English and Italian documents. Among the 122 English documents, 102 are attributed to Visco, 19 to Draghi, 1 to Ciampi, and none to Fazio. Given that we had too few examples from Ciampi in English, we were obliged to exclude him from the classifier trained solely on English documents. This resulted in a total of 121 documents for the English classifier. For the classifier trained exclusively on Italian documents, we had a total of 377 documents distributed among the Governors as follows: 31 for Ciampi, 94 for Fazio, 75 for Draghi, and 177 for Visco. Lastly, the classifier trained on all documents—both English and Italian—comprised 499 documents: 32 for Ciampi, 94 for Fazio, 94 for Draghi, and 279 for Visco.

The first necessary step in training and fairly evaluating any classifier is to split the dataset into a *training set* and a *test set*. The training set consists of labelled documents used to train the classifier, aiming at predicting the Governor based on the document. Then, we use the test set to evaluate how well the trained classifier performs on new documents and assess its generalizability. An additional but crucial step is to reserve a portion of the training set to create a *validation set*. The latter is used for *hyperparameter tuning*. By observing the model's performance on the validation set, we can identify which hyperparameters yield the best generalization to unseen data. Without this step, the model may *overfit* to the training data, reducing its effectiveness on new, unseen documents.

In our case, we decided to use an 80%-20% split, where 80% of the documents belong to the training set and the remaining 20% to the test set. We employed the `sklearn.model_selection.train_test_split` function, setting the `stratify` parameter to the labels of the dataset and `random_state=42` to guarantee *reproducibility* of results. Using `stratify` ensures that the class distribution is maintained in both the training and test sets. The resulting distribution of Governors in the training and test sets was the following:

- For the English-only dataset, the training set consists of 96 documents, and the test set has 25 documents. The Governor distribution is:

  - Train set: `{"Visco":81, "Draghi":15}`
  - Test set: `{"Visco":21, "Draghi":4}`
- For the Italian-only dataset, the training set consists of 301 documents, and the test set has 76 documents. The Governor distribution is:
  - Train set: `{"Ciampi":25, "Fazio":75, "Draghi":60, "Visco":141}`
  - Test set: `{"Ciampi":6, "Fazio":19, "Draghi":15, "Visco":36}`
- For the complete dataset, the training set consists of 399 documents, and the test set has 100 documents. The Governor distribution is:
  - Train set: `{"Ciampi":26, "Fazio":75, "Draghi":75, "Visco":223}`
  - Test set: `{"Ciampi":6, "Fazio":19, "Draghi":19, "Visco":56}`

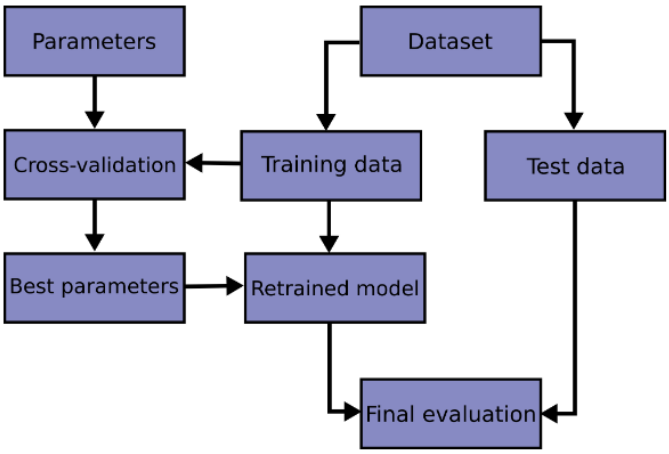


Figure 4.1. Hyperparameter Tuning. Source: Pedregosa et al. (2011).

We first tested four different combinations of document embedding techniques and classification methods. Specifically, we used TF-IDF and Doc2Vec to generate document embeddings, which were then fed into either Support Vector Machines (SVM) or Logistic Regression models. Additionally, we fine-tuned pre-trained Transformer-based models. For these models, we employed their associated tokenizers to process the documents, splitting document into lists of token and appropriately converting these tokens into IDs.

4.1.1 Evaluation Metrics

When evaluating the generalization capability of a classification model, a commonly used metric is the *Accuracy* obtained on the test set. In the case of binary classification, accuracy is calculated as

$$Accuracy = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}, \tag{4.1}$$

where TP (*True Positives*) is the number of correctly predicted positive instances, TN (*True Negatives*) is the number of correctly predicted negative instances, FP (*False Positives*) is the number of negative instances incorrectly classified as positive, and FN (*False Negatives*) is the number of positive instances incorrectly classified as

negative. The sum of true positives, true negatives, false positives, and false negatives equals the total number of examples in the dataset. For multiclass classification, accuracy is generalized as

$$Accuracy = \frac{\sum_{i=1}^{\mathcal{C}} \mathrm{TP}_i}{\text{Total Number of Instances}}, \tag{4.2}$$

where $\mathcal{C}$ is the number of classes and TP_i is the number of correctly classified instances of class i.

Standard accuracy is suitable when class sizes are relatively balanced. However, in scenarios where *class imbalance* is significant—as is the case in our dataset, which is heavily skewed towards Visco, who has the most documents in both English and Italian—standard accuracy can be misleading. For instance, in the English-only case, the test set contains 21 documents by Governor Visco (*majority class*) and only 4 by Draghi (*minority class*). Always predicting Visco would yield an 84% accuracy, which is a high value despite the model not learning to distinguish between classes. To address this, we can use the *Balanced Accuracy*, which equally weights each class regardless of size. For the SVM and Logistic Regression models we used the `balanced_accuracy_score` function from the `sklearn.metrics` module, while for the Transformer-based models we manually implemented this metric.

Balanced accuracy is computed as the average recall (or sensitivity) across all classes

$$Balanced\ Accuracy = \frac{1}{\mathcal{C}} \sum_{i=1}^{\mathcal{C}} \frac{\mathrm{TP}_i}{\mathrm{TP}_i + \mathrm{FN}_i}, \tag{4.3}$$

where TP_i is the number of true positives for class i and FN_i is the number of false negatives for class i, i.e. the number of instances of class i incorrectly predicted as another class.

To interpret the accuracy of a model we can compare it to the *baseline* accuracy. For binary classification, the baseline accuracy is 50%, while for a 4-class classification problem, the baseline is 25%. Any accuracy above the baseline shows an improvement over random guessing.

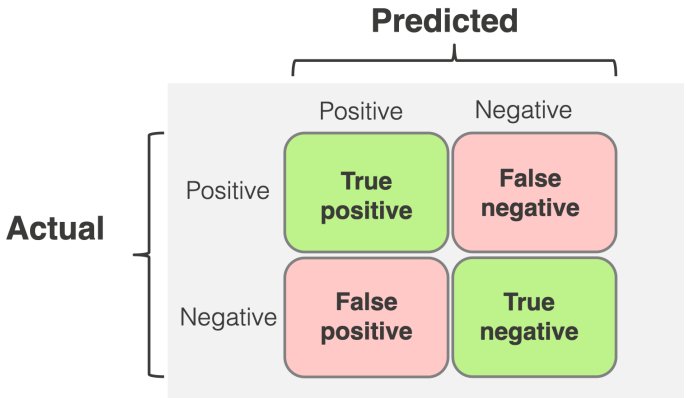


Figure 4.2. Confusion matrix. Source: Evidently AI Team (n.d.).

In addition to accuracy, the *Confusion Matrix* provides deeper insights into the errors made by the classifier: are the errors random or do they highlight a specific

pattern? In the binary classification case (see Figure 4.2), the confusion matrix is (2×2) matrix containing the counts of true positives, true negatives, false positives, and false negatives. For multiclass classification with $\mathcal{C}$ classes, the classification matrix is a $(\mathcal{C} \times \mathcal{C})$ matrix, where the diagonal elements represent correctly classified instances, and the off-diagonal elements represent *misclassifications*. Precisely, we used a normalized version of the confusion matrix, where each value represents the *percentage* of instances classified for each class, with each row summing to 1. This normalization step increases the interpretability of results.

4.1.2 TF-IDF

The first embedding technique we tested is TF-IDF, for which we utilized the `TfidfVectorizer` class from the `sklearn.feature_extraction.text` module. This class allowed us to specify a custom document preprocessing function through the `tokenizer` parameter. Our preprocessing pipeline began by (1) identifying and masking all numbers and dates in the text using regular expressions. Specifically, we employed the `match` function from the `regex` Python library to detect patterns representing dates and numbers. Once these patterns were matched, the corresponding substrings in the text were replaced with *placeholder* tokens: `<NUMBER>` for numbers and `<DATE>` for dates and years. The next step involved (2) converting all text to lowercase using Python's `lower()` method. We then (3) removed punctuation from the documents, including characters listed in `string.punctuation` and additional special characters like "©". The fourth step consisted of (4) tokenizing the text by splitting the document strings into lists of token (word) strings. Finally, we (5) removed the *stopwords*[1], i.e. a set of commonly used (high-frequency) words in a language, that carry very little useful information for distinguishing between documents. We defined our stopword set by combining English (`stopwords.words("english")`) and Italian (`stopwords.words("italian")`) stopwords from the `nltk.corpus` library, supplemented with additional stopword lists from external sources[2]. Furthermore, we treated the tokens `<NUMBER>` and `<DATE>` as stopwords, removing them from the documents as they appeared frequently across all texts and did not provide specific information about any particular document.

In the `TfidfVectorizer`, we can specify a range of hyperparameters, each of which can significantly influence the resulting document embeddings and, consequently, the model's performance. The hyperparameters are:

- The `ngram_range` (`min_n`, `max_n`) defines the "lower and upper boundary of the range of n-values for different n-grams to be extracted. All values of n such that `min_n` $\leq n \leq$ `max_n` will be used. For example, an `ngram_range` of $(1, 1)$ means only unigrams, $(1, 2)$ means unigrams and bigrams, and $(2, 2)$ means only bigrams" Pedregosa et al. (2011). A larger `ngram_range` results in a longer (and sparser) embedding vector. Including n-grams with $n \geq 2$ introduces contextual information, as words are considered in combination with neighboring words rather than in isolation. The `ngram_range` values we tested are $(1, 1), (1, 2), (2, 2), (2, 3), (3, 3)$.

[1] Examples of English stopwords are: ["the", "a", "in"]. Examples of Italian stopwords are: ["il", "e", "così"].

[2] Italian stopwords and English stopwords.

- The `min_df` parameter specifies the minimum document frequency threshold. The terms that appear in fewer documents than the specified threshold are ignored when building the vocabulary. We tested the following `min_df` values: $2, 3, 4, 5, 6, 7, 8$.
- When not `None`, the `max_features` parameter limits the vocabulary size to the top `max_features` terms ordered by term frequency across the corpus. To determine the `max_features` values for testing during grid search, one can select a percentage of the total number of features. We tested the following percentages: $0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9$. The total number of features is first determined by fitting a `TfidfVectorizer` with a specific `ngram_range` and extracting the vocabulary size. The features that meet both the `min_df` and `max_features` conditions are then identified. If the number of terms respecting the condition imposed by `min_df` exceeds the specified `max_features` value, the vocabulary size is reduced to `max_features`. Otherwise, the number of features is determined solely by the `min_df` condition.
- The `analyzer` parameter must be set to "`word`" when providing a custom tokenization function to `tokenizer`.
- We specified the stopwords by passing a list to the `stop_words` parameter.
- The `norm` parameter was set to "`l2`" meaning that the sum of squares of vector elements is 1.

For each combination of hyperparameters, the `vectorizer` was defined as

```
vectorizer = TfidfVectorizer(**inputs).
```

The `vectorizer` learns the vocabulary and inverse document frequencies by fitting to the training documents using the `fit` method. Document embeddings for the training and test sets were then generated employing the `transform` method. Since the vocabulary is built from the training set, the length of the embeddings depends on the number of unique tokens in the training data. If a token is present in the training set but absent in the test set, its corresponding value in the test set embedding will be zero. However, tokens appearing in the test set but not in the training set are simply ignored.

For the English documents, the total vocabulary consisted of 12106 words, with 4247 tokens common to both the training and test sets. In addition, 6908 tokens were found only in the training set, while 951 tokens appeared exclusively in the test set, making up approximately 8% of the vocabulary. For the Italian documents, the total vocabulary included 45422 tokens, of which 13813 were shared between the training and test sets, 27986 appeared only in the training set, and 3623 tokens were unique only to the test set, also representing around 8% of the vocabulary.

While ignoring these 951 English tokens and 3623 Italian tokens might mean losing valuable information for classification, including them would indirectly introduce test set data into the training process, thereby contaminating the model with information it should not have during training. Moreover, this approach allows the trained `vectorizer` to be applied directly to new documents for classification without having to rebuild all the embeddings from scratch.

Let's call the training and test set embedding matrices `X_train` and `X_test`. The TF-IDF feature vectors contained in `X_train` (and `X_test`) were high-dimensional

and sparse due to the large number of features v. To mitigate the risk of overfitting and reduce computational time, we decided to reduce their dimensionality by applying *Principal Component Analysis* (PCA). The latter is a classical dimensionality reduction technique introduced by Pearson (1901) that aims to project high-dimensional data into a lower-dimensional space while maximizing the variance of the data in the reduced space. In practice, we used the `PCA` class from the `sklearn.decomposition` module. It employs *Singular Value Decomposition* (SVD) for the PCA decomposition, which is more computationally efficient than the standard PCA method that requires computing and storing the $(v \times v)$ covariance matrix of `X_train`. Specifically, when the number of data points d is much smaller than the number of features v, `PCA` utilizes a variant of SVD known as *Truncated SVD* to avoid the computational burden of full SVD. Truncated SVD decomposes `X_train` as

$$\texttt{X\_train} = \mathrm{U}_{\mathbf{k}} \mathbf{\Sigma}_{\mathbf{k}} \mathrm{V}_{\mathbf{k}}^{\intercal}. \tag{4.4}$$

Here, $\mathrm{U}_{\mathbf{k}}$ is a $(d \times k)$ matrix with orthonormal columns, $\mathbf{\Sigma}_k$ is a $(k \times k)$ diagonal matrix with the top k singular values, V is a $(v \times k)$ matrix with orthonormal columns, and k represents the number of *principal components* to retain. Initially, k was set to $\min\{d, v\}$. The dataset `X_train` was then projected onto the space defined by the principal components. To ensure that a significant portion of the original variance is preserved, principal components were selected such that at least 70% of the variance was retained. Once the `PCA` instance was fitted on `X_train`, dimensionality reduction was applied to both `X_train` and `X_test` using the `transform` method.

SVM

Hyperparameter tuning for SVM models was done using the `GridSearchCV` class from the `sklearn.model_selection` module. An instance of the `sklearn.svm.SVC` class was passed to the `estimator` parameter. Given the class imbalance in the dataset, it is best to provide `class_weights` to the SVM classifier. These class weights were computed using the scikit-learn `compute_class_weight` function, where the weight for each class i in the label vector $\mathbf{y}$ was calculated as

$$w_i = \frac{n}{\mathcal{C} \cdot n_i}, \tag{4.5}$$

where n represents the total number of samples in the dataset, n_i is the total number of samples belonging to class i, and $\mathcal{C}$ is the total number of classes. This ensured that larger weights were assigned to minority classes, making errors in these classes more heavily penalized compared to errors in majority classes. For instance, in the case of English-only data, the weights for the Draghi (minority class) and Visco (majority class) classes were respectively

$$w_{\text{Draghi}} = \frac{121}{2 \cdot 19} = 3.18, \quad w_{\text{Visco}} = \frac{121}{2 \cdot 102} = 0.59.$$

For multiclass scenarios, the `decision_function_shape` parameter in `SVC` was set to "`ovr`" (one-vs-rest). Besides the `estimator`, in `GridSearchCv`, the balanced accuracy score was specified as the `scoring` function, and the `parameter_grid` was defined as a dictionary containing the following SVM hyperparameters:

- The inverse regularization parameter `C`, with values $0.1, 1, 10$.

- The kernel type, set to either `rbf` or `poly`.
- The degree for the polynomial kernel, taking values $1, 2, 3$.
- The gamma parameter for the RBF kernel, with values $0.001, 0.01, 0.1, 1$.

Lastly, the `cv` parameter was set to 5, indicating the use of *5-fold cross-validation*. This combined grid search and cross-validation process was employed to identify the optimal hyperparameters, i.e. the ones that yield the highest validation balanced accuracy. The 5-fold cross-validation procedure, illustrated in Figure 4.3, involves splitting the training data into 5 folds. The model is trained on 4 of these folds and validated on the remaining fold, producing a balanced accuracy score. This process is repeated five times, with each fold serving as the validation set once. The resulting five balanced accuracy scores were then averaged to provide a more stable measure of the model's generalization capability.

Once the SVM hyperparameters yielding the highest cross-validation balanced accuracy were identified, we retrained the model on the entire training set using these optimal hyperparameters (look at Figure 4.1). We selected the model with the highest cross-validation accuracy among all those trained with different combinations of TF-IDF parameters (`ngram_range`, `min_df`, `max_features`). The final trained classifier was then used to predict the labels on the test set and evaluate its performance.

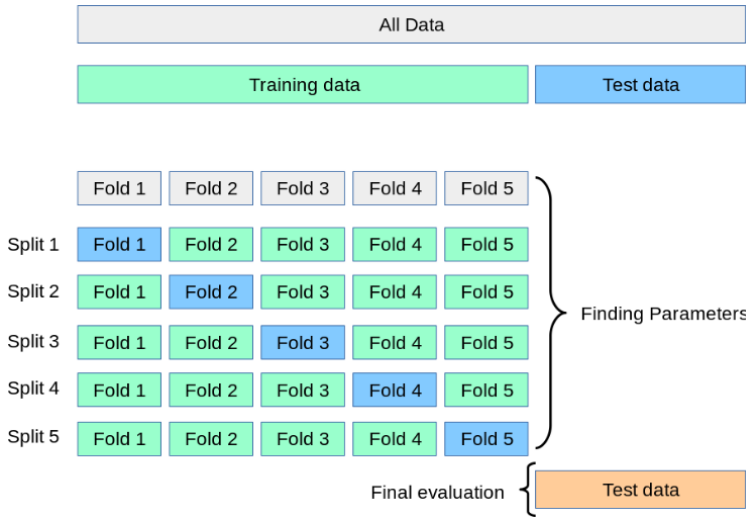


Figure 4.3. K-fold Cross-Validation. Source: Pedregosa et al. (2011).

For the classifier trained exclusively on the English documents, the class weights were set to

$$w_{\text{Draghi}} = 3.18, \;\; w_{\text{Visco}} = 0.59.$$

The best model was trained using the TF-IDF embeddings with an `ngram_range` of $(2, 2)$ meaning only bigrams were used. The `min_df` parameter was set to 2, considering tokens that appear in more than one document, and `max_features` were limited to 7636, representing the top 10% of the overall vocabulary. The SVM classifier used a polynomial kernel of degree 1 with an inverse regularization strength of 1. The model achieved a validation balanced accuracy of 88.79%, a training balanced accuracy of 98.76%, and a test balanced accuracy of 87.50%.

As seen in Figure 4.4, the confusion matrix shows that only 1 out of 4 of Draghi's documents was misclassified, with the rest of the documents correctly classified.

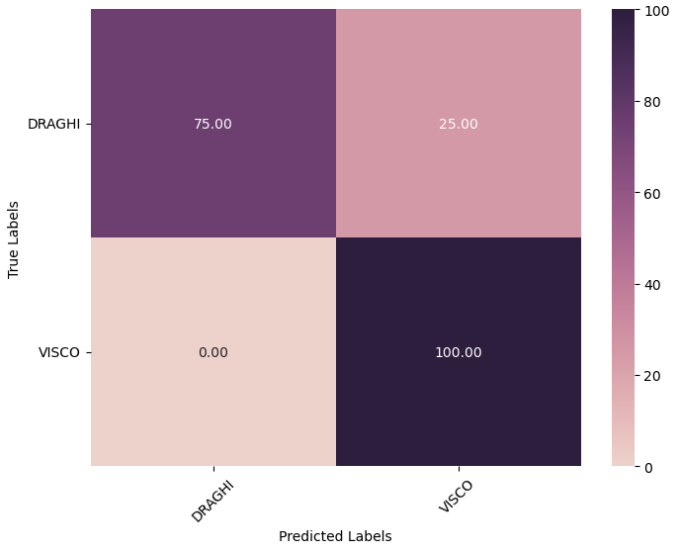


Figure 4.4. Test confusion matrix for the English TF-IDF SVM model.

For the classifier trained exclusively on the Italian documents, the class weights were set to

$$w_{\text{Ciampi}} = 3.04,\ w_{\text{Fazio}} = 1.00,\ w_{\text{Draghi}} = 1.26,\ w_{\text{Visco}} = 0.53,$$

with Ciampi being the smallest minority class and Visco the majority class. The optimal model was trained using a TF-IDF embedding with an `ngram_range` of $(2, 3)$, incorporating both bigrams and trigrams. The `min_df` parameter was set to 7, and the maximum number of features was set to 96797, representing 10% of the entire vocabulary. After applying the `min_df` condition, the vocabulary size was reduced to 6280 terms. The SVM model used a polynomial kernel of degree 1 with an inverse regularization strength `C` of 10. This model yielded a validation balanced accuracy of 91.95%, a training balanced accuracy of 100%, and a test balanced accuracy of 87.16%.

Figure 4.5 shows the confusion matrix for this model, demonstrating that the majority of Fazio's and Visco's test documents were classified correctly. However, 16.67% of Ciampi's test documents were misclassified as Visco's, and 26.67% of Draghi's documents were also incorrectly attributed to Visco. Most classification errors were temporal, meaning the incorrect predictions were for the nearest temporal Governors.

Finally, for the classifier trained on the combined English-Italian dataset, the class weights were set to

$$w_{\text{Ciampi}} = 3.90,\ w_{\text{Fazio}} = 1.33,\ w_{\text{Draghi}} = 1.33,\ w_{\text{Visco}} = 0.45.$$

As with the Italian model, Ciampi represented the smallest minority class, and Visco represented the majority class. The best model was trained using the TF-IDF

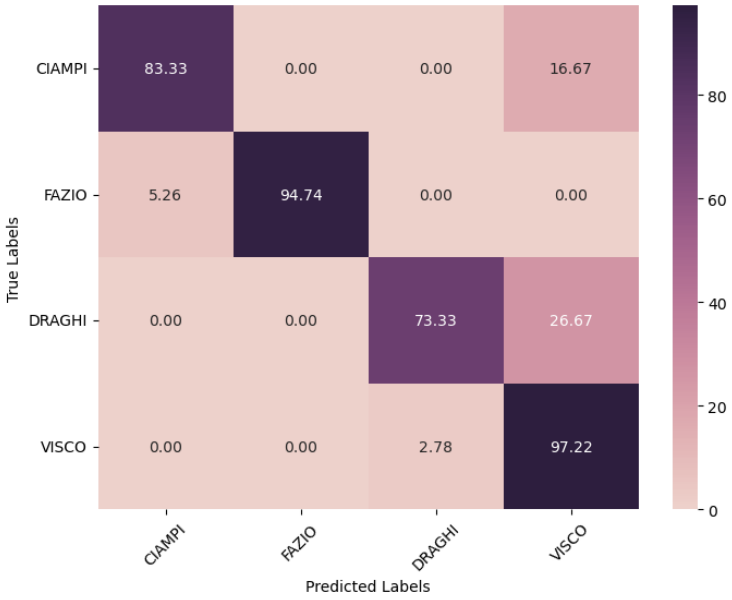


Figure 4.5. Test confusion matrix for the Italian TF-IDF SVM model.

embeddings with an `ngram_range` of $(1, 2)$, using unigrams and bigrams. The `min_df` parameter was set to 5, and the maximum number of features was capped at 53845, which is 10% of the total vocabulary. After applying the `min_df` condition, the vocabulary was reduced to 24409 terms. The SVM classifier employed a radial basis function (RBF) kernel with a gamma value of 0.1 and an inverse regularization strength of 10. This model achieved a validation balanced accuracy of 90.70%, a training balanced accuracy of 98.99%, and a test balanced accuracy of 86.18%.

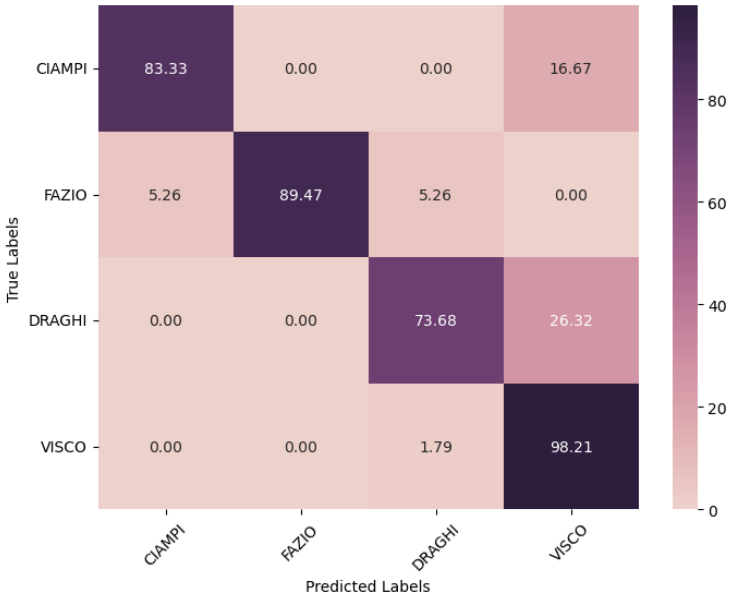


Figure 4.6. Test confusion matrix for the English-Italian TF-IDF SVM model.

The confusion matrix in Figure 4.6 shows that most of Fazio's and Visco's test documents were correctly classified, and 16.67% of Ciampi's documents were classified as Visco's, and 26.67% of Draghi's documents were attributed to Visco.

Multiple Logistic Regression

Hyperparameter tuning for the Multiple Logistic Regression models was again performed using the scikit-learn `GridSearchCV` class. We passed an instance of the `LogisticRegression` class from the `sklearn.linear_model` module to the `estimator` parameter. The following inputs were specified for `LogisticRegression`:

- To address class imbalance, `class_weights` were used, computed in the same manner as for the SVM models.
- The `random_state` was set to 111 to ensure the reproducibility of results, given the stochastic nature of the optimization process.
- The maximum number of iterations (`max_iter`) was set to 1000 to allow sufficient time for the solver to converge.
- The "`lbfgs`" (Limited-memory BFGS) `solver` was chosen for its compatibility with multi-class problems.
- The tolerance for convergence (`tol`) was left at the default value of 1e-4. If the maximum number of iterations is not yet reached, the tolerance serves as an additional criterion for stopping the optimization process.
- For binary classification problems, the `multi_class` parameter was set to "`ovr`" (one-vs-rest), while for multi-class problems, it was set to "`multinomial`", where all classes are considered simultaneously during optimization.
- The penalty term `penalty` was set to "`l2`", which compatible with the `lbfgs` solver.

Besides the `estimator`, in the `GridSearchCv` class we specified the balanced accuracy score as the scoring metric, and the `parameter_grid` was defined as a dictionary containing the values for `C`, the inverse regularization strength, which is tested with values in $[0.001, 0.01, 0.1, 1, 10]$.

The cross-validation parameter (`cv`) was set to 5, indicating that a 5-fold cross-validation procedure was applied to tune the hyperparameters of the Logistic Regression model. Once the optimal hyperparameters were identified, the model was retrained on the entire training set. The model that achieved the highest cross-validation accuracy among all combinations of the TF-IDF parameters was then selected. Finally, the balanced accuracy of this best-performing model was computed on the test set.

For the classifier trained exclusively on the English documents, the best model was trained using the TF-IDF embedding with an `ngram_range` of $(3, 3)$, meaning only trigrams were used. The `min_df` parameter was set to 3, which means only tokens appearing in at least three documents were considered, and the maximum number of features was limited to 9349, representing the top 10% of the overall vocabulary. The optimal logistic regression classifier used an inverse regularization of 0.1. The model achieved a validation balanced accuracy of 90%, a training balanced accuracy of 96.67%, and a test balanced accuracy of 87.50%.

As shown in Figure 4.7, the confusion matrix reveals that only 1 out of 4 of Draghi's documents was misclassified, with the rest of the documents correctly classified.

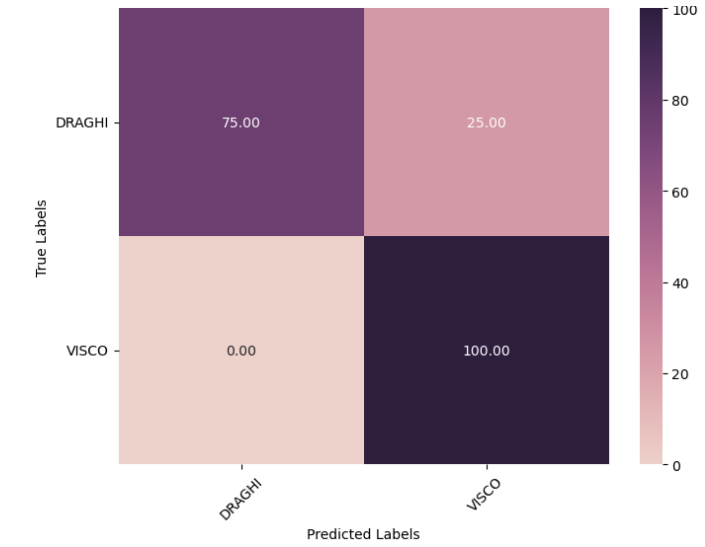


Figure 4.7. Test confusion matrix for the English TF-IDF logistic regression model.

For the classifier trained exclusively on the Italian documents, the optimal model was trained using a TF-IDF embedding with an `ngram_range` of $(2, 3)$, incorporating both bigrams and trigrams. The `min_df` parameter was set to 8, meaning only tokens present in at least eight documents were considered, and the maximum number of features was set to 96797, representing 10% of the entire vocabulary. After applying the `min_df` condition, the vocabulary was reduced to 4915 n-grams. The logistic regression model used an inverse regularization strength `C` of 1. This model achieved a validation balanced accuracy of 93.17%, a training balanced accuracy of 99.31%, and a test balanced accuracy of 84.52%.

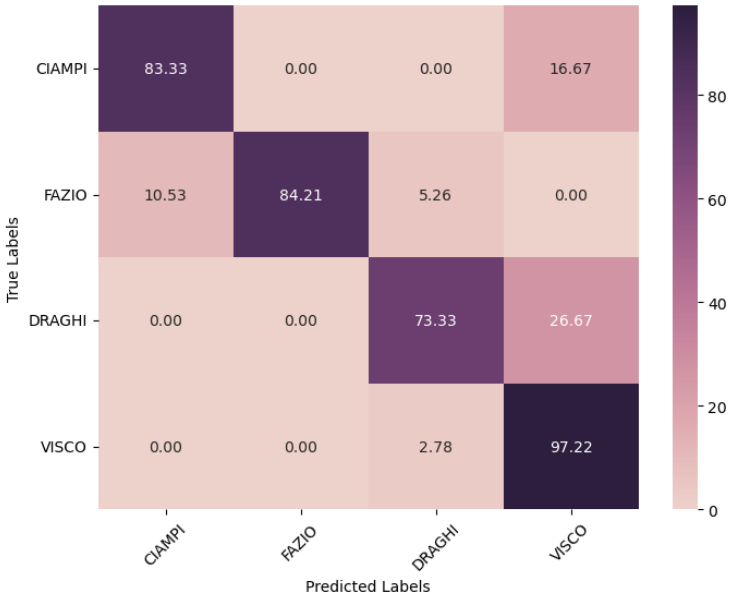


Figure 4.8. Test confusion matrix for the Italian TF-IDF logistic regression model.

Figure 4.8 illustrates the confusion matrix for this model, showing that Visco's test documents were correctly classified 97.22% of the time. However, Draghi's documents experienced the highest misclassification rate, with about 27% assigned to the wrong class.

For the classifier trained on the combined dataset of English and Italian documents, the best model was trained using the TF-IDF embeddings with an `ngram_range` of $(2, 3)$, using bigrams and trigrams. The `min_df` parameter was set to 7, and the maximum number of features was limited to 115485, which is 10% of the total vocabulary. After applying the `min_df` condition, the vocabulary was reduced to 7502 n-grams, which is equivalent to 6.5% of the original vocabulary size. The logistic regression classifier employed an inverse regularization strength of 10. This model achieved a validation balanced accuracy of 91.16%, a training balanced accuracy of 99.66%, and a test balanced accuracy of 86.62%.

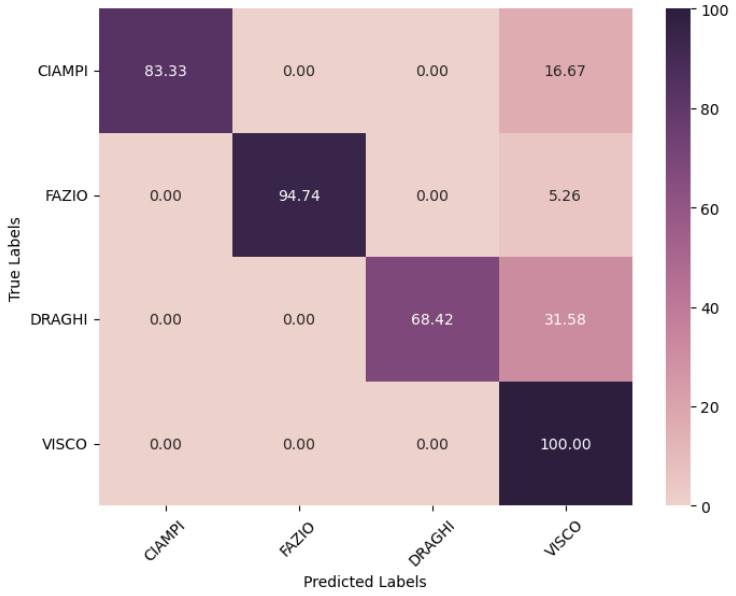


Figure 4.9. Test confusion matrix for the English-Italian TF-IDF logistic regression model.

As depicted in Figure 4.6, all of Visco's test documents were correctly classified. However, Draghi's documents were misclassified 32% of the time, and all errors were attributed to Visco's class. Unlike previous models, which tended to misclassify documents based on temporal proximity of the Governors, this model showed a clear bias toward Visco despite the use of balanced accuracy scores and class weighting adjustments during training.

Table 4.1 summarizes the optimal hyperparameters for the six TF-IDF models. It can be seen that including at least bigrams in the TF-IDF document embeddings tends to outperform using individual words. Specifically, an `ngram_range` of $(2, 3)$, incorporating both bigrams and trigrams, is the most frequently selected option, adding some contextual information to the feature vectors, as opposed to considering words in isolation. Concerning the `df_min` parameter, higher values were favored for the Italian and multilingual models, especially the Italian model, where terms were

retained only if they appeared in at least 7 or 8 documents. Interestingly, all optimal models kept only 10% of the total vocabulary. However, this value becomes relevant only for small `df_min` values. When `df_min` is set to larger values, it imposes a stricter condition on the vocabulary size than the `max_features` parameter. Apart from the SVM model for the English dataset, in all other cases, the `df_min` threshold reduced the vocabulary to a size smaller than what the `max_features` parameter would allow, making it unnecessary to increase the latter.

Dataset	Model	`ngram_range`	`df_min`	`% max_features`	`C`
English	SVM	(2,2)	2	0.1	1
	LR	(3,3)	3	0.1	0.1
Italian	SVM	(2,3)	7	0.1	10
	LR	(2,3)	8	0.1	1
English-Italian	SVM	(1,2)	5	0.1	10
	LR	(2,3)	7	0.1	10

Table 4.1. Hyperparameters of the best English, Italian and multilingual TF-IDF models.

4.1.3 Doc2Vec

Doc2Vec embeddings were generated using the `Doc2Vec` function from the Gensim `models.doc2vec` module. Before feeding the documents into `Doc2Vec`, we applied a series of preprocessing steps similar to those used for the TF-IDF embeddings. Specifically, the text was cleaned by (1) removing dates and numbers, (2) converting all documents to lowercase, (3) eliminating punctuation and special characters, and (4) tokenizing the text, i.e. splitting the documents into ordered sequences of words. The only difference from the TF-IDF preprocessing pipeline concerns the removal of stopwords. Unlike TF-IDF, Doc2Vec does not rely on word frequencies but rather on learning semantically meaningful paragraph and word embeddings from the context in which they appear. Hence, retaining stopwords is crucial to preserve the contextual information without disrupting the learning process. After preprocessing, the documents were converted into a suitable format for `Doc2Vec` using the `TaggedDocument` class from the same `gensim` library. An instance of this class produces an iterable list of tokenized documents, each associated with an integer tag in the range `(0, corpus_size-1)`, as recommended in the Gensim documentation.

In the `Doc2Vec` class, we can specify a wide range of hyperparameters that can significantly change the resulting embeddings. The key hyperparameters are:

- The `dm` parameter, which was set to `1`, defines the training algorithm. When set to `1`, the model uses the Distributed Memory Model of Paragraph Vectors (PV-DM) algorithm, which is based on the continuous bag of words (CBOW) Word2Vec model. If set to `0`, it would use the Distributed Bag of Words (DBOW) algorithm, which is based on the skip-gram Word2Vec model.
- The `vector_size` parameter specifies the dimensionality of the feature vectors. We tested the following `vector_size` values: $50, 100, 150, 200, 250$.

- The `window` parameter controls "the maximum distance between the current and predicted word within a sentence" Řehůřek and Sojka (2010). In simpler terms, it determines the number of context words to consider. We tested window sizes of $2, 3$, and 4.
- The `min_count` parameter sets the minimum frequency threshold for terms to be included in the model. Words with a total frequency lower than this threshold are ignored, which helps to filter out rare or potentially misspelled words. The `min_count` values tested were $1, 2, 3$.
- The initial learning rate `alpha` used during the optimization process was set to 0.25, while the minimum learning rate `min_alpha` was set to 0.00025. As training progresses, the learning rate decreases linearly from `alpha` to `min_alpha`.
- The `seed` was set to 111 to ensure reproducibility of results.
- The `workers` parameter, which defines the number of worker threads used to train the model, was set to 1 to maintain reproducibility of results.
- Finally, the `epochs` parameter, which defines the maximum number of training iterations over the corpus, was set to 20 (the default value is 10).

After instantiating a Doc2Vec model with a specific set of hyperparameters, the vocabulary was constructed using the `build_vocab` method. This vocabulary, accessible via the `wv.index_to_key` attribute, is a list of all the unique words extracted from the training corpus. Once the vocabulary was built, the model was trained on the training set documents using the `train` method. The training and test set document embeddings were generated using the `infer_vector` method by iterating over the tags in the `TaggedDocument` objects. During this phase, words appearing exclusively in the test set were ignored, as the Doc2Vec model did not learn representations for those words during training.

Although the feature vectors were dense and not as high-dimensional as the TF-IDF embeddings, we applied PCA to further reduce their dimensionality. This step was performed because it yielded better results compared to using the embeddings in their original form. For dimensionality reduction, we used the `PCA` class from the scikit-learn library, following the same procedure as for the TF-IDF embeddings.

SVM

After obtaining the document embeddings, we trained the SVM classifier and optimized its hyperparameters (`vector_size`, `window`, `min_count`, kernel type, `C`) using `GridSearchCV`, following the same procedure described in the earlier section dedicated to TF-IDF embeddings. Here, we present the results of the three best-performing models across the three datasets: English-only, Italian-only, and the combined English-Italian dataset.

For the classifier trained exclusively on the English documents, the best model utilized Doc2Vec embeddings with a `vector_size` of 200. The `window` parameter was set to 3, meaning that each word's context included the 3 words to the left and 3 to the right. The `min_count` parameter was set to 2, excluding from the vocabulary tokens that appeared only once in the corpus. The SVM classifier used a polynomial kernel of degree 1 with an inverse regularization of 0.1. The model achieved a validation balanced accuracy of 90.32%, a training balanced accuracy of 89.88%, and a test balanced accuracy of 82.74%.

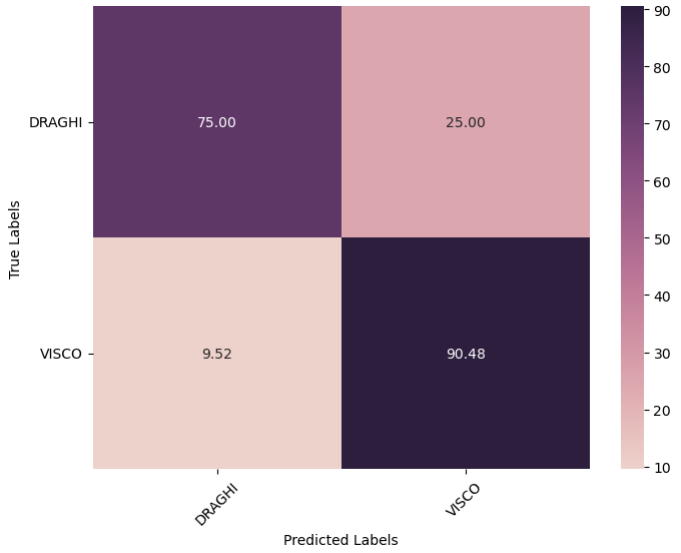


Figure 4.10. Test confusion matrix for the English Doc2Vec SVM model.

As shown in Figure 4.10, the confusion matrix reveals that only 1 out of 4 of Draghi's documents was misclassified, while 19 out of 21 of Visco's documents were correctly classified.

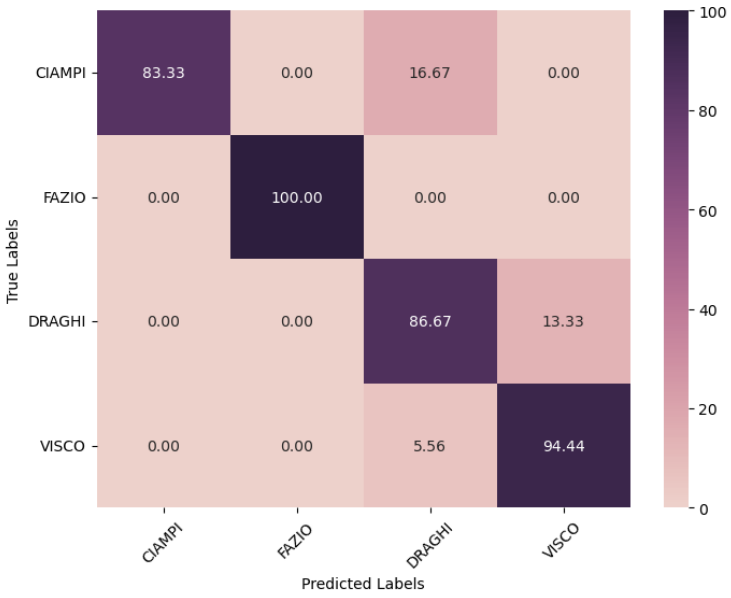


Figure 4.11. Test confusion matrix for the Italian Doc2Vec SVM model.

For the classifier trained on Italian documents, the optimal model used Doc2Vec embeddings with a `vector_size` of 150, a `window` size of 4 (the largest value tested), and the `min_count` parameter was set to 2. The SVM model was trained with a radial basis function (RBF) kernel, with a gamma of 0.001 and an inverse regularization strength `C` of 10. This model obtained a validation balanced accuracy of 95.17%, a training balanced accuracy of 98.27%, and a test balanced accuracy of 91.11%.

As depicted in Figure 4.11, the confusion matrix reveals that all of Fazio's test set documents were correctly classified, while nearly all of Visco's, Ciampi's, and Draghi's documents were also predicted accurately.

Finally, for the classifier trained on the combined dataset of English and Italian documents, the best model was based on Doc2Vec embeddings with a `vector_size` of 250, a `window` size of 3, and `min_count` set to 1, meaning that all words from the training set were included in the vocabulary. The SVM classifier employed an RBF kernel with a gamma of 0.001 and an inverse regularization strength `C` of 10. This model achieved a validation balanced accuracy of 90.93%, a training balanced accuracy of 96.88%, and a test balanced accuracy of 78.71%.

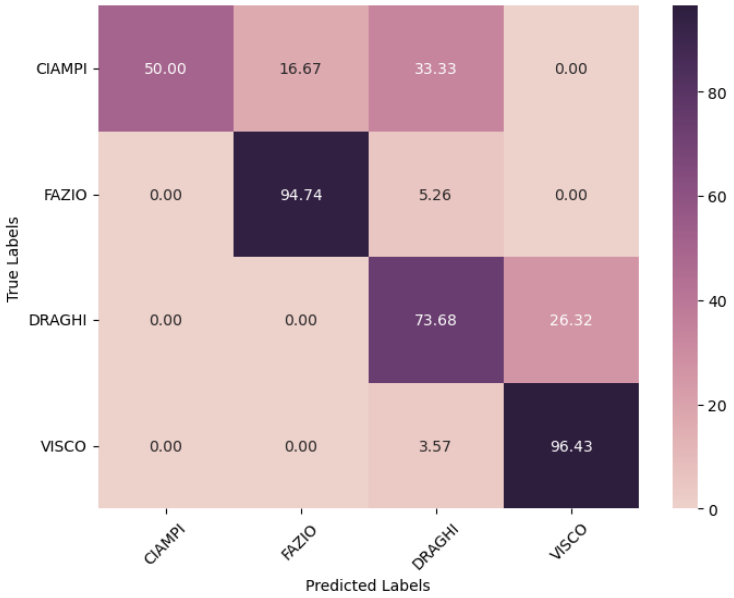


Figure 4.12. Test confusion matrix for the English-Italian Doc2Vec SVM model.

As seen in Figure 4.12, most of Fazio's and Visco's documents were correctly classified. On the other hand, 26.32% of Draghi's documents were mistaken for Visco's, and only 50% of Ciampi's documents were accurately classified, with the remainder incorrectly attributed to Fazio and Draghi.

Multiple Logistic Regression

Using `GridSearchCV`, we optimized the hyperparameters for the logistic regression classifier, similar to the approach used for the SVM. Here, we present the results of the best-performing models for the three datasets: English-only, Italian-only, and the combined English-Italian dataset.

For the classifier trained exclusively on the English documents, the best model used the Doc2Vec embeddings with a `vector_size` of 150. The `window` parameter was set to 3, and `min_count` was also set to 3, discarding words appearing fewer than 3 times. The optimal logistic regression classifier employed an inverse regularization of 1. The model achieved a validation balanced accuracy of 88.74%, a training balanced accuracy of 93.58%, and a test balanced accuracy of 85.12%.

As shown in Figure 4.13, the confusion matrix reveals that only 1 out of 4 of Draghi's documents was misclassified, and approximately 95% of Visco's test documents were correctly classified.

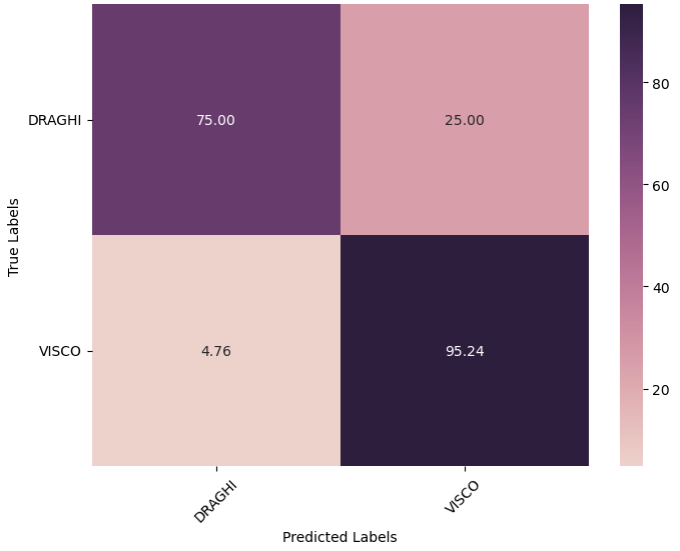


Figure 4.13. Test confusion matrix for the English Doc2Vec logistic regression model.

For the classifier trained on the Italian documents, the optimal model was based on Doc2Vec embeddings with a `vector_size` of 250, a `window` size of 2, and a `min_count` of 2. The logistic regression model used an inverse regularization strength of 0.1. This model achieved a validation balanced accuracy of 93.94%, a training balanced accuracy of 98.89%, and a test balanced accuracy of 90.49%.

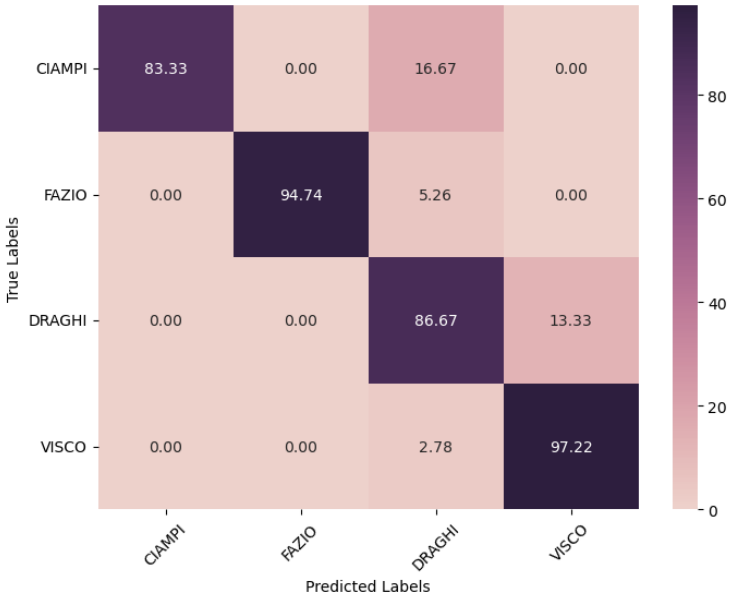


Figure 4.14. Test confusion matrix for the Italian Doc2Vec logistic regression model.

Figure 4.14 illustrates the confusion matrix for this model, showing that Visco's and Fazio's test documents were almost entirely correctly classified. The model performed reasonably well on Ciampi's and Draghi's documents. Except for Draghi, all misclassifications were attributed to Draghi.

For the classifier trained on the combined English-Italian dataset, the optimal model used Doc2Vec embeddings with a `vector_size` of 250, a `window` size of 3, and a `min_count` of 2. The logistic regression classifier employed an inverse regularization strength of 0.1. This model achieved a validation balanced accuracy of 90.03%, a training balanced accuracy of 97.43%, and a test balanced accuracy of 78.71%.

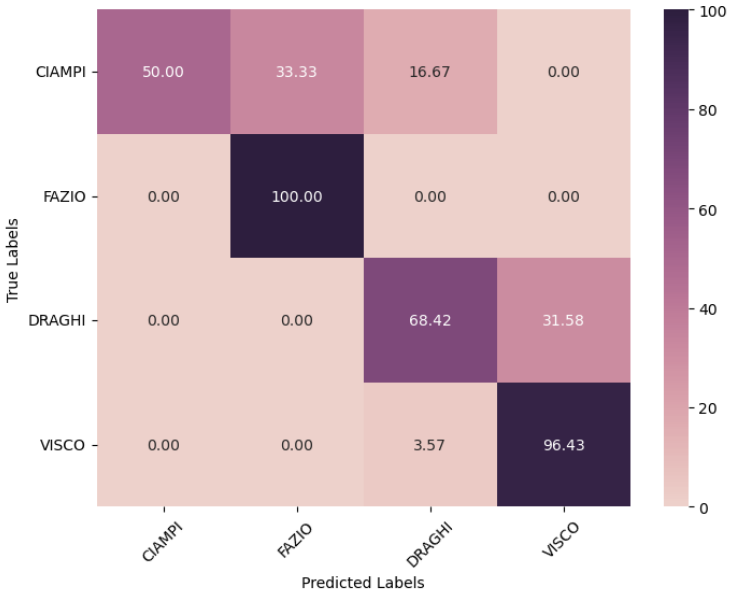


Figure 4.15. Test confusion matrix for the English-Italian Doc2Vec logistic regression model.

As depicted in Figure 4.15, all of Fazio's documents were correctly classified. However, Draghi's documents were misclassified about 32% of the time, with all errors attributed to Visco's class. Similar to the SVM model, only 50% of Ciampi's documents were accurately classified, with the remaining half misclassified between Fazio and Draghi.

Table 4.1 recaps the optimal hyperparameters for the six Doc2Vec models. Models with vector sizes of 50 or 100 were never selected. Instead, longer embeddings were preferred, likely because they better capture the semantic richness of the documents, particularly in the combined English-Italian dataset. Regarding the `window` size, a value of 3 was chosen in four out of the six models, meaning each word was contextualized by three words on either side, for a total of six context words during training. The `min_count` parameter was set to 2 in four of the models, indicating that words appearing only once were excluded from the vocabulary. These words most likely include typographical errors and rare terms. However, increasing this parameter further could harm the learning process, as even infrequent words can provide meaningful context for understanding other terms, which is why stop words were not removed either.

Dataset	Model	vec_size	window	min_count	C
English	SVM	200	3	2	0.1
	LR	150	3	3	1
Italian	SVM	150	4	2	10
	LR	250	2	2	0.1
English-Italian	SVM	250	3	1	10
	LR	250	3	2	0.1

Table 4.2. Hyperparameters of the best English, Italian and multilingual Doc2Vec models.

4.1.4 Transformer-based Fine-tuned Models

To fine-tune the three transformer-based models for classifying the Governors, we employed the RoBERTa architecture, using different versions for each document corpus:

- `FacebookAI/roberta-base` for the English dataset,
- `osiria/roberta-base-italian` for the Italian dataset,
- and `FacebookAI/xlm-roberta-base` for the multilingual dataset.

The first step in fine-tuning these models was preparing the data. For each dataset—English, Italian, and the combined English and Italian documents—we split the data into an 80% training set and a 20% test set, using the same random seed across all datasets to ensure fair comparison with the classification models from the previous two subsections. The training set was then further divided into two parts: 70% of the total data used for training and 10% for validation. This validation set was crucial for monitoring how well the models generalized to unseen data during training, helping to prevent overfitting, typical in deep learning due to the large number of parameters.

As noted in Section 3.2.3, these models have a maximum token limit of 512 per input sequence. Consequently, longer documents would be truncated, posing the risk of losing valuable information for classification. To deal with this problem, the documents were split into smaller chunks of 128[3] tokens or fewer. Each chunk was labelled with the same label as the original document, ensuring that all chunks from a document remained together and were not spread across the training, validation, and test sets.

No data cleaning was necessary. Moreover, all three models are *case-sensitive* meaning they distinguish between “english” and “English”.

The English Model

For the English dataset, the 121 original documents were split into 2273 document chunks: 1613 for training, 228 for validation, and 432 for testing. The distribution of labels between the two Governors (Draghi labelled as 0 and Visco as 1) was:

- The training set contained 248 Draghi chunks and 1365 Visco chunks.
- The validation set contained 25 Draghi chunks and 203 Visco chunks.
- The test set contained 59 Draghi chunks and 373 Visco chunks.

[3]The maximum document length was set to 128 due to due to GPU memory constraints during training.

After chunking the data, the next step involved tokenization. For each dataset, we used the tokenizer corresponding to each model. Tokenization was performed using the `from_pretrained` method of the `AutoTokenizer` class from the `transformers` library, with the `pretrained_model_name_or_path` parameter set to `roberta-base`. During tokenization, padding was applied to ensure uniform length across sequences, and the `max_num_tokens` parameter was set to 128. Truncation was unnecessary, as no sequence exceeded this length. Once tokenized, the data was converted into TensorFlow datasets by first transforming the labels into TensorFlow tensors using the `convert_to_tensor` function and then using the `from_tensor_slices` function from the `tf.data.Dataset` module to create the final dataset.

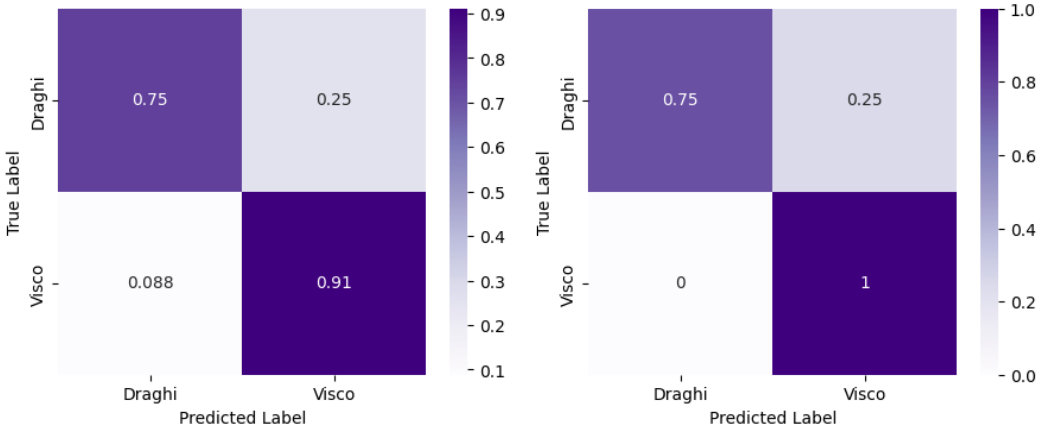
Now that the data has been prepared, we can proceed to configure the models for fine-tuning. The English base model was imported using the `TFAutoModelForSequenceClassification` class of the `transformer` library, specifying the `num_labels` parameter as 2, corresponding to a binary classification task (Draghi or Visco). The model automatically attaches a *classification head*, which performs a linear transformation on the final hidden layer of size 768 to output predictions for the two Governors. The additional parameters introduced by this classification layer are 1538, consisting of 768 weights for each class plus 2 biases. These classification weights are randomly initialized, and a seed must be set to ensure reproducibility. Thus, these additional parameters, along with the pre-trained model's existing weights, were trained during the fine-tuning process.

To train the model, we conducted a grid search to find the optimal combination of *learning rate* and *dropout rate*. Each model trained with a different combination of hyperparameters followed the same procedure:

- First, the base model was augmented by adding a `tf.keras.layers.Dropout` layer to the original classifier layer, accessible via `model.classifier`.
- To reduce the number of trainable parameters and mitigate overfitting, we *froze* the first six encoder blocks, keeping the last six encoder blocks and the classifier layer trainable. This decision was based on the fact that earlier layers typically capture more general features, while later layers focus on task-specific features. We achieved this by setting `layer.trainable` to `False` for the first six encoder blocks.
- The model was *compiled* using the `compile` method, where we specified `AdamW` as `optimizer`—"a stochastic gradient descent method that is based on adaptive estimation of first-order and second-order moments with an added method to decay weights" Chollet et al. (2015)—with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and an L2 weight decay of 1e-2. Moreover, the loss function was set to a manually defined weighted `sparse_categorical_crossentropy` loss from the `tf.losses` module. The weights were computed using the scikit-learn `compute_class_weight` function. The loss function was weighted to emphasize errors in the minority class (Draghi) more heavily than in the majority class (Visco), to avoid favoring the majority class at the expense of the minority class.
- Training was carried out using the `fit` method, where the following inputs were specified:
  - Training and validation data.
  - Batch size (`batch_size`): 32.
  - Number of epochs (`epochs`): 40.

- AdamW learning rates (`learning_rate`): 5e-6, 7e-6, 1e-5 2e-5. The learning rates were chosen to be small enough to avoid wiping out the knowledge of the pre-trained base model.
- Dropout rates (`dropout_rate`): 0.3, 0.5, 0.7, 0.8. Dropout is a well-known regularization technique that helps to prevent overfitting by randomly deactivating (dropping out) certain neurons and their connections during training, such that they no longer contribute to the forward and backward passes. Each neuron has a probability of being "turned off", corresponding to the dropout rate. Higher rates impose stronger regularization.
- Callbacks (`callbacks`): the Early Stopping (`EarlyStopping`) callback was used to monitor the loss computed on the validation set and stop the optimization process if the loss did not improve after a certain number of epochs. This amount of time is called *Patience*, here set to 5. At the end of the optimization process, the best model weights are saved.

After training, the validation balanced accuracy was computed and stored for each model. The best model, i.e. the one yielding the highest validation balanced accuracy, was the one trained with a learning rate of 7e-6 and a dropout rate of 0.3. The optimization process stopped after 23 epochs, with the best model weights restored from epoch 18. The model achieved a validation loss of 0.1178, a validation accuracy of 95.01%, a training accuracy of 97.76%, and a test accuracy of 82.86%.



(a) (b)

Figure 4.16. On the left (a) the test confusion matrix on the English document chunks, on the right (b) the confusion matrix on the whole English documents.

However, to directly compare these results with the TF-IDF and Doc2Vec models, we needed to aggregate the chunk-level predictions back to the original documents. We determined the prediction for each document by selecting the most common prediction across its chunks. This method is advantageous because some document chunks may not be representative of the Governor, but most of the chunks could still reflect the correct label. Non-representative chunks could include content like footnotes, titles, or other irrelevant parts of the document. Therefore, using chunk-level results alone wouldn't provide a fair comparison. After aggregating the chunk predictions, the final balanced accuracies for the full documents are 99.31% for the

training set, 100% for the validation set, and 87.50% for the test set.

Figure 4.16 displays two confusion matrices: on the left, the confusion matrix for the classification of the English document chunks, and on the right, the confusion matrix for the classification of the entire English documents. The right subplot highlights that only 1 out of 4 of Draghi's documents was misclassified, with all other documents correctly classified.

The Italian Model

For the Italian dataset, 377 original documents were split into 12284 chunks: 8926 for training, 894 for validation, and 2464 for testing. The label distribution among the four Governors (Ciampi labelled as 0, Fazio as 1, Draghi as 2, and Visco as 3) was:

- The training set contained 699 Ciampi chunks, 2501 Fazio chunks, 1729 Draghi chunks, and 3997 Visco chunks.
- The validation set contained 67 Ciampi chunks, 258 Fazio chunks, 201 Draghi chunks, and 368 Visco chunks.
- The test set contained 202 Ciampi chunks, 492 Fazio chunks, 494 Draghi chunks, and 1276 Visco chunks.

The data preparation process mirrored that used for the English model. The only difference involved the `pretrained_model_name_or_path` parameter, which was set to `osiria/roberta-base-italian`. The Italian base model was imported using the `TFAutoModelForSequenceClassification` class, with the `num_labels` parameter set to 4 to reflect the four Governors. In this case, the classification head added 3076 parameters, comprising of 768 weights for each class and 4 bias terms. The training process followed the same procedure as described for the English model.

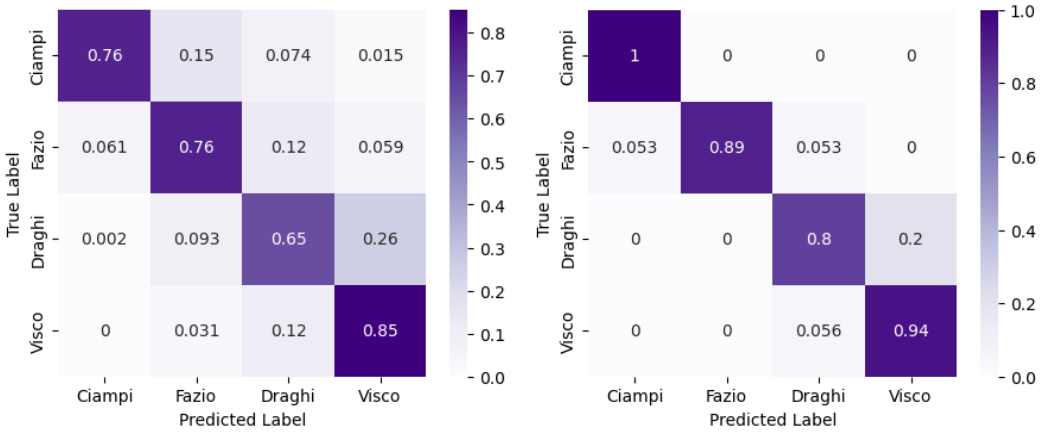
The best model had a learning rate of 1e-5 and a dropout rate of 0.8. The optimization process stopped after 39 epochs, and the best model weights were restored from epoch 34. The model achieved a validation loss of 0.6893, a validation accuracy of 75.71%, a training accuracy of 96.43%, and a test accuracy of 75.49%.

As with the English dataset, chunk-level predictions were aggregated to produce predictions for the entire document. This aggregation resulted in the following balanced accuracies for the complete documents: 99.43% for the training set, 84.23% for the validation set, and 90.98% for the test set.

Figure 4.17 presents two confusion matrices: the left side shows the classification results for the Italian document chunks, while the right side displays the results for the entire Italian documents. From the plot on the right, we can conclude that all 6 of Ciampi's documents were correctly classified. For the other Governors, 17 out of 19 Fazio's documents were correctly classified, 3 out of 15 Draghi's documents were misclassified, and 34 out of 36 Visco's documents were correctly classified. Interestingly, the misclassification errors were not random but tended to be assigned to the temporally closest Governors. For instance, 20% of Draghi's documents were erroneously attributed to Visco, and 10.6% of Fazio's documents were misclassified, split evenly between Ciampi's and Draghi's classes.

The Multilingual Model

For the combined dataset of English and Italian documents, 499 original documents were split into 14587 chunks: 10885 for training, 1089 for validation, and 2613 for



(a) **(b)**

Figure 4.17. On the left (a) the test confusion matrix on the Italian document chunks, on the right (b) the confusion matrix on the whole Italian documents.

testing. The label distribution among the four Governors was as follows:

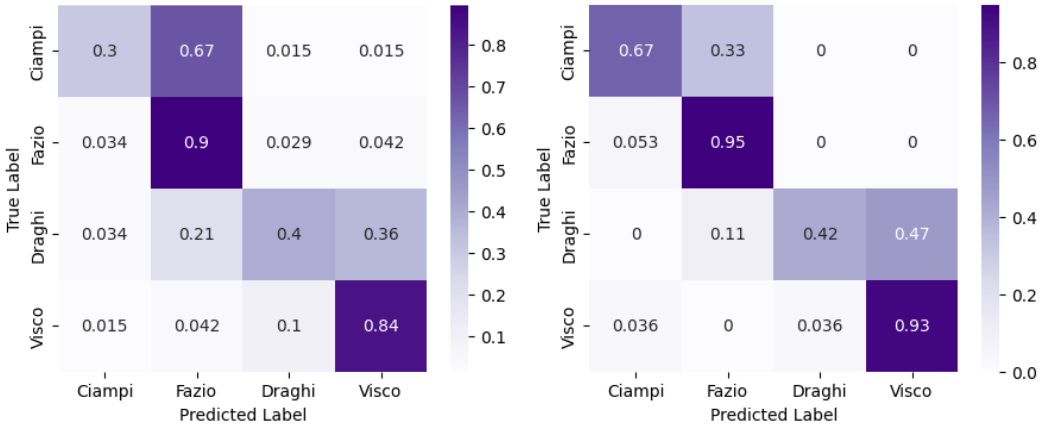
- The training set contained 663 Ciampi chunks, 2459 Fazio chunks, 2197 Draghi chunks, and 6556 Visco chunks.
- The validation set contained 71 Ciampi chunks, 268 Fazio chunks, 153 Draghi chunks, and 597 Visco chunks.
- The test set contained 264 Ciampi chunks, 524 Fazio chunks, 406 Draghi chunks, and 1419 Visco chunks.

The data preparation and model training followed the same process as for the previous two models. However, for this task, the `pretrained_model_name_or_path` parameter was set to `FacebookAI/xlm-roberta-base`, a multilingual model, and the batch size was reduced from 32 to 16, due to the larger number of parameters. The XLM-RoBERTa model has 270 million parameters, which is more than double the 125 million parameters of the previous two models. Despite the larger model size, the hidden dimension remained 768, and the classification head added the same 3076 parameters (768 weights per class and 4 biases).

The optimal model was found with a learning rate of 5e-6 and a dropout rate of 0.7. Training stopped after 28 epochs, with the best model weights restored from epoch 23. The model achieved a validation loss of 0.4441, a validation accuracy of 82.92%, a training accuracy of 87.35%, and a test accuracy of 60.96%.

After aggregating the chunk-level predictions, the balanced accuracies for the complete documents were 97.42% for the training set, 85.23% for the validation set, and 74.09% for the test set. These results indicate that this model tended to overfit more compared to the previous two models, likely due to its higher number of parameters.

Figure 4.18 displays two confusion matrices: on the left (a), the results for the document chunks; on the right (b), the results for the entire documents. The right-hand subfigure shows that, while Fazio and Visco were classified well, the model struggled with Ciampi's and Draghi's documents, the two smaller classes in the dataset. This was particularly true for Draghi, whose documents were more often assigned to Visco's class than to its own.



(a) (b)

Figure 4.18. On the left (a) the test confusion matrix on the combined English and Italian document chunks, on the right (b) the confusion matrix on the whole combined English and Italian documents.

From Table 4.3, it is evident that separating the English and Italian documents and classifying them individually produced better results than combining them into a single dataset classified with a multilingual model.

Dataset	lr	dr	epochs	train acc	val acc	test acc
English	7e-6	0.3	23	99.31%	100.00%	87.50%
Italian	1e-5	0.8	39	99.43%	84.23%	90.98%
English-Italian	5e-6	0.7	28	97.42%	85.23%	74.09%

Table 4.3. Results and hyperparameters (learning rate, dropout rate and number of epochs) of the best English, Italian and multilingual models.

4.1.5 Model Comparison

Tables 4.4 and 4.5 present the balanced test accuracies for all models discussed so far. For the English-only dataset, the TF-IDF and Transformer-based models achieved an accuracy of 87.50%. All 21 of Visco's documents were correctly classified, while 1 out of 4 of Draghi's documents was misclassified. This misclassification consistently involved the same document, namely Draghi's 2010 opening remarks at the "2nd EFIGE Scientific Workshop and Policy Conference"[4] on June 18. When using Doc2Vec embeddings, the models performed slightly worse, misclassifying not only the same Draghi document but also some of Visco's documents.

For the Italian-only dataset, the SVM with Doc2Vec embeddings outperformed the other models, achieving the highest balanced test set accuracy of 91.11%.

For the combined dataset of English and Italian documents, the highest accuracy, 86.62%, was achieved by the TF-IDF logistic regression model. In contrast, the

[4] https://www.bancaditalia.it/pubblicazioni/interventi-governatore/integov2010/draghi_-efige_180610.pdf.

performance of the Doc2Vec and Transformer-based models significantly worsened when applied to the multilingual dataset.

Specifically, the XLM-RoBERTa Transformer, which was pre-trained on 100 languages, is not specialized just for English and Italian. Given that our dataset consists only of documents in these two languages, separating them prior to classification and using monolingual models results in higher balanced accuracy scores, as it is observable in Table 4.5.

On the other hand, poor performance of the Doc2Vec models on the English-Italian dataset could be attributed to the way embeddings are generated. In these models, words are embedded based on their context within documents of the same language. This means that semantically similar words in English and Italian may be represented very differently in the vector space, contributing to lower accuracies in the multilingual case.

One can conclude that classifying English and Italian documents separately leads to better results, even though it reduces the amount of training data available for each classifier.

We also experimented with resampling techniques to address the class imbalance in the datasets. Specifically, we tested oversampling and undersampling methods. For oversampling, we implemented the SMOTE (Synthetic Minority Over-sampling Technique), which creates synthetic samples of the minority classes until they match the size of the majority class. This was done using the `SMOTE` class from the `imblearn.over_sampling` module. For undersampling, we reduced the size of the majority classes by randomly removing instances until they matched the size of the minority class. This was accomplished using the `RandomUnderSampler` from the `imblearn.under_sampling` module. While these techniques did not improve performance in our case, it is important to document failed attempts to keep track of what worked and what did not.

Dataset	Model	TF-IDF	Doc2Vec
English	SVM	**87.50**%	82.74%
	LR	**87.50**%	85.12%
Italian	SVM	87.16%	**91.11**%
	LR	84.52%	90.49%
English-Italian	SVM	86.18%	78.71%
	LR	**86.62**%	78.71%

Table 4.4. Balanced test set accuracies of TF-IDF and Doc2Vec classification models.

Dataset	Model	Accuracy
English	`FacebookAI/roberta-base`	87.50%
Italian	`osiria/roberta-base-italian`	90.98%
English-Italian	`FacebookAI/xlm-roberta-base`	74.09%

Table 4.5. Balanced test set accuracies of Transformer-based classification models.

4.2 Topic Modeling

The second major task of this work involves identifying topics within our three document collections: English-only, Italian-only, and the combined dataset of all documents. To perform topic modeling, we initially employed a widely popular method known as *Latent Dirichlet Allocation* (LDA), introduced in Blei et al. (2003). "LDA is a generative probabilistic model of a corpus. The basic idea is that the documents are represented as random mixtures over latent topics, where a topic is characterized by a distribution over words. [...] The words with the highest probabilities in each topic usually give a good idea of what the topic is" Jelodar et al. (2019). In practice, we used the `LatentDirichletAllocation` class from the `sklearn.decomposition` module. This class allows for several inputs, such as the desired number of topics (`n_components`), the maximum number of iterations (`max_iter`) for the optimization process, and the `learning_method`, which determines whether batch or online gradient descent is used for optimization. Once the LDA model is trained, the `pyLDAvis` Python tool can be utilized to create interactive topic model visualizations. However, this method produced disappointing results on our corpus. Despite experimenting with different values of `n_components`, the resulting topics were consistently mixed and lacked clear distinctions, making it challenging to identify meaningful topics. This issue likely arose because LDA requires specifying a fixed number of topics in advance, and it enforces this number even if fewer distinct topics are present in the data. As a result, we decided to switch to a more recent, state-of-the-art topic modeling technique known as BERTopic.

BERTopic was implemented using the `BERTopic` class from the `bertopic` library. As explained in Section 3.3, this topic modeling algorithm generates topic representations through the following sequence of steps:

- First, Sentence Transformers are used to create semantically meaningful document embeddings. These models are accessible through the `SentenceTransformer` class from the `sentence_transformer` library. For English documents, we utilized the `all-MiniLM-L6-v2` model, and for Italian documents, we used the `paraphrase-multilingual-MiniLM-L12-v2` model. Both models have token limits—256 tokens for the English model and 128 tokens for the Italian model. To handle documents exceeding these limits and to avoid truncating potentially valuable information, we split the documents into chunks of 128 tokens or fewer. Before splitting, a light text cleaning process was performed, which involved lowercasing the text and removing punctuation, dates, and numbers to ensure cleaner token representations. Consequently, from the original 122 English documents, we obtained 2223 chunks, and from the 377 Italian documents, we obtained 11463 chunks. For the combined English-Italian dataset, we chose to merge the results from the English and Italian documents separately rather than running BERTopic directly on the combined dataset. The Sentence Transformer models were passed to the `embedding_model` parameter of the `BERTopic` instance.
- After obtaining document chunk embeddings with a vector size of 384[5], we reduced the dimensionality using `UMAP`. This was achieved by instantiating a

[5]Both Sentence Transformer models have a hidden dimension of 384.

`umap.UMAP` object, which is passed to the `umap_model` parameter in BERTopic. For this step, we set the number of neighbors (`n_neighbors`) to 15, the reduced embedding dimension (`n_components`) to 2, the `min_distance` to 0 to facilitate clustering, the similarity metric (`metric`) to "`cosine`", and a random seed (`random_seed=111`) to ensure the reproducibility of results.

- Once the dimensionality was reduced, we clustered the data using the `HDBSCAN` class from the `hdbscan` library, which was passed to the `hdbscan_model` parameter in `BERTopic`. Within `HDBSCAN` we set `min_cluster_size` to 5, discarding all clusters smaller than this threshold. The `metric` was set to "`euclidean`", as recommended by the BERTopic documentation, and the `cluster_selection_method` was specified as "`eom`" (Excess of Mass).
- Once the optimal clusters were identified, the next step involved assigning a topic to each cluster. This was done by gathering all document chunks within a cluster and concatenating them to form longer documents. On these larger texts, we computed a bag of words (BOW) using the `CountVectorizer` class from `sklearn.feature_extraction.text`. This was passed to the `vectorizer_model` parameter. For the `CountVectorizer`, we specified the following parameters: `analyzer` was set to "`word`" to ensure word-level tokenization, the n-gram range (`n_gram`) was set to $(1, 2)$ to include both unigrams and bigrams and the `stop_words` were specified using the same stopword lists (both Italian and English) as those created for the TF-IDF embeddings in Section 4.1.2.
- The topics were then identified using Class-TF-IDF on the clusters found by HDBSCAN. To perform this, we passed the `ClassTfidfTransformer` object from the `bertopic.vectorizers` module to the `ctfidf_model` parameter, with the `reduce_frequent_words` option set to "`True`" to reduce the impact of overly frequent words. For each topic, we extracted the top 30 words by setting the `top_n_words` parameter (which defaults to 10).

We then fit the `BERTopic` instance to the document chunks using the `fit_transform` method. Finally, the number of topics was reduced by applying the `reduce_topics` method, setting the `nr_topics` parameter to "`auto`". This process involves running HDBSCAN clustering on the c-TF-IDF representations of the topics. Topics deemed similar, i.e. those grouped into the same cluster, are merged, while topics that are classified as noise, i.e. those that do not belong to any cluster, remain unchanged.

4.2.1 Main Results

For the English documents, BERTopic identified 28 topics, excluding outliers labelled as `-1`. Each topic is represented by up to 30 unigrams or bigrams, ranked in descending order based on their c-TF-IDF scores, which indicate their relevance to the topic. Information about these topics, including their names, representative words, and the number of document chunks contributing to each topic, can be accessed using the `get_topic_info` method from the trained `BERTopic` instance. These details are summarized in Table 4.6.

Topic	Count	Topic Representation (top 2 tokens)	Gov
-1	458	monetary policy, inflation	
0	1122	monetary policy, inflation expectations	D,V
1	119	technological unemployment, technological progress	D,V
2	87	climaterelated financial, sustainable finance	D,V
3	43	italian households, wealth italy	V
4	41	macroeconomics, economics	V
5	32	payments euro, instant payments	V
6	31	payments sector, payments ecosystem	D,V
7	25	poverty count, extreme poverty	V
8	24	esg rating, esg score	V
9	23	consequences pandemic, countries pandemic	V
10	21	financial literacy, improve financial	V
11	19	international trade, multilateral trade	D,V
12	18	developing countries, development committee	D,V
13	14	islamic finance, islamic financial	D,V
14	14	asset managers, asset management	D,V
15	14	transactions europe, negotiations eurosystem	D,V
16	13	ene, pte toni	D,V
17	13	economic historians, economic history	V
18	12	gdp disposable, dynamics gdp	V
19	11	cyber defence, cybersecurity	V
20	11	liquidity investors, hedge funds	D
21	11	digital financial, financial inclusion	V
22	11	crossborder payments, payments international	V
23	8	exchanges competitive, trading systems	D,V
24	8	sharing ibrd, ibrd shareholding	D,V
25	7	population aged, europe ageing	V
26	7	fintech endevour, italy innovation	V
27	6	unemployment rate, italy unemployment	V

Table 4.6. English topics info in descending order of Count, where Count represents the topic size, D stands for Draghi and V for Visco.

The table shows that 458 out of the 2223 document chunks were categorized as noise, while most (1122) of the remaining chunks (1765) were grouped into `Topic0Eng`, which revolves around themes of monetary policy and inflation. The other document chunks are distributed across the remaining 27 clusters[6], with `Topic16Eng` representing a cluster of typographical errors, fortunately, grouped together rather than polluting other topic representations. The column labelled "Gov" in the table indicates whether the document chunks contributing to each topic originate from speeches by Draghi, Visco, or both. Except `Topic20Eng`, which relates to hedge fund liquidity, the majority of topics are primarily driven by Visco's documents. These include:

- `Topic9Eng`, which pertains to the Covid-19 pandemic, emerging after Draghi's mandate as Governor.

[6]Note that no topic has Count < 5 due to setting the HDBSCAN `min_cluster_size` parameter to 5. Moreover, bigrams were more frequently chosen to represent topics as opposed to unigrams.

- `Topic19Eng`, related to cybersecurity and defense against cyberattacks, reflecting a growing concern in recent years.
- `Topic7Eng` and `Topic27Eng`, focused on socio-economic issues such as poverty and unemployment.

In several topics where both Draghi and Visco contribute, Draghi's document chunks appear more frequently, despite Visco's overall larger volume of speeches. These topics are:

- `Topic11Eng`, related to global trade and international economics, including discussions on tariffs, quotas, trade barriers, trade liberalization, global supply chains, and the *World Trade Organization* (WTO).
- `Topic14Eng`, focused on investment management, covering areas such as mutual funds, hedge funds, pension fund management, and the asset management industry.
- `Topic16Eng`, related to typographical errors, suggesting that Draghi's English documents contain more typos compared to Visco's.
- `Topic23Eng`, related to financial markets and stock exchanges, including topics like stock exchange competition, trading systems and platforms, and brokerage.

This trend suggests that Draghi's speeches placed more emphasis on financial and economic systems. Conversely, in other topics where both Draghi and Visco contribute, Visco's document chunks appear more frequently. For example:

- `Topic1Eng`, related to the impact of technology on the economy and labor markets.
- `Topic2Eng`, focused on *Sustainable Finance*, which "refers to the process of taking environmental, social, and governance (ESG) considerations into account when making investment decisions in the financial sector, leading to more long-term investments in sustainable economic activities and projects. Sustainable Finance, therefore, applies the concept of sustainable development to financial activities" L'economia per tutti - Banca d'Italia (2022). Similarly, `Topic8`, addressing *ESG ratings*, is another topic frequently discussed by Visco. "Environmental, social and governance (ESG) ratings provide an opinion on a company or financial instrument's sustainability profile or characteristics, exposure to sustainability risks or impact on society and/or the environment" Commission (2022).
- `Topic6Eng`, concerning the payment industry, digital currencies, and innovations in payment systems.
- `Topic13Eng`, focused on *Islamic Banking and Finance* (Shari'ah-compliant finance), which adheres to the moral principles of Islam (Shari'ah).

Figure 4.19 displays the English topic similarity heatmap, generated using the `visualize_heatmap` method of the trained `BERTopic` model. This heatmap visualizes the cosine similarity (ranging between -1 and $+1$) matrix between topic embeddings, with topics ordered by their frequency in the document set. The heatmap reveals a diverse range of topics, with some, like `Topic6Eng` and `Topic22Eng` both related to payments, showing high similarity (75%), and others, like `Topic20Eng` on hedge fund liquidity and `Topic25Eng` on demographic trends, showing low similarity (18%). Despite this, the similarity values are all positive, reflecting that, while the topics vary, they are not entirely different. This is expected given that the topics are related to financial and socio-economic issues pertinent to the Bank of Italy and its sphere

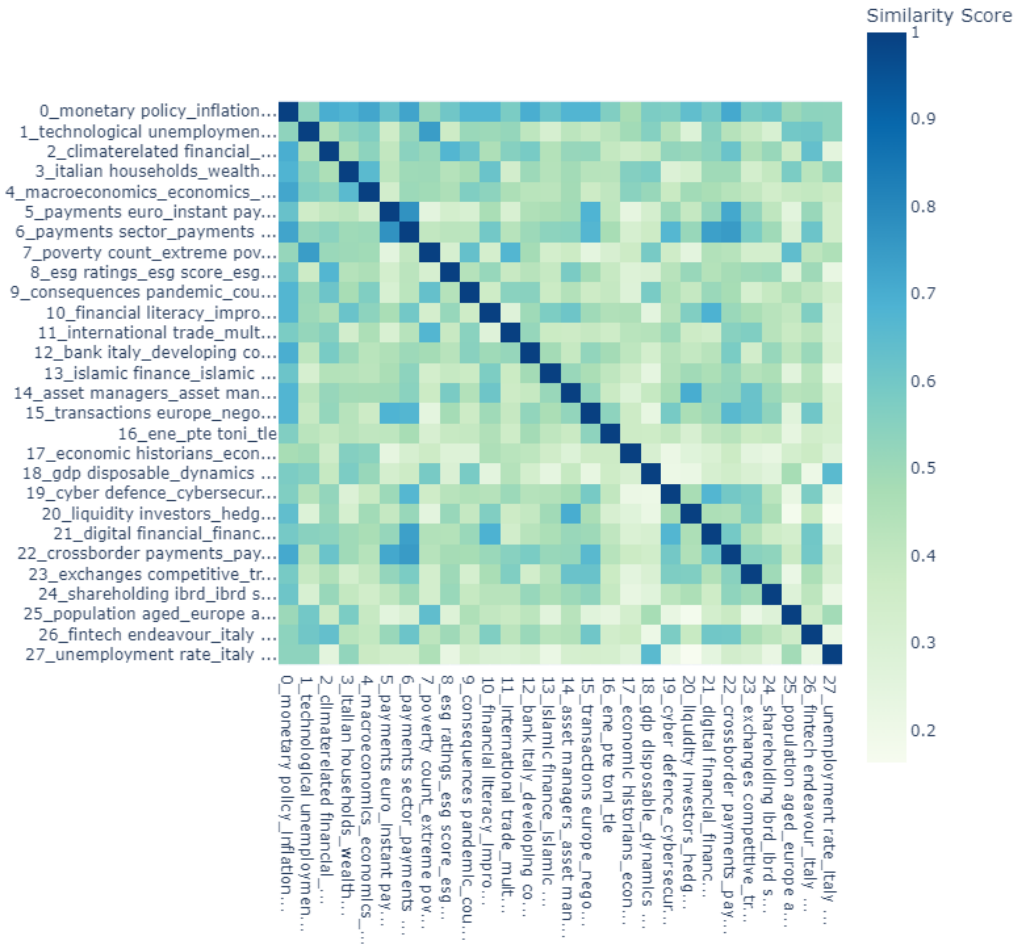


Figure 4.19. English topics heatmap.

of influence. Hence, our topics cannot be completely dissimilar, unlike topics from a broader source such as a newspaper, which covers a much wider range of subjects like political events, crime, business, sports, and the weather.

Interpreting all the topics identified by the BERTopic algorithm can be tedious, and some topics might be better merged to form higher-level themes. To address this, we can exploit *Hierarchical Topic Modeling*, which is directly implemented in the `bertopic` library. We can use the `hierarchical_topics` method of the trained model to extract a dendrogram that visually represents the cluster hierarchy. This dendrogram is constructed by iteratively merging the closest topic pairs, where the proximity is measured by the cosine similarity between the c-TF-IDF topic representations. The final representation of each merged topic is obtained by summing their bag-of-words representations. The dendrogram can be visualized with the `visualize_hierarchy` method as shown in Figure 4.20. The choice of macro-topics is somewhat subjective, as the dendrogram can be "cut" at any point depending on the desired level of detail. For simplicity, we decided to consider as macro-topics the ones identified by different colors in the plot, summarized in Table 4.7.

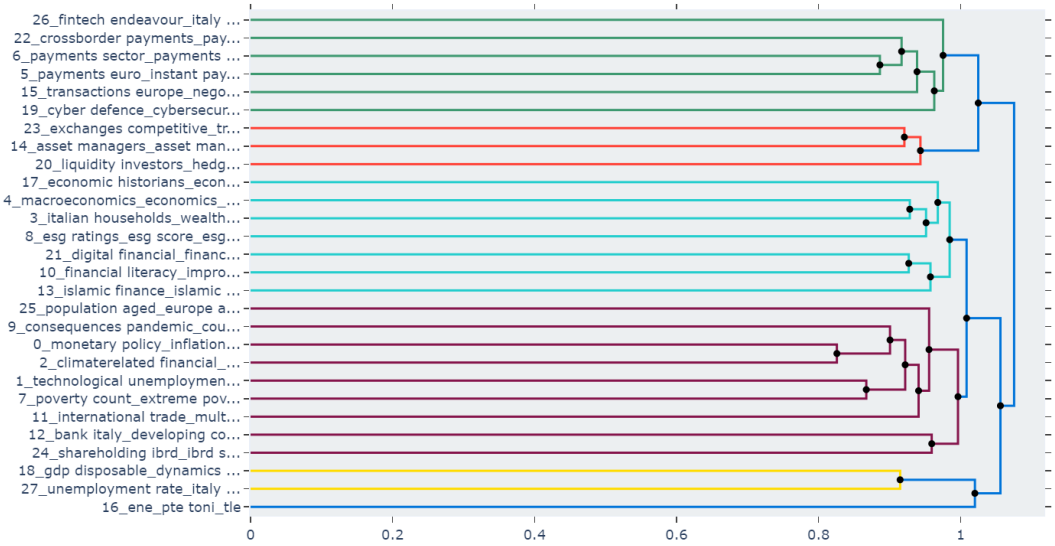


Figure 4.20. English topics hierarchy.

ID	Topics	Macro-topic Representation
1	26,22,6,5,15,19	instant payments, payment infrastructures, digital euro
2	23,14,20	hedge funds, investors hedge, counterparties investors
3	17,4,3,8,21,10,13	economic systems, economics, financial literacy
4	25,9,0,2,1,7,11,12,24	bank italy, central banks, monetary policy
5	18,27	dynamics gdp, disposable income, gdp
6	16	ene, pte toni, tle

Table 4.7. Hierarchical Clustering on English Topics.

Although these macro-topics provide a good starting point, further refinements can be made. Let's analyze each of these six macro-topics:

- `MacroTopic1Eng` revolves around the theme of payments and digital financial infrastructures. However, `Topic19Eng`, which focuses on cybersecurity and cyber defense, seems uncoupled from the other topics $(5, 6, 15, 22)$ unless it is focusing specifically on payment-related cybersecurity. It would be more appropriate to separate it into its own macro-topic. Furthermore, `Topic26Eng` (related to fintech innovation) might be considered distinct enough to form a separate macro-topic or be merged with other technology-focused topics.
- `Macrotopic2Eng` covers asset management, trading, and hedge funds, is well-defined. The included terms are closely related within the financial markets and investment strategies domain. No adjustments are necessary.
- `Macrotopic3Eng` addresses economic systems, economics, and financial literacy. While it is generally well-constructed, `Topic3Eng` (focusing on Italian households and wealth in Italy) is more specific to Italy than global economic systems. `Topic8Eng` which deals with ESG scores, is specific enough to be categorized separately under a "Sustainable Finance" macro-topic. `Topic2Eng`, currently placed in `MacroTopic4Eng`, could also fit well in this sustainable finance category.
- `Macrotopic4Eng` is overly broad, grouping many unrelated topics. For instance,

`Topic9Eng` (on the Covid-19 pandemic), `Topic25Eng` (related to ageing population), and `Topic7Eng` (focused on poverty), all represent socio-demographic trends that do not align with `Topic1Eng` (technological progress) or `Topic0Eng` (monetary policy).

- `Macrotopic5Eng` clusters `Topic18Eng` (on GDP dynamics) and `Topic27Eng` (related to unemployment in Italy), which fit together well, as both topics relate to GDP and income.
- `Macrotopic6Eng` deals with typographical errors. It is correctly separated from the other topics.

We can manually adjust these macro-topics to avoid losing some important topics, as shown in Table 4.8.

ID	Topics	Macro-topic Representation
1	22,6,5,15	instant payments, payment infrastructures, digital euro
2	23,14,20	hedge funds, investors hedge, counterparties investors
3	17,4,21,10,13	economic systems, economics, financial literacy
4	0,11,12,24	bank italy, central banks, monetary policy
5	18,27	dynamics gdp, disposable income, gdp
6	16	typographical errors
7	9,25,7	socio-demographic trends (Covid, population, poverty)
8	2,8	sustainable finance, esg ratings
9	1,26	technological progress, innovation, fintech
10	3	italian households, wealth italy
11	19	cyber defence, cybersecurity

Table 4.8. Manually adjusted English macro-topics.

For the Italian documents, BERTopic identified 56 topics (excluding the noise labelled as `-1`), double the amount compared to the English documents. The topics ranked in descending order of c-TF-IDF scores are summarized in Table 4.2.1.

Table 4.9. Italian topics info in descending order of Count, where Count represents the topic size, C stands for Ciampi, F for Fazio, D for Draghi, and V for Visco.

Topic	Count	Topic Representation (top 2 tokens)	Gov
-1	4602	politica monetaria, banca italia	
0	5660	area euro, euro	C,F,D,V
1	92	erogazione pensioni, schemi pensionistici	C,F,D,V
2	85	restrizioni commercio, equilibri internazionali	C,F,D,V
3	69	dottrina sociale, filosofica	C,F,D,V
4	60	comportamento teorico, modelli empirici	C,F,D,V
5	54	cambiamenti climatici, cambiamento climatico	C,F,V
6	51	deprezzamento yen, cambio yen	C,F,D,V
7	50	sistema educativo, sistema scolastico	F,D,V
8	48	economisti, economista	C,F,D,V
9	37	italia editoria, italia stampa	V
10	33	donne economia, occupazione femminile	F,D,V
11	32	politica migratoria, dinamica demografica	F,V

Continued on next page

Topic	Count	Topic Representation (Top 2 Tokens)	Gov
12	29	confronti innovazione, innovazione competenze	C,F,D,V
13	28	correnti spese, spese netto	F,D,V
14	28	globale rettito, povertà estrema	F,D,V
15	27	ew, sa	C,F,V
16	25	economic review, economic literacy	F,D,V
17	23	participantitenuta roma, partecipanti roma	C,F,D,V
18	23	economia giapponese, economie asiatiche	F,D,V
19	22	economici criminalità, diffusione criminalità	F,D,V
20	22	manovra bilancio, deputati bilancio	F,D,V
21	21	rischi ambientali, rischi climatici	V
22	20	finanziaria scuole, educazione finanziaria	F,D,V
23	19	rischi cibernatici, rischi tecnologici	D,V
24	17	keynesian, teoria keynesiana	C,F,D,V
25	17	monarchi principato, monarchia possibilità	F,V
26	16	dividendo importo, dividendo distribuzione	V
27	15	valuta digitale, euro digitale	D,V
28	15	variazioni mortalità, mortalità età	D,V
29	15	ott fonte, econo	C,D,V
30	13	popolazione italiana, rapporto popolazione	F,D,V
31	13	banche centronord, bancario meridionale	C,F,D
32	12	finanze governatori, ministri finanze	D,V
33	12	partecipanti crisi, roma autorità	F,D,V
34	12	legittimare diritto, diritto naturale	C,F
35	11	struttura occupazionale, european jobs	F,D,V
36	10	collegio sindacale, proposta direttorio	C,F,V
37	9	commissione antimafia, associazioni criminali	C,F,D,V
38	9	finanziari convegno, convenzione interbancaria	C,D,V
39	9	industrializzazione provincia, modernizzazioni settore	F,D,V
40	9	istituzionali garantiscano, legittimità regole	C,F,D,V
41	9	innovazione capacità, confronti innovazione	D,V
42	7	istituzionescuola istituazionebanca, banche	C,D,V
43	6	ripercussioni domanda, inflazione implicazioni	F,V
44	6	livello istruzione, istruzione inferiore	F,D,V
45	6	integrità riservatezza, fiduciarie istituiti	F,D,V
46	6	riflessioni vliaissva	F,D,V
47	6	governare effetti, economico servizio	C,D,V
48	6	ringraziare partecipanti, italia documenti	D,V
49	6	tendenza invecchiamento, demografico frenano	F,V
50	6	società italiana, italia convegno	F,V
51	5	cresciuti tassi, prezzi abitazioni	C,F,D
52	5	agganciare innovazione, disponibilità innovazione	F,D,V
53	5	europea affermazione, unificazione europea	C,F,V
54	5	economia criminale, criminalità economica	V
55	5	conseguenze sociali, politica concerne	F,D,V

The table shows that, out of 11463 document chunks, 4602 were classified as

outliers, while 5660 document chunks construct the main topic cluster, labelled `Topic0Ita`. The latter focuses on monetary policy and the Euro area, akin to `Topic0Eng` in the English model. The other 1201 document chunks are spread across the remaining 55 smaller clusters, many of which are less frequently discussed topics in the Governors' speeches. Clusters `Topic15Ita` and `Topic29Ita` mainly consist of typographical errors, again fortunately grouped together rather than corrupting other topic representations. Among these 56 Italian topics, there are several new topics compared to the English model, along with some topics that are similar but represented by different keywords. The new topics include:

- `Topic6Ita` and `Topic18Ita` are focused on the Japanese yen and its role in global and Asian economies, particularly covering themes such as yen's exchange rate, depreciation, and its value against other major currencies like the U.S. dollar.
- `Topic10Ita` revolves around women's role in the economy and gender equality, with a particular interest in the female employment rate. It is a subject frequently addressed by Visco, reflecting a growing concern in recent years.
- `Topic19Ita`, `Topic37Ita` and `Topic54Ita` are related to organized crime, the Mafia, and criminal networks, with a focus on corruption and anti-money laundering efforts. Both Draghi and Visco contributed significantly to these topics.
- `Topic7Ita`, `Topic22Ita`, and `Topic44Ita` focused on Italy's educational system, education levels in comparison to other countries, and discussions around introducing financial education in schools. This topic appears primarily in Visco's speeches.
- `Topic24Ita` pertains to Keynesian theory[7], a topic predominantly addressed by Governor Fazio in 10 of his document chunks.
- `Topic1Ita` is related to Italy's pension schemes and the pension system. This topic is mainly contributed by Draghi, with approximately 40 document chunks.

In addition to these new topics, several themes appear in both the Italian and English documents, although represented by different keywords. These include:

- `Topic5Ita` and `Topic21Ita`, focused on sustainable finance and climate change, with discussions on global warming, greenhouse gases, and emission reduction. This topic was addressed by Visco in at least 70 document chunks.
- `Topic12Ita`, `Topic41Ita` and `Topic52Ita` concern technology, modernization and innovation.
- `Topic23Ita` addresses cybersecurity and technological risks, with contributions mainly from Visco, like in the English model.
- `Topic27Ita` is focused on digital currency, particularly on the digital Euro.
- `Topic11Ita`, `Topic14Ita`, `Topic28Ita`, `Topic30Ita` and `Topic49Ita` revolve around socio-demographic trends, including discussions on population ageing, migration, demographic decline, mortality rates, and poverty.

The analysis of the 56 Italian topics shows a surprising absence of any reference to the Covid-19 pandemic, which, on the other hand, was identified as a distinct topic in the English model. This could be because Visco's Covid-related speeches were

[7]"Keynesian economists justify government intervention through public policies that aim to achieve full employment and price stability" (Jahan et al., 2014) or prevent economic recessions.

mostly delivered in English or because these mentions were too sparse in Italian to form a significant cluster (remember that `min_cluster_size = 5`).

Another point of interest is the overall quality of the Italian topic representations, which appears to be lower than the English ones. This discrepancy can be attributed to the difference in the embedding models used for each language. Italian topic modeling relies on a multilingual model (`paraphrase-multilingual-MiniLM-L12-v2`), which has not been fine-tuned specifically for Italian. In contrast, the English model (`all-MiniLM-L6-v2`) delivers better results as it is optimized for English. One issue is the frequent appearance of verbs in the Italian topic representations, which weakens clarity. A potential improvement could be to expand the stopwords list to include common verbs among the other frequent tokens, reducing their appearance in the topic representations and thereby improving the overall quality of the clusters.

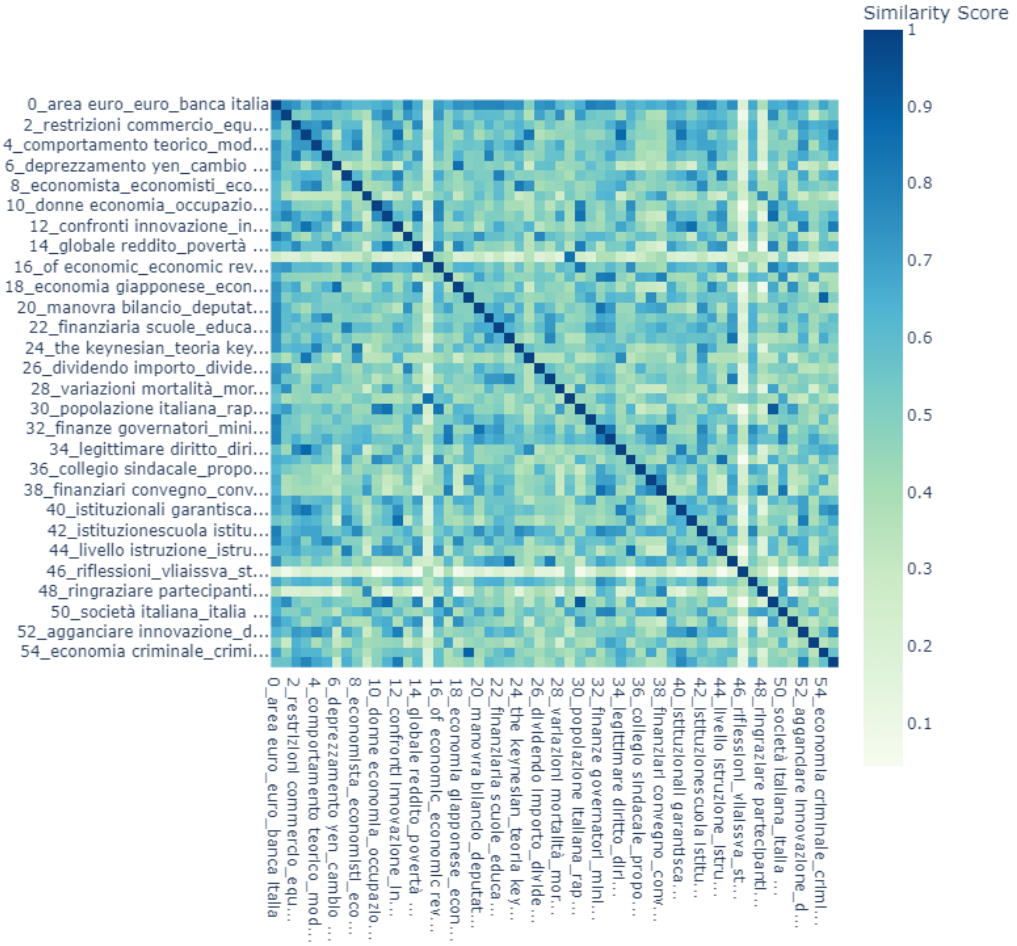


Figure 4.21. Italian topics heatmap.

Figure 4.21 illustrates the similarity heatmap for the Italian topics, calculated akin to the English model. Due to the larger number of topics, only every other topic representation is shown for visualization purposes. The similarity values in the heatmap range from approximately 5% to 95%, excluding the diagonal. The heatmap reveals that some topics are highly similar. For example, `Topic30Ita` and

`Topic39Ita` have a 93% cosine similarity, as both focus on demographic aspects. Conversely, there are topics with little to no correlation. `Topic15Ita`, which contains typographical errors, shows high similarity (85%) only with `Topic29Ita`, another cluster of typographical errors. As for the English model, no negative similarity values are present in the heatmap, indicating that while some topics are unrelated, there are no signs of dissimilarity beyond a lack of correlation.

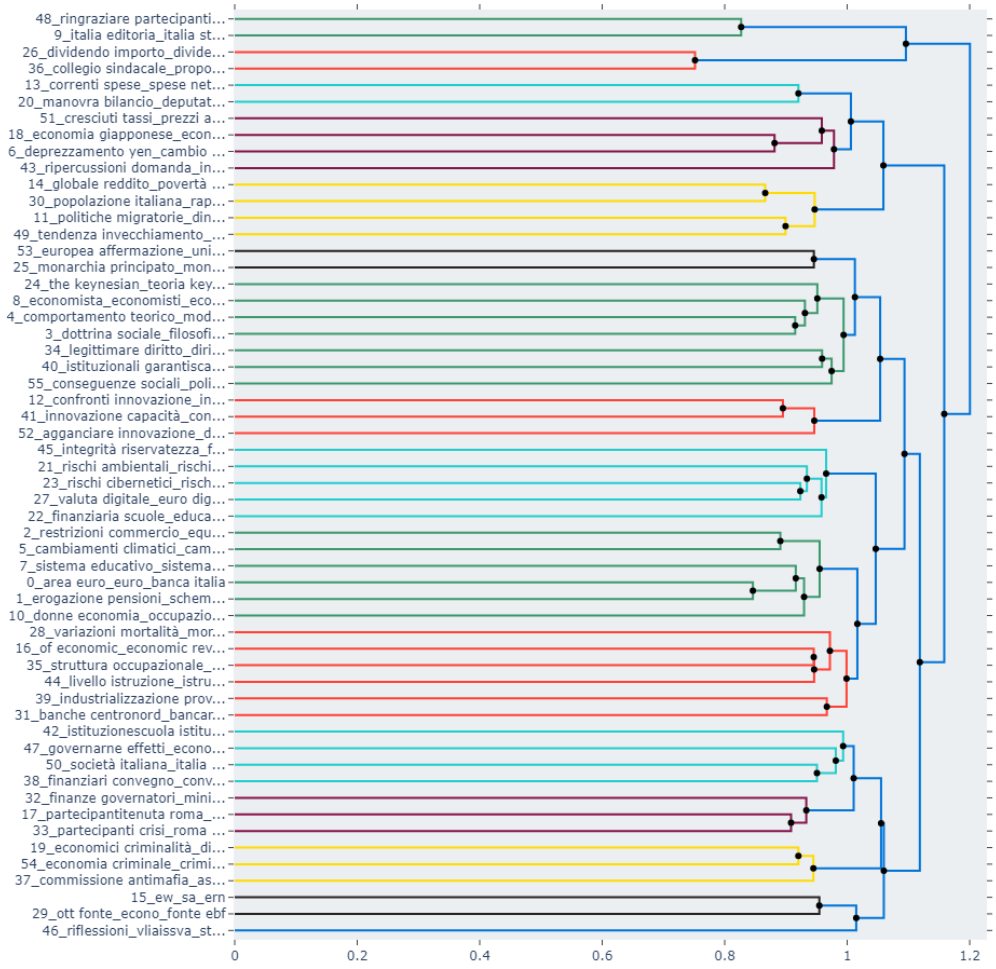


Figure 4.22. Italian topics hierarchy.

By applying the `bertopic.hierarchical_topics` method to the trained Italian `BERTopic` model, we generated the dendrogram depicted in Figure 4.22, leading to the identification of 16 macro-topics, as summarized in Table 4.10. Let's analyze some of these macro-topics in more detail:

- `Macrotopic2Ita` revolves around the theme of dividends and their distribution. The merging of `Topic26Ita` and `Topic36Ita` is appropriate, as both focus on financial dividends.
- `Macrotopic4Ita` addresses the Japanese economy, specifically the depreciation and value of the yen. While `Topic6Ita` and `Topic18Ita` align well with this theme, `Topic43Ita` is related to a more general concept of supply and

ID	Topics	Macro-topic Representation
1	48,9	italia stampa, banca italia
2	26,36	dividendo importo, dividendo distribuzione
3	13,20	spese netto, spese correnti
4	51,18,6,43	deprezzamento yen, valore yen
5	14,30,11,49	riduzione popolazione, povertà estrema
6	53,25	costituzione europea, diritto romano
7	24,8,4,3,34,55,40	teoria economica, modelli econometrici
8	12,41,52	confronti innovazione, innovazione capacità
9	45,21,23,27,22	rischi cibernetici, rischi finanziari
10	2,5,7,0,1,10	politica monetaria, bilancio
11	28,16,35,44,39,31	economic review, economic literature
12	42,47,50,38	italia convegno, società italiana
13	17,32,33	roma autorità, participantitenuta roma
14	19,54,37	economia criminale, diffusione criminalità
15	15,29	ott fonte, sa
16	46	riflessioni, vliaissva

Table 4.10. Hierarchical Clustering on Italian Topics.

demand market, as well as inflation—concepts that are not exclusive to the Asian economy. Similarly, `Topic51Ita` focuses on inflation and rising interest rates. To improve coherence, it would be better to separate these topics into two distinct macro-topics: one focused on the Japanese economy and yen depreciation and the other on general inflation and market dynamics.

- `Macrotopic5Ita` is related to socio-demographic trends, including population ageing, poverty, and immigration policies. `Topic11Ita`, `Topic14Ita`, `Topic30Ita` and `Topic49Ita` are appropriately merged. However, `Topic28Ita`, which focuses on mortality rates, should also be included in this macro-topic (instead of `Macrotopic11Ita`), as it relates to demographic changes.
- `Macrotopic7Ita` covers economic theory and modeling, such as Keynesian theory and econometric models. This cluster seems adequate.
- `Macrotopic8Ita` is centered on innovation and technology, including discussions on developing new professional skills, fostering innovation, and technological progress. No adjustments are necessary.
- `Macrotopic9Ita` clusters various topics related to risks, such as financial risks, cybersecurity issues, and climate-related risks. While these subjects all involve risk, they represent equally important, distinct areas of focus. It would be more effective to separate them into individual macro-topics.
- `Macrotopic10Ita`, `Macrotopic11Ita`, and `Macrotopic12Ita` are overly broad macro-topics, with many unrelated topics clustered together. For example:
  - `Topic5Ita` in `Macrotopic10Ita`, which focuses on climate change and sustainable finance, would be more appropriately merged with `Topic21Ita` from `Macrotopic9Ita`, as both relate to environmental issues.
  - `Topic7Ita`, `Topic42Ita` and `Topic44Ita`, which are spread across these three macro-topics, all address issues related to schooling systems and education. These should be grouped into a single macro-topic.
- `Macrotopic14Ita` focuses on organized crime and the Mafia, incorporating

discussions on criminal economies and mafia-related activities. This macro-topic and its representation are coherent.

- `Macrotopic15Ita` and `Macrotopic16Ita` both deal with typographical errors. It would be practical to merge them into a single macro-topic that includes `Topic15Ita`, `Topic29Ita` and `Topic46Ita`.

We can refine and manually adjust the macro-topics based on the aforementioned considerations. The updated merging is summarized in Table 4.11.

ID	Topics	Macro-topic Representation
1	48,9	italia stampa, banca italia
2	26,36	dividendo importo, dividendo distribuzione
3	13,20	spese netto, spese correnti
4	18,6	deprezzamento yen, valore yen
5	14,30,11,49,28	riduzione popolazione, povertà, mortalità
6	53,25,35	costituzione europea, unificazione, diritto romano
7	24,8,4,3,34,55,16,40	teoria economica, modelli econometrici
8	12,41,52,39	innovazione capacità, modernizzazione
9	21,5	finanza sostenibile, cambiamento climatico
10	7,22,42,44	sistema scolastico, educazione finanziaria
11	0,43,51	politica monetaria, euro
12	17,32,33	considerazioni finali
13	19,54,37	economia criminale, diffusione criminalità, mafia
14	23	rischi cibernetici, sicurezza informatica
15	10	occupazione femminile, gender equality
16	1	erogazione pensioni, schemi pensionistici
17	27	valuta digitale, euro digitale
18	2	restrizioni commercio, equilibri internazionali
19	31	banche, bilanci, finanziamenti
20	15,29,46	typographical errors
21	47,50,38,45	miscellaneous

Table 4.11. Manually adjusted Italian macro-topics.

We experimented with alternative methodologies to extract meaningful and coherent macro-topics from the original topics, including clustering techniques such as HDBSCAN and Spectral Clustering[8]. These methods were applied to the c-TF-IDF topic representations but did not outperform the results obtained through BERTopic's built-in hierarchical topic modeling. Given the relatively small corpus of documents and the limited number of topics, it is possible to manually refine the results from BERTopic's hierarchical method. However, in larger datasets with hundreds of topics, manually adjusting and analyzing each one would become

[8]For spectral clustering, the adjacency matrix of the graph $\mathcal{G}$ was derived by applying a user-defined threshold to the topic similarity matrix. If the resulting graph is disconnected, meaning it consists of multiple connected components, we directly take those components as clusters, i.e., the macro-topics. If the graph is connected, we can proceed by computing the eigen-decomposition of its Laplacian matrix. To determine the optimal number of clusters, we examine the eigenvalue-gap plot, which shows the differences between consecutive eigenvalues. The number of clusters is selected based on the first significant gap in the plot. After identifying the optimal number of clusters k, we perform k-means++ clustering on the data defined by the first k eigenvectors of the Laplacian.

impractical and time-consuming. In such cases, more automated approaches should be explored if BERTopic's hierarchical topic modeling does not yield satisfactory results.

4.2.2 Digression on Final Consideration Speeches

As mentioned in Chapter 2, the 377 Italian documents include 30 annual reports, one for each year, except for the years 1993, 1994, 1999, and 2001. These final considerations, published at the end of May (usually on the 31st), provide "a comprehensive analysis of the key developments in the Italian and international economy in the previous year and the early months of the current year"[9].

It may be of interest to study the topics contained in these annual reports and examine how they evolve across the years—and, therefore, across Governors. To achieve this, we can apply *Dynamic Topic Modeling* (DTM) using the `bertopic` library, where DTM is defined as "a collection of techniques aimed at analyzing the evolution of topics over time" Grootendorst (2024) in the BERTopic documentation. With DTM, we can observe not only how the frequency of a topic changes year by year but also how its representation varies as well.

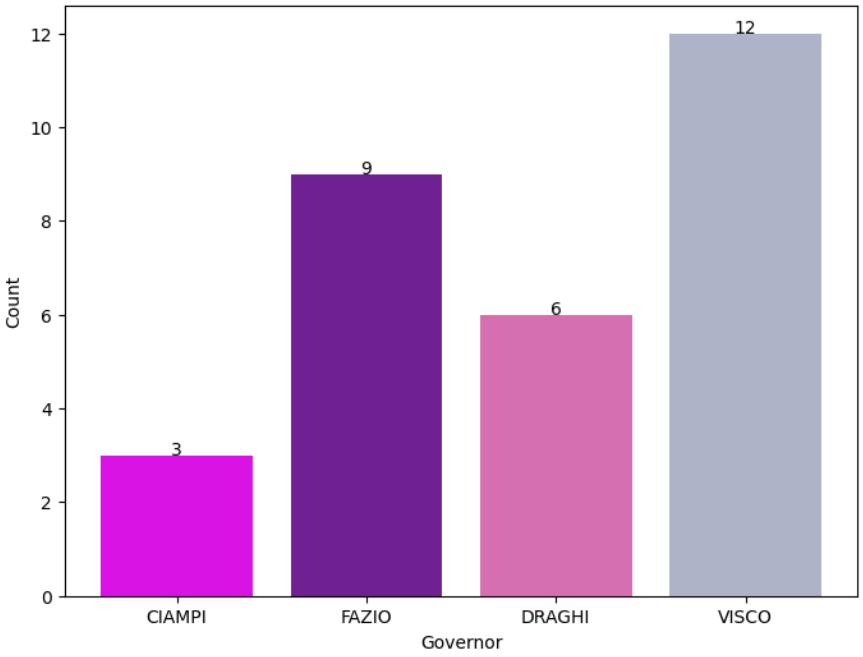


Figure 4.23. Governor distribution in the final consideration speeches.

The 30 final consideration speeches are distributed among the four Governors, as shown in Figure 4.23. Carlo Azeglio Ciampi contributed three documents covering the years 1990 to 1992. Antonio Fazio contributed nine documents spanning 1995–1998, 2000, and 2002–2005. Mario Draghi provided six documents from 2006 to 2011,

[9] https://www.bancaditalia.it/pubblicazioni/relazione-annuale/index.html?com.dotmarketing.htmlpage.language=1.

while Ignazio Visco contributed the largest share, with 12 documents from 2012 to 2023.

Annual reports tend to be significantly longer than other communications from the Governors, as they include the Governor's final remarks on the preceding year. Figure 4.24 shows the average word count of each Governor's annual reports. Ciampi's reports are the longest on average, followed by Fazio, Visco, and Draghi. Notably, Draghi's annual reports are much shorter—44% shorter than Ciampi's and 40% shorter than Fazio's.

Due to the length of these documents, which far exceeds the 128-token limit of our Sentence Transformers, it is essential to split the documents into smaller chunks to avoid truncation when generating the embeddings. By doing so, we divided the 30 annual reports into 2132 chunks, each containing 128 tokens or fewer, distributed in the following way: 268 chunks for Ciampi, 73 for Fazio, 301 for Draghi, and 824 for Visco. Since these documents are in Italian, we used the `paraphrase-multilingual-MiniLM-L12-v2` Sentence Transformer model to embed the document chunks. The rest of the `BERTopic` pipeline remained consistent with the one previously used for training the other two models (English and Italian).

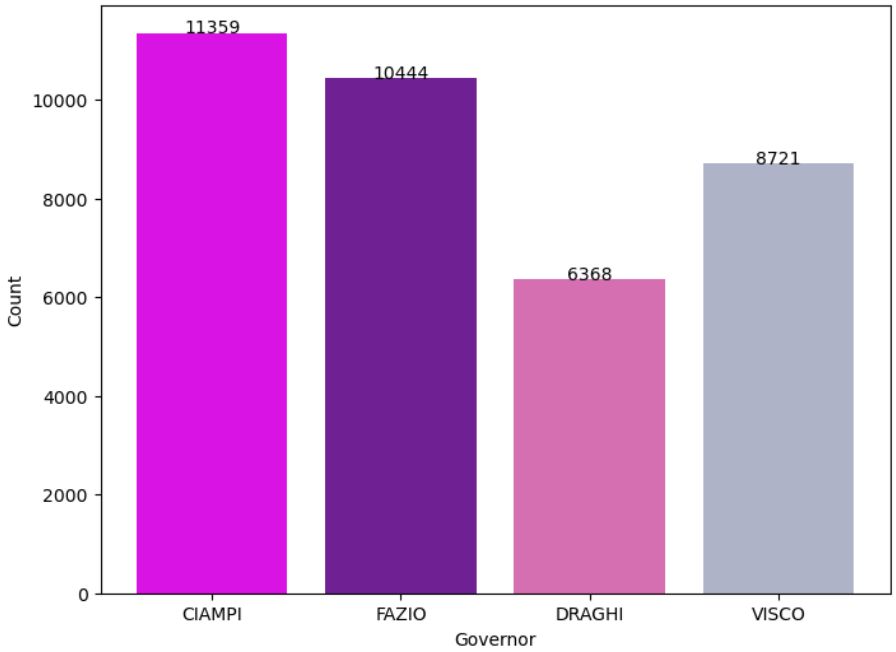


Figure 4.24. Average final consideration length per Governor.

The `Bertopic` model trained specifically on the final considerations, which we refer to as `model_cf` for simplicity, identified 30 topics (excluding noise). These topics, along with their size and representation, are listed in Table 4.12. The table shows that 897 out of 2132 document chunks were labelled as noise. The vast majority (617) of the remaining chunks (1235) form the main topic, primarily focusing on Italian banks and Banca d'Italia. However, this topic lacks specificity and is not highly informative. Interestingly, unlike the results obtained with the English and Italian models, `model_cf` did not identify a cluster related to typographical errors.

Topic	Count	Topic Representation (top 2 tokens)	Gov
-1	897	politica monetaria, banca italia	
0	617	banche italiane, sistema bancario	C,F,D,V
1	145	aumento produttività, prodotto interno	C,F,D,V
2	85	mercato lavoro, produttività lavoro	C,F,D,V
3	45	trasformazione economica, proviene economie	C,F,D,V
4	35	direttorio banca, direttore centrale	C,F,D,V
5	32	roma autorità, participantitenuta roma	C,F,D,V
6	26	pensionistico pubblico, sistemi pensionistici	C,F,D,V
7	18	sistema scolastico, sistema istruzione	F,D,V
8	16	conseguenze crisi, crisi pandemica	V
9	16	innovazioni possono, innovazione finanziaria	F,V
10	16	moneta giapponese, yen valute	C,F
11	15	economia mondiale, crescita economia	C,F,D,V
12	15	europea dobbiamo, comunità europea	C,F,V
13	15	pubblici efficienza, pubblica amministrazione	C,F,D,V
14	13	imprese borsa, modalità privatizzazione	C,F,D,V
15	13	euro italia, italia percentuale	V
16	13	rischi cibernetici, mercati criptoattività	C,D,V
17	12	finanziari rischio, aumento insolvenze	C,D,V
18	12	produttività competitività, piccole imprese	C,F,D,V
19	11	macroeconomiche deterioravano, monetaria sfociato	C,F,D,V
20	9	rischi climatici, ecosostenibili	V
21	8	fiscale rimasta, europea spesa	F,D,V
22	7	banche provenienti, economico finanziari	C,F,D
23	7	aumento produttività, produttivo cresciuto	F
24	7	europeo compimento, monetaria europeo	C,V
25	7	titoli tasso, gestione monetaria	C,V
26	6	banca italia, roma stampato	V
27	5	impiego impediscono, sistema occupazionale	C,F,D
28	5	incentivazione attività, investimento potranno	F,V
29	5	normativo governo, riforme attuate	C,D,V

Table 4.12. Annual report topics info in descending order of Count, where Count represents the topic size, C stands for Ciampi, F for Fazio, D for Draghi and V for Visco.

It could imply that the final consideration speeches contain minimal significant typographical mistakes, or that such errors were filtered out and categorized as noise by the HDBSCAN model. Another notable observation is the emergence of `Topic8`, which is focused on the COVID-19 pandemic and its economic implications, while this topic was not identified by the Italian model trained on all Italian documents. Moreover, as for the Italian model, `model_cf` also faces the problem of including verbs in its topic representations, an issue that requires further refinement, as previously discussed. Surprisingly, none of the models, whether trained on the English, Italian, or final consideration speeches, highlighted any topics related to the war in Ukraine, despite its coverage in the 2022 and 2023 annual reports.

For simplicity, we can merge several topics from Table 4.13 into the 15 macro-topics shown in Table 4.13, ordered by size. This can be done by utilizing the

ID	Topics	Macro-topic Representation
0	0,4,22,25,26	politica monetaria, inflazione, banche, finanziamenti
1	1,2,18,23,27,28	mercato lavoro, produttività, occupazione, reddito
2	3,11	economia mondiale, globalizzazione, paesi emergenti
3	5,19	considerazioni finali, verbi
4	12,15,21,24	unione europea, area euro, spesa europea, italia bce
5	6	pensioni, sistema pensionistico, fondi pensione
6	14,17	aumento insolvenze, liquidità mercato, rischio finanziario
7	7	sistema scolastico, istruzione, investimenti istruzione
8	10	moneta giapponese, yen, giappone
9	9	innovazione, innovazione finanziaria, nuove tecnologie
10	8	conseguenze crisi, crisi pandemica, crisi sanitaria
11	13	pubblica amministrazione, servizi pubblici, efficienza
12	16	rischi cibernetici, sicurezza informatica, criptoattività
13	20	rischi climatici, investimenti sostenibili, ecosostenibile
14	29	normative governo, riforme attuate

Table 4.13. Final considerations macro-topics in descending order of count.

`merge_topics` method in `model_cf`, where we manually pass a list of topics to be combined. By merging these topics, we generate new representations for the resulting macro-topics. The heatmap in Figure 4.25 displays the similarity matrix of the 15 annual report macro-topics.

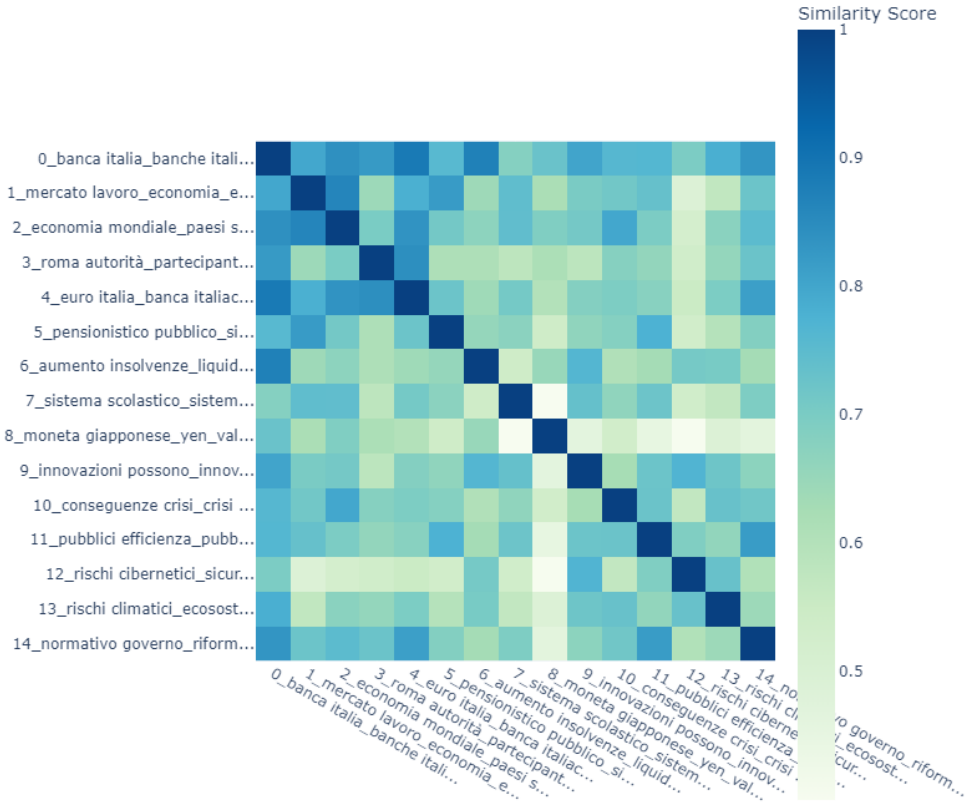


Figure 4.25. Merged final considerations heatmap.

Many of the topics in Table 4.12 align closely with those identified by the broader Italian model. Let's analyze a few examples in more detail and examine the contribution of the final considerations to the Italian macro-topics summarized in Table 4.11:

- `Macrotopic6Cf`, which focuses on Italian pension systems, is closely related to `Macrotopic16Ita`, which also covers pensions. The size of `Macrotopic6Cf` is 26, while `Macrotopic16Ita` has a size of 92. Thus, the final consideration documents contribute approximately 28% to the second-largest Italian macro-topic.
- `Macrotopic7Cf`, addressing the schooling systems and investments in education, is linked to the 10th Italian macro-topic. With a size of 18, `Macrotopic7Cf` contributes around 22% to the 83-sized `Macrotopic10Ita`.
- `Macrotopic8Cf`, focused on the Japanese economy and the yen, is closely related to `Macrotopic4Ita`, the Italian macro-topic on similar issues. With a size of 16, `Macrotopic8Cf` accounts for about 22% of the Italian macro-topic which has a size of 74.
- `Macrotopic9Cf`, centered on innovation and technology, aligns with `Macrotopic8Ita`, which also covers innovation and modernization. The size of `Macrotopic9Cf` is 16, while `Macrotopic8Ita` has size 52. Hence, the annual reports contribute approximately 31% to the Italian macro-topic.
- `Macrotopic12Cf`, which focuses on cybersecurity and cyber risks, is nearly identical to `Macrotopic14Ita`. The final considerations contribute significantly here, with `Macrotopic12Cf`'s size of 13 representing 68% of the Italian macro-topic of size 19.
- `Macrotopic13Cf`, revolving around climate change and eco-friendly investments, correlates with `Macrotopic9Ita`. The size of `Macrotopic13Cf` is 9, while the size of the broader Italian macro-topic is 75. Thus, the final consideration documents only contribute around 12% to `Macrotopic9Ita`. This suggests that sustainable finance and climate-related investments were more prominently discussed in other Governor speeches rather than in the annual reports.

Based on these observations, we can conclude that the final consideration speeches significantly contribute to the diversity and depth of topics within the Italian document corpus.

To study how the topics in Table 4.13 change over time, we can utilize `BERTopic`'s Dynamic Topic Modeling (DTM) functionality. As described in the documentation, "BERTopic enables DTM by calculating the topic representation at each time step without the need to rerun the entire model for each period" Grootendorst (2024). This can be done using the `topics_over_time` method, whose steps are illustrated in Figure 4.26.

Each topic representation at time t is fine-tuned both globally and evolutionary by setting the `global_tuning` and `evolution_tuning` parameters to "`True`". We then set the `timestamps` parameter equal to the list of years corresponding to each document chunk, covering the range from 1990 to 2023. In addition, we specify the date format by setting `date_format="%Y"` (four-digit year). Finally, we can visualize the evolution of the topics using the `visualize_topics_over_time` method. This allows us to either specify the number of most frequent topics (`top_-`

`n_topics`) or select specific topics to display. While it is possible to visualize all topics altogether, this approach is not recommended, as it would lead to a cluttered visualization. We focus on four macro-topics in this analysis: `Macrotopic8Cf`, related to the yen and the Japanese economy; `Macrotopic10Cf`, centered on the Covid-19 pandemic; `Macrotopic12Cf`, covering cybersecurity and cyber risks; and finally, `Macrotopic13Cf`, concerning climate change and sustainable finance.

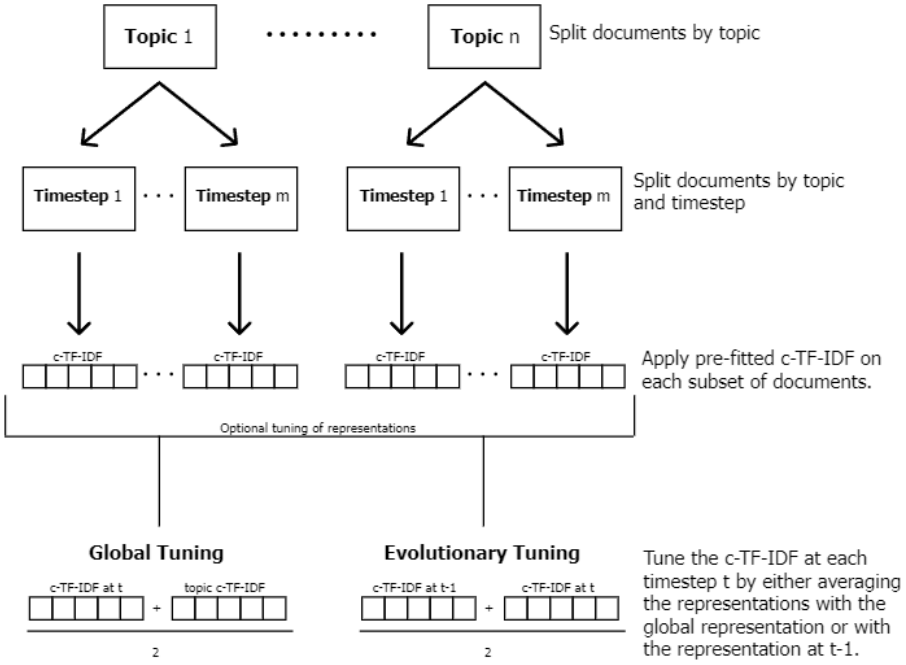


Figure 4.26. Dynamic Topic Modeling. Source: Grootendorst (2024).

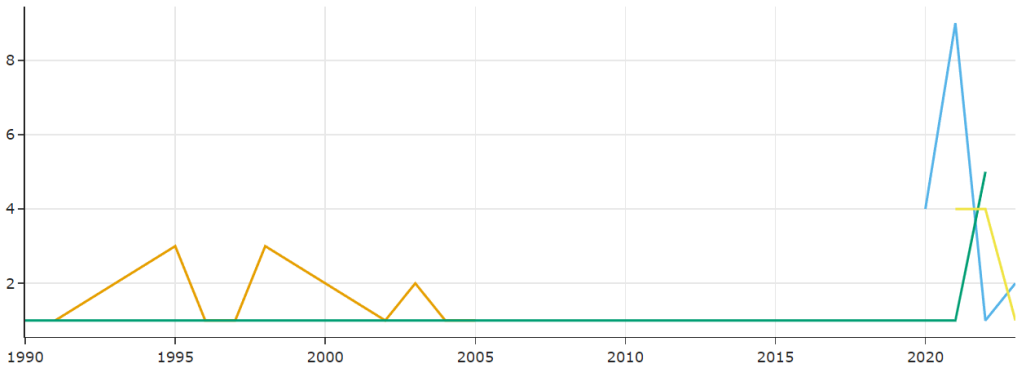


Figure 4.27. Evolution of `Macrotopic8Cf` (orange), `Macrotopic10Cf` (blue), `Macrotopic12Cf` (green) and `Macrotopic13Cf` (yellow) over time.

The evolution of these four topics is depicted in Figure 4.27, where three of the four topics are linked to more recent issues, while one is associated with the 1990s and the early 2000s.

Let's begin with `Macrotopic8Cf`, represented by the orange line in the figure.

We can observe the evolution of this topic along with its frequency in each period, as shown in Table 4.14. This topic was consistently discussed between 1991 and 2005 during the tenure of Governors Carlo Azeglio Ciampi and Antonio Fazio. Overall, the temporal representations of this topic focus on the value of the yen, especially in relation to the U.S. dollar and, later, the Euro after its introduction. More broadly, the annual reports included discussions centered around the Japanese economy, particularly the depreciation of the yen against the U.S. dollar.

Historically, this period coincides with Japan's "Lost Decade" from 1991 to 2000, which eventually extended into the "Lost 20 Years", following the burst in 1991 of the speculative bubble that had formed since 1986, affecting both the stock market and real estate sector. Japan's economy faced severe challenges during this time, including recession and economic stagnation. In 2005, Fazio highlighted the increased activity of the yen and the strengthening of the capital base of the Japanese banking system.

Year	Freq	Representation
1991	1	deprezzamento dollaro, inflazione livello, dollaro yen
1995	3	movimento valute, bancario giapponese, dollaro domina
1996	1	yen, moneta giapponese, banca giappone
1997	1	spinte inflazionistiche, inflazionistiche connesse, mercati tassi
1998	3	yen ridotto, yen, valute maggiori, oscillazioni valute
2000	2	giappone tasso, cambio yen, dollari yen, yen scendeva
2002	1	tenuta yen, diminuita profittabilità, yen
2003	2	mercati riserve, apprezzamento yen, dollari accumulo
2004	1	debito estero, cambio euro, moneta giapponese, livelli dollaro
2005	1	indebolimento dollaro, moneta giapponese, principali valute

Table 4.14. `Macrotopic8Cf` temporal frequencies and representations.

The evolution of `Macrotopic10Cf`, represented by the blue line in Figure 4.27, is further detailed in Table 4.15, which outlines the relevant years, frequencies, and temporal topic representations. As this topic is associated with the Covid-19 pandemic, it first appeared in the final consideration speech of 2020, which addresses the period from May 2019 to May 2020. In Italy, the pandemic officially began on January 30, 2020, when the first two Covid-19 cases were detected, followed by the first national lockdown on March 9, 2020. The pandemic was officially declared over on May 5, 2023, marking a period of approximately three years.

Governor Visco regularly addressed the impact of the pandemic in his speeches from 2020 onwards due to its massive impact on both the Italian and global economies. In his 2020 speech, Visco focused on how the pandemic drastically changed the way we work to prioritize public health. In 2021, as the topic representation shows, attention shifted to the vaccination campaign. "The 'Vaccine Day' on December 27, 2020, marked the official start of the COVID-19 vaccination campaign across Europe, with distribution beginning in Italy on December 31, 2020" EpiCentro ISS (2021). By 2022, the focus shifted to the economic repercussions of the pandemic, and by 2023, discussions highlighted the end of the pandemic crisis and the gradual dilution of its long-term consequences.

Let's now focus on `Macrotopic12Cf`, represented by the green line in Figure 4.27.

Year	Freq	Representation
2020	4	pandemia recessione, riforme desiderabili, sviluppo pandemia
2021	9	vulnerabili governi, campagna vaccinale, crisi pandemica
2022	1	economia crisi, stabilizzazione macroeconomica, rischi pandemia
2023	2	usciti crisi, misure contribuito, diluire conseguenze

Table 4.15. `Macrotopic10Cf` temporal frequencies and representations.

While the global representation of this topic focuses on cybersecurity and cyber risks, the temporal representations reveal a broader scope, as seen in Table 4.16. The first three document chunks were erroneously included in this cluster, as they were not actually related to this topic and had little to no influence on the overall macro-topic representation. Beyond cybersecurity, the global topic representation obscures other significant keywords and themes that emerge from the temporal representations, such as artificial intelligence, data analysis, crypto-activity, and the growing need for regulatory frameworks to address advancements in AI.

Year	Freq	Representation
1990	1	complessiva gestione, vigilanza responsabili
2011	1	provvedimenti banca, affiancano ispezioni, tutelato ragioni
2012	1	efficienza richiesta, standardizzazione informazioni
2017	1	sicurezza informatica, utilizzate vulnerabili, rischi operativi
2018	1	settore finanziario, controllata criptoattività, rischi protezione
2019	1	intelligenza artificiale, strumenti intelligenza, intelligenza
2020	1	analisi dati, estrarre informazioni, vigilanza livello
2021	1	rischi cibernetici, rischi operativi, sottovalutato pericolo
2022	5	rischiosità regolamentazione, rischi cibernetici, criptoattività

Table 4.16. `Macrotopic12Cf` temporal frequencies and representations.

Finally, focusing on `Macrotopic13Cf`, represented by the yellow line in Figure 4.27, we observe that only the annual reports from 2021 to 2023 contributed to this macro-topic. As shown in Table 4.17, the temporal representations are consistent and closely aligned, addressing highly relevant issues related to climate change and climate risks. Specifically, in 2021, sustainable finance emerged as a key focus, while in 2023, the emphasis shifted towards energy transition and CO2 emissions reduction.

Year	Freq	Representation
2021	4	climatico finance, cambiamento climatico, rischi climatici
2022	4	ambientali strategie, iniziative ambientali, rischi climatici
2023	1	rischi climatici, transizione energetica, riduzione emissioni

Table 4.17. `Macrotopic13Cf` temporal frequencies and representations.

4.3 Masked Authorship Classification

This last section focuses on classifying the Governors based on writing style rather than content. Previously, we classified documents using the entire text, but it remains unclear whether the classification models were learning patterns related to content, style, or both. Our goal here is to attempt to *disentangle content from style*—although fully separating them is likely impossible, especially in an automated fashion—to determine whether we can classify the documents based purely on style. Specifically, we seek to address two questions: Do the Governors (Ciampi, Fazio, Draghi, Visco) have significant stylistic differences? And can these differences be detected automatically without expert supervision?

We drew inspiration from a 2023 study by Wang et al., titled "Can Authorship Representation Learning Capture Stylistic Features?" (Wang et al., 2023). The authors argue that, unlike earlier studies where style was measured through specific numerical stylistic features (e.g., formality, simplicity, politeness), their goal was to *capture style in a more automated way* using deep learning models. Unfortunately, since these models act as black boxes, "we cannot directly interrogate their parameters to determine what information such representations contain" Wang et al. (2023). Moreover, Wang et al. acknowledge that "high accuracy in authorship prediction does not necessarily imply that stylistic features have been successfully learned, as it might simply be based on non-stylistic cues, like topic". Thus, it is essential to find a method to isolate stylistic elements within the documents to prevent content-related information from driving predictions.

In their study, the authors generated style-focused authorship representations by "*masking training data* in a way that preserves the syntactic structure, something which is known to relate to style, while removing thematic or topical information" Wang et al. (2023). Specifically, authorship representations are trained "with restricted access to topic signal by masking varying proportions of content-related words in the training data" Wang et al. (2023). These models were then evaluated on unmasked test data to assess their ability to capture stylistic features independently from content.

According to the authors, words can be broadly divided into two categories: *content words* and *function words*. Content words primarily convey topic signals and are nouns, adjectives, main verbs, and adverbs. Function words, on the other hand, carry stylistic signals through their usage patterns and include auxiliary verbs, prepositions, determiners, pronouns, and conjunctions. Hence, content words were masked based on their *Parts-of-Speech (POS)* tags, in a process they call *Perturbing Linguistic Expressions (PertLE)*. They employed two masking strategies:

- *PertLE Grande*, a greedy approach that masks nouns, adjectives, main verbs, and adverbs, ensuring that all content words are masked, "at the expense of occasionally masking some function words" Wang et al. (2023). For example, both "happily" and "instead" are adverbs (categorized as content words), but "happily" carries content information, while "instead" conveys stile.
- *PertLe Lite*, a more conservative approach that masks only nouns, which are the most likely carriers of content information, but may leave other content-related words unmasked.

Both strategies are heuristic and imperfect, as there is no foolproof way to fully

disentangle content from style. The words to mask were identified using Stanford NLP Group's Stanza tokenizer and POS tagger, with the selected words replaced by the token `<mask>`. The authors created three training corpora: one masked with PertLE Grande, one masked using PertLE Lite, and one left unmasked, serving as the *baseline*. All models were then evaluated on the same unmasked test data to ensure a fair comparison. This method could lead to two possible outcomes: either the models would learn stylistic patterns during training and perform well despite the masked content, or performance would decline, suggesting that the model could not compensate for the loss of content signals.

For our Governor classification task—essentially an authorship attribution problem—we adopted a similar approach to isolate style in the documents. Rather than relying on POS tags to identify content words, we selected the most significant words contributing to the topics identified through BERTopic in Section 4.2. Furthermore, instead of using a generic `<mask>` token, we employed multiple masks that specify the POS tag of each masked content word.

- Hold me closer, tiny dancer. Count the headlights on the highway. Lay me down in sheets of linen. You had a busy day today.
- Just a small-town girl, livin' in a lonely world. She took the midnight train going anywhere.
- All I wanna do is have a little fun before I die, says the man next to me out of nowhere.

(a) Unmasked

- <mask> me <mask>, <mask> <mask>. <mask> the <mask> on the <mask>. <mask> me <mask> in <mask> of <mask>. You <mask> a <mask> <mask> <mask>.
- <mask> a <mask> <mask>, <mask> in a <mask> <mask>. She <mask> the <mask> <mask> <mask> <mask>.
- All I <mask>na <mask> <mask> <mask> a <mask> <mask> before I <mask>, <mask> the <mask> <mask> to me out of <mask>.

(b) PertLE Grande

- Hold me closer, tiny <mask>. Count the <mask> on the <mask>. Lay me down in <mask> of <mask>. You had a busy <mask> <mask>.
- Just a small-town <mask>, livin' in a lonely <mask>. She took the <mask> <mask> going anywhere.
- All I wanna do is have a little <mask> before I die, says the <mask> next to me out of nowhere.

(c) PertLE Lite

Figure 4.28. Unmasked (a), PertLE Grande (b), PertLE Lite (c) example. Source: Wang et al. (2023).

4.3.1 Masking Content Words

Content words to mask were identified by iterating through the topics extracted via BERTopic for both the Italian and English models. The words were extracted using BERTopic's `get_topic` method. Since the `top_n_words` parameter was set to 30, a maximum of 30 of the most important tokens (unigrams and bigrams) per topic was extracted. In cases where a topic representation contained fewer than 30 tokens, empty strings were inserted as placeholders, which were subsequently removed to prevent accidental replacements of all strings in the documents during masking. From the list of content unigrams and bigrams for each topic, we retained sets of the top 5, 10, 15, 20, and 30 most significant words. This allowed us to mask varying proportions of content words, thus isolating writing style with different degrees of precision, similar to the PertLE Grande and PertLe Lite approaches.

For the Italian model, we excluded two topics from the 56 identified, namely `Topic15` and `Topic29`, which focused on typographical errors. Additionally, `Topic44` contained no more than 10 n-grams. For the remaining topics, the 30 most representative words were extracted. The total number of unique Italian unigrams and bigrams for different token sets was 270 words for the top 5 tokens, 540 for the top 10, 805 for the top 15, 1070 for the top 20, and 1600 for the top 30 tokens.

Similarly, for the English model, we excluded `Topic16`, the only one related to typographical errors. Furthermore, the bigram "Bank Italy" was replaced with "Bank of Italy" to match the original phrasing in the text, as "of" had been omitted in the topic representation during the c-TF-IDF step (stopwords were removed to improve clarity). The total number of unique English content words for the various token sets was 135 for the top 5 tokens, 270 for the top 10, 405 for the top 15, 540 for the top 20, and 810 for the top 30 tokens.

Once we obtained the list of topic-related words, we proceeded to mask those words in the documents. First, the content words were identified in the text and replaced with a generic `<MASK>` token. This was done using the `sub` function from the `regex` library. Next, we retrieved the POS tags for each masked content token. For bigrams, POS tags were extracted for each word. In the English case, we used the `en_core_web_sm` model from the `spacy` library, while for the Italian case, we used the `it_core_news_sm` model, both loaded via the `load` method. Fine-grained POS tags were accessed through the `tag_` attribute. Then, the generic `<MASK>` token was replaced with more specific token masks corresponding to Spacy's POS tags.

- For Italian documents, POS tags were interpreted using this source[10].
  - Adjective (`ADJ`): `A` (adjective), `NO` (ordinal number).
  - Noun (`NOUN`): `S` (noun), `SP` (proper noun).
  - Verb (`VERB`): `V` (verb), `VM` (verb, modal), `VA` (verb, auxiliary).
  - Determiner (`DET`): `RD` (definite article), `DI` (indefinite determiner), `AP` (possessive adjective).
  - Adverb (`ADV`): `B` (adverb).
  - Pronoun (`PRON`): `PI` (indefinite pronoun), `PC` (clitic pronoun).
  - Conjunction (`CONJ`): `CC` (coordinating conjunction).
  - Adposition (`ADP`): `E` (preposition).

[10] https://universaldependencies.org/docs/it/pos/all.html.

  - Number (NUM): N (number).
  - Other (X): SW (foreign word).
- For English documents, the Penn TreeBank (PTB) POS tags were interpreted using source 1[11] and source 2[12].
  - Adjective (ADJ): JJ (adjective), JJS (adjective, superlative), JJR (adjective, relative).
  - Noun (NOUN): NN (noun, singular or mass), NNS (noun, plural), NNP (proper noun, singular).
  - Verb (VERB): VB (verb, base form), VBG (verb, gerund or present participle), VBP (verb, non-3rd person singular present), VBZ (verb, 3rd person singular present), VBD (verb, past tense), VBN (verb, past participle).
  - Determiner (DET): DT (determiner).
  - Number (NUM): CD (cardinal number).
  - Particle (PART): RB (adverb).
  - Subordinating conjunction (SCONJ): IN (preposition or subordinating conjunction).
  - Other (X): FW (foreign word).

Language	Top K	Number of Unique Masks	Total Number of Masked Words	Percentage of Masked Corpus
English	5	135	8058	2.9%
	10	270	12161	4.3%
	15	405	16625	5.9%
	20	540	20843	7.4%
	30	810	26912	9.6%
Italian	5	270	15332	1.1%
	10	540	30997	2.1%
	15	805	44833	3.1%
	20	1070	63405	4.4%
	30	1600	97128	6.7%

Table 4.18. Details of the masked documents, where K represents the number of topic-related tokens masked taking values 5,10,15,20,30, the total number of words in the English document corpus is 280411, and the total number of words in the Italian document corpus is 1443595.

Table 4.18 provides an overview of the masked document corpora. For the English corpus, masking the top 5 tokens per topic resulted in approximately 3% of the text masked, while masking the top 30 tokens per topic increased this to nearly 10%. For the Italian corpus, masking the top 5 tokens per topic covered about 1% of the text, and masking the top 30 tokens per topic resulted in approximately 7% of the text masked. Compared to the PertLE Grande and PertLE Lite approaches our method applied a softer content masking. In fact, while PertLE Grande masked

[11] https://github.com/UniversalDependencies/docs/tree/pages-source/_en/pos.
[12] https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html.

up to half the text, our approach masked only up to 10% the document corpus.

To simplify the original POS tags provided by Spacy for both English and Italian texts, we grouped the original tags into 10 broader, more interpretable categories. For the English corpus, the POS tags were simplified as follows:

- Verbs (`<VERB>`): `<VBG>`, `<VB>`, `<VBD>`, `<VBN>`, `<VBP>`, and `<VBZ>`.
- Nouns (`<NOUN>`): `<NN>`, `<NNS>`, and `<NNP>`.
- Adjectives (`<ADJECTIVE>`): `<JJ>`, `<JJS>`, and `<JJR>`.
- Conjunctions and prepositions (`<CONJUN/PREP>`): `<IN>`.
- Determiners (`<DETERMINER>`): `<DT>`.
- Adverbs (`<ADVERB>`): `<RB>`.
- Numbers (`<NUMBER>`): `<CD>`.
- Foreign words (`<FOREIGN>`): `<FW>`.

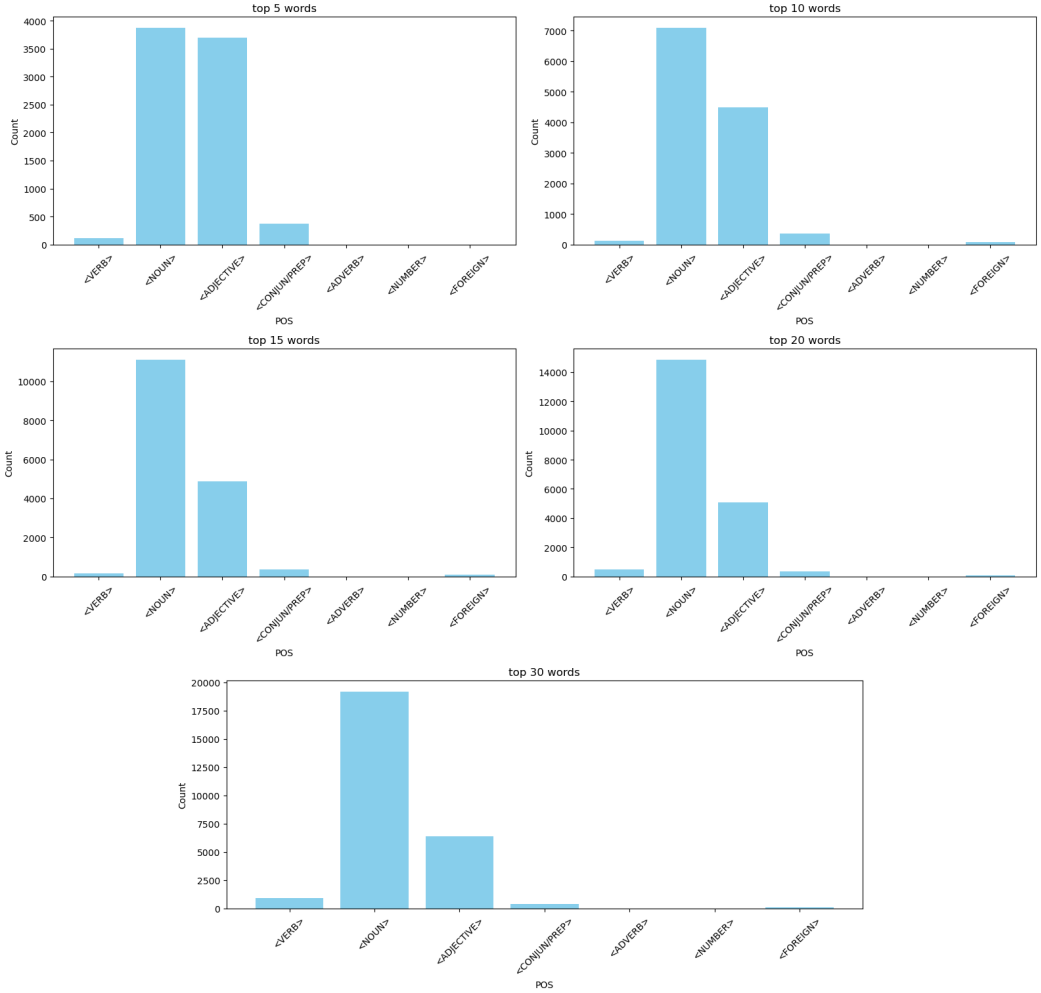


Figure 4.29. Distribution of the simplified English masks.

As shown in Figure 4.29, nouns and adjectives dominate the set of masked topic words, aligning with the definition of content words provided by Wang et al. (2023). As the number of top tokens per topic increases, the proportion of nouns relative to

adjectives grows, with nouns becoming the most frequent category. Moreover, the <CONJUN/PREP> tag is exclusively represented by the word "of" within the "Bank of Italy" trigram.

For the Italian corpus, the original POS tags were similarly grouped into the same broader categories:

- Verbs (<VERB>): <V>, <V_PC>, <VM>, and <VA>.
- Nouns (<NOUN>): <S> and <SP>.
- Adjectives (<ADJECTIVE>): <A> and <NO>.
- Determiners (<DETERMINER>): <RD>, <DI>, and <AP>.
- Pronouns (<PRONOUN>): <PI> and <PC>.
- Adverbs (<ADVERB>): <B>.
- Conjunctions and prepositions (<CONJUN/PREP>): <CC>.
- Numbers (<NUMBER>): <N>.
- Foreign words (<FOREIGN>): <SW>.
- Adpositions (<ADPOSITION>): <E> and <E_RD>.

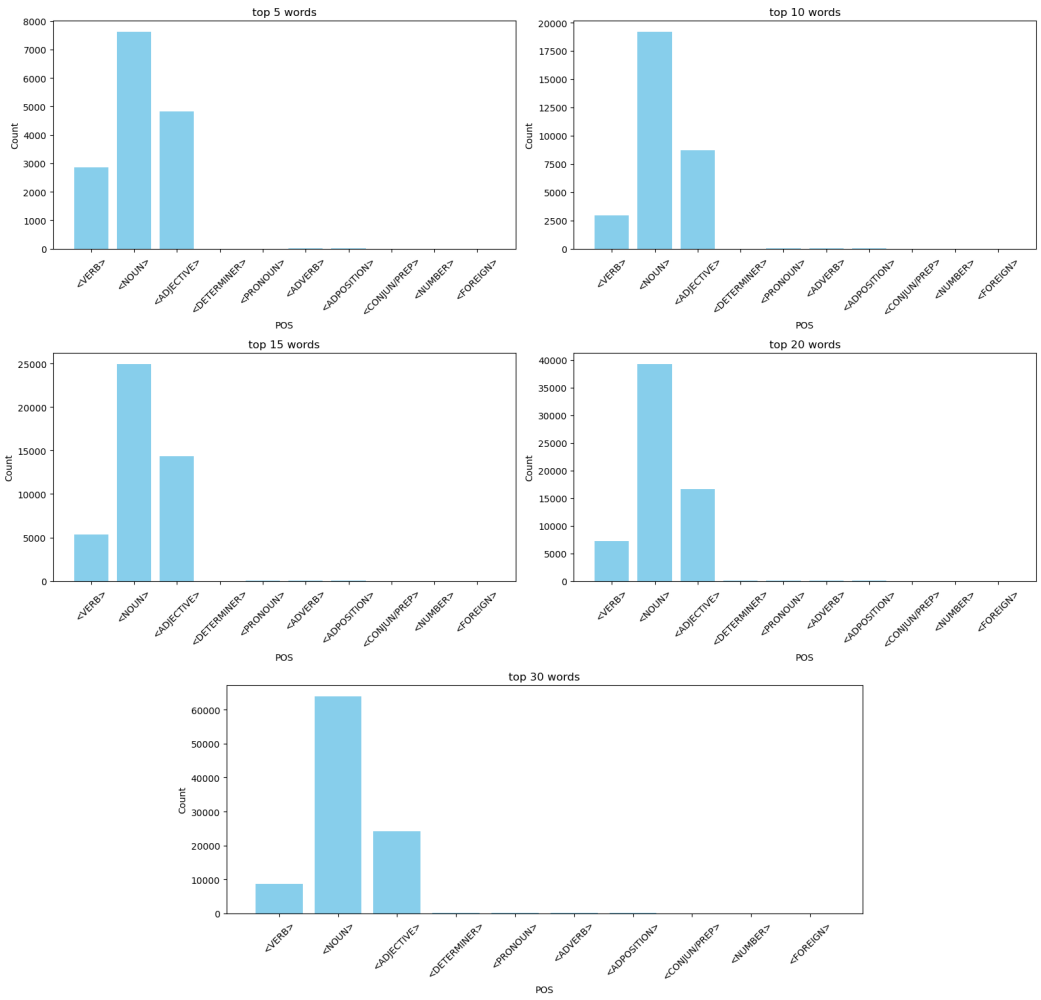


Figure 4.30. Distribution of the simplified Italian masks.

Figure 4.30 illustrates that, akin to the English case, nouns and adjectives are the most commonly masked topic words in the Italian corpus. However, in contrast to the English case, the Italian model shows a higher percentage of verbs. This observation aligns with that of Section 4.2, where we noted that verbs were more prevalent in the Italian topic representations, likely due to a less effective embedding model (multilingual) compared to the English version (monolingual).

4.3.2 TF-IDF

All masked models were trained in the same manner as the baseline models described in Section 4.1, which were trained on unmasked documents. The only difference in the masked models lies in the fact that the training set consists of masked documents, while the test set contains the original documents, allowing a fair comparison with the baseline classifiers.

In the case of TF-IDF embeddings, the `TfidfVectorizer` learns embeddings that include the 10 masking tags but exclude the content-related words replaced by the tags. This leads to a reduced vocabulary size. When creating test set embeddings, the `TfidfVectorizer` applies the vocabulary learned from the training set. Consequently, the test set embeddings contain zero values in correspondence of the masking tags (which are present only in the training set), while other words not in the training vocabulary (including topic-related words) are ignored.

Top K	Model	Accuracy	ngram_range	df_min	% max_features
baseline	SVM	**87.50%**	(2,2)	2	0.1
	LR	**87.50%**	(3,3)	3	0.1
5	SVM	50.00%	(2,3)	2	0.2
	LR	62.50%	(2,3)	2	0.1
10	SVM	62.50%	(2,2)	2	0.2
	LR	**87.50%**	(2,2)	6	0.1
15	SVM	50.00%	(2,3)	2	0.2
	LR	62.50%	(3,3)	2	0.1
20	SVM	**87.50%**	(2,2)	5	0.1
	LR	82.74%	(2,3)	7	0.1
30	SVM	50.00%	(2,3)	4	0.1
	LR	85.12%	(3,3)	5	0.1

Table 4.19. Hyperparameters and balanced test set accuracies of English TF-IDF masked classification models.

Table 4.19 displays the results of all optimal masked models for the English dataset. It is evident that, except for the top 20 masked tokens case, the logistic regression models consistently outperform the SVM models. The highest balanced test set accuracy is 87.5%, achieved by both baseline models as well as two out of the ten masked models. In this case, all of Visco's documents were correctly classified, though 1 out of 4 of Draghi's documents was misclassified. We can also notice that

many SVM models reach only 50% accuracy, equivalent to random guessing, with all instances classified as Visco. Despite this, the logistic regression model trained on the dataset with the most amount of masking still achieved an accuracy of 85.12%, suggesting that the classifier was able to detect stylistic patterns in the data.

Top K	Model	Accuracy	ngram_range	df_min	% max_features
baseline	SVM	87.16%	(2,3)	7	0.1
	LR	84.52%	(2,3)	8	0.1
5	SVM	**88.13%**	(2,3)	7	0.1
	LR	85.15%	(2,2)	5	0.1
10	SVM	86.12%	(2,2)	4	0.1
	LR	85.77%	(2,3)	6	0.1
15	SVM	86.46%	(2,3)	4	0.1
	LR	85.77%	(2,3)	5	0.1
20	SVM	86.60%	(2,3)	7	0.1
	LR	83.27%	(2,2)	6	0.1
30	SVM	85.28%	(2,3)	6	0.1
	LR	86.12%	(2,3)	6	0.1

Table 4.20. Hyperparameters and balanced test set accuracies of Italian TF-IDF masked classification models.

Table 4.20 shows the balanced test set accuracy and optimal hyperparameters for all masked classifiers trained on the Italian dataset. Overall, the models perform well, with the lowest balanced accuracy being 83.27%. The highest accuracy, 88.13%, was achieved by the SVM model trained on the dataset with the least amount of masking, outperforming the baseline model. An interesting pattern emerges regarding the optimal `ngram_range`, which was almost always set to $(2, 3)$, meaning that bigrams and trigrams were used for generating the TF-IDF embeddings. Moreover, the `TfidfVectorizer` always retained at most 10% of the total vocabulary. However, this setting had little effect as the `df_min` values were large enough to impose stricter conditions on the number of features. One can see that increasing the number of masked tokens per topic did not lead to a drop in accuracy, demonstrating that the models were still able to capture some differences between the Governors' speeches, even after excluding the most relevant topic-related words.

Table 4.21 presents the results of all masked classifiers applied to the combined dataset of English and Italian documents. The highest balanced test set accuracy, 87.92%, was achieved by the SVM model trained on the dataset masked with the top 15 tokens per topic, outperforming the baseline model. In this case, the `ngram_range` was set to $(1, 2)$, indicating that both unigrams and bigrams were used in constructing the TF-IDF embeddings, and the `df_min` parameter was set to 3. While there is a slight decrease in accuracy as the amount of masked content increases, the drop is not substantial, suggesting that the classifiers were still able to recognize some differences between Governors. However, as noted in Section 4.1 for the baseline models, it remains more effective to classify English and Italian documents separately

using distinct classifiers rather than combining them into a single model.

Top K	Model	Accuracy	ngram_range	df_min	% max_features
baseline	SVM	86.18%	(1,2)	5	0.1
	LR	86.62%	(2,3)	7	0.1
5	SVM	84.86%	(1,2)	3	0.1
	LR	80.04%	(2,3)	7	0.1
10	SVM	87.04%	(1,2)	5	0.1
	LR	82.23%	(2,3)	8	0.1
15	SVM	**87.92%**	(1,2)	3	0.1
	LR	82.68%	(2,2)	7	0.1
20	SVM	86.60%	(1,1)	7	0.3
	LR	78.93%	(2,3)	5	0.1
30	SVM	80.62%	(2,3)	8	0.1
	LR	81.74%	(2,3)	7	0.1

Table 4.21. Hyperparameters and balanced test set accuracies of multilingual TF-IDF masked classification models.

4.3.3 Doc2Vec

For the Doc2Vec embeddings, the model learns document representations that incorporate the 10 masking tags but omit the content-related words replaced by these tags. After training the Doc2Vec model on the masked training documents, embeddings for both the training and test sets were generated using the `infer_vector` method. During this inference phase, words appearing solely in the test set and masked topic-related words were ignored since the embedding model had not learned their representations during training.

Table 4.22 illustrates the balanced test set accuracy and optimal hyperparameters for all masked classifiers trained on the English dataset. The highest accuracy is 92.86%, achieved by the logistic regression model with the most extensive masking, outperforming the baseline model. This result indicates that all four of Draghi's test set documents were correctly classified, while approximately 7% of Visco's documents were misclassified. Although there is a slight decline in performance as the level of masking increases, the drop is not significant enough to suggest that classification relies purely on content-related patterns. No consistent trends were observed for the `vector_size` or `min_count` parameters. However, the `window` parameter never selected the value of 4, favoring values of 2 and 3 instead.

Table 4.23 presents the results of the masked classifiers trained on the Italian document set. The highest accuracy, 93.47%, was achieved by logistic regression models trained on the datasets masked with the top 5 and 15 tokens per topic, outperforming the baseline model. This table highlights that increasing the level of masking did not lead to a decrease in performance. In fact, models trained on the most heavily masked datasets performed better than baseline models. It suggests

Top K	Model	Accuracy	min_count	window	vector_size
baseline	SVM	82.74%	2	3	200
	LR	85.12%	3	3	150
5	SVM	82.74%	2	2	50
	LR	82.74%	1	2	200
10	SVM	82.74%	2	2	150
	LR	82.74%	2	2	150
15	SVM	82.74%	2	2	150
	LR	80.36%	2	2	50
20	SVM	80.36%	3	3	200
	LR	77.98%	3	3	200
30	SVM	80.35%	3	3	100
	LR	**92.86%**	3	2	150

Table 4.22. Hyperparameters and balanced test set accuracies of English Doc2Vec masked classification models.

that the Doc2Vec model could capture stylistic differences between the Governor's speeches, compensating for the lack of content-related information. In terms of hyperparameters, the `min_count` value of 2 was frequently chosen, meaning words appearing only once in the documents were excluded. The `vector_size` parameter often selected a value of 250, the highest among the options we tested, indicating that longer document embeddings were more effective for the Italian dataset than the English one.

Top K	Model	Accuracy	min_count	window	vector_size
baseline	SVM	91.11%	2	4	150
	LR	90.49%	2	2	250
5	SVM	89.52%	1	2	100
	LR	**93.47%**	2	2	250
10	SVM	93.06%	2	4	150
	LR	91.81%	2	4	250
15	SVM	86.94%	2	3	250
	LR	**93.47%**	3	4	250
20	SVM	91.81%	1	3	100
	LR	90.42%	2	4	250
30	SVM	91.81%	2	4	250
	LR	92.78%	2	2	200

Table 4.23. Hyperparameters and balanced test set accuracies of Italian Doc2Vec masked classification models.

Table 4.24 displays the balanced test set accuracy and optimal hyperparameters

for all masked classifiers applied to the combined English-Italian dataset. The highest accuracy, 79.11%, was achieved by the SVM model trained on the dataset masked with the top 5 tokens per topic, outperforming the baseline model. Overall, logistic regression models tend to perform better than SVM models, and performance generally declines as the amount of masking increases. For instance, the SVM model trained on the dataset masked with the top 30 tokens per topic shows a significant drop in accuracy—approximately 10% lower than the baseline. As discussed in Section 4.1.3, Doc2Vec multilingual models tend to perform worse than separate models trained individually on the Italian and English datasets. Regarding the embedding hyperparameters, the table highlights that the most common `min_count` value is 2, meaning that words appearing only once were excluded from training. Moreover, larger `vector_size` values were often selected, suggesting that longer document vectors are required to capture the multilingual content effectively.

Top K	Model	Accuracy	min_count	window	vector_size
baseline	SVM	78.12%	1	3	250
	LR	78.71%	2	3	250
5	SVM	74.76%	2	3	250
	LR	76.06%	1	3	200
10	SVM	**79.11%**	2	3	250
	LR	74.27%	2	3	200
15	SVM	73.40%	2	4	250
	LR	77.32%	2	4	200
20	SVM	70.27%	3	2	200
	LR	71.87%	2	3	200
30	SVM	68.08%	3	2	250
	LR	74.29%	2	3	100

Table 4.24. Hyperparameters and balanced test set accuracies of multilingual Doc2Vec masked classification models.

4.3.4 Transformer-based Fine-tuned Models

For the Transformer-based classification models, the training process followed a similar approach to the baseline models described in Section 4.1.4, with one key difference. After splitting the dataset into training, validation, and test sets and further dividing these into smaller chunks to respect the maximum token limit, we proceeded with tokenizing the documents. However, unlike the baseline models, we augmented the tokenizer by adding the set of 10 masking tokens to the vocabulary. This was done using the `add_tokens` method from the `transformers.PreTrainedTokenizer` module[13]. These newly added tokens represented the masking tags used to replace topic-related words in the documents.

[13] https://discuss.huggingface.co/t/adding-new-tokens-while-preserving-tokenization-of-adjacent-tokens/12604/3.

To ensure that the model could accommodate these new tokens, we expanded its embedding matrix by applying the `resize_token_embeddings` method from the `transformers.PreTrainedModel` module. Without this step, tokenization would rely on the model's pre-trained knowledge, splitting the masking tokens like "`<NOUN>`" into three separate tokens: "`<`", "`NOUN`", and "`>`". By explicitly adding these tokens to the vocabulary, the model learns to treat them as unique elements during fine-tuning, developing an understanding of their meaning.

Then, the models were trained and evaluated in the same manner as the baseline models, including a grid search of the same hyperparameters (dropout rate, learning rate, and number of epochs) to ensure fair comparison. With these fine-tuned models, we can generate predictions for new documents after appropriately tokenizing them with the updated tokenizer. In addition, they can be used as a pre-trained foundation for further fine-tuning, such as incorporating a new Governor like Fabio Panetta.

Table 4.25 shows the balanced test set accuracy, both at the chunk and document levels, along with the optimal hyperparameters for all masked models trained on the English dataset. The highest accuracy, 87.5%, was achieved by the baseline model and three out of five masked models (those masking the top 5, 15, and 20 tokens per topic). By looking at the table, we can see that the performance of the models did not worsen as the level of masking increased. These masked models, however, required more patience in early stopping compared to the baseline model. The patience parameter had to be set to 8, as the loss took more than 5 epochs to decrease during training, indicating a plateau phase that had to be surpassed before improvements could be observed. If we had maintained a patience of 5, training would have stopped prematurely, resulting in poor performance. Overall, the results confirm that good classification performance can be achieved even when masking the most significant content-related words. It suggests that the model can detect deeper relationships between words, likely linked to stylistic differences between the Governors rather than based purely on content.

Top `K`	**Chunk accuracy**	**Doc accuracy**	`lr`	`dr`	`n_epochs`
baseline	82.86%	**87.50%**	7e-6	30%	23
5	81.17%	**87.50%**	5e-6	30%	31
10	84.47%	85.12%	5e-6	50%	37
15	86.48%	**87.50%**	2e-5	50%	22
20	**86.52%**	**87.50%**	2e-5	30%	22
30	82.95%	85.12%	1e-5	70%	40

Table 4.25. Hyperparameters (learning rate, dropout rate, and number of epochs) and balanced test set accuracies of English Transformer-based masked classification models.

Table 4.26 presents the results of all masked models trained on the Italian dataset. The highest document-level balanced test set accuracy, 94.66%, was achieved by the classifier trained on the dataset masked with the top 5 tokens per topic, surpassing the baseline model by approximately 4%. Interestingly, increasing the amount of masking did not lead to a significant drop in performance. For instance, the model

trained on the dataset masked with the top 20 words per topic still achieved a higher accuracy than the baseline. It suggests that the model could detect meaningful patterns in the data, even in the absence of significant content-related words. The results reveal that, unlike the English models, the Italian models favored higher dropout rates. This indicates that stronger regularization was needed for the model to generalize to new data. Moreover, larger learning rates were more frequently selected in the AdamW optimization process within the allocated 40 training epochs. Overall, the results confirm that the fine-tuned models were capable of identifying patterns in data that go beyond mere content and may be related to stylistic writing differences between Governors.

Top K	Chunk accuracy	Doc accuracy	lr	dr	n_epochs
baseline	75.49%	90.98%	1e-5	80%	39
5	73.45%	**94.66%**	1e-5	70%	27
10	75.63%	92.65%	1e-5	80%	30
15	**76.38%**	88.34%	5e-6	80%	40
20	74.05%	92.23%	2e-5	80%	40
30	73.25%	88.00%	1e-5	70%	40

Table 4.26. Hyperparameters (learning rate, dropout rate, and number of epochs) and balanced test set accuracies of Italian Transformer-based masked classification models.

Top K	Chunk accuracy	Doc accuracy	lr	dr	n_epochs
baseline	60.96%	74.09%	5e-6	70%	28
5	**67.26%**	79.35%	1e-5	80%	40
10	61.73%	**82.41%**	7e-6	70%	26
15	64.60%	80.65%	5e-6	80%	31
20	65.01%	77.15%	1e-5	80%	31
30	60.98%	77.15%	1e-5	50%	21

Table 4.27. Hyperparameters (learning rate, dropout rate, and number of epochs) and balanced test set accuracies of multilingual Transformer-based masked classification models.

Table 4.27 displays the balanced test set accuracy, both at the chunk and document levels, along with the optimal hyperparameters for all masked models trained on the combined English-Italian dataset. The highest accuracy, 82.41%, was achieved by the model trained on the dataset masked with the 10 most significant words per topic, exceeding the baseline model by approximately 8%. We can notice from the table that all masked models outperformed the baseline multilingual model, concluding that it is possible to classify Governors' speeches even when a significant portion of content-related words is masked. However, as discussed for the baseline

model, it remains more effective to classify the English and Italian documents separately rather than using a combined multilingual approach.

4.3.5 Model Comparison

Table 4.28 highlights the top-performing models for the English document classification task. The highest balanced test set accuracy of 92.86% was achieved by the Doc2Vec model trained on the dataset masked with the top 30 most relevant words per topic. In contrast, for all other masking levels, including the baseline, the best accuracy reached was 87.5%. In these cases, 1 out of 4 of Draghi's documents was misclassified, while all of Visco's documents were correctly classified. Apart from one standout result, Doc2Vec models underperformed compared to TF-IDF and Transformer-based models. The 92.86% accuracy was only achieved because all four of Draghi's documents were classified correctly. Given that there are fewer Draghi's test set documents than Visco's, misclassifying a Draghi instance has a heavier effect on the balanced test set accuracy. Thus, to more effectively assess the overall performance of these models, it would be best to use a larger test set with more instances of Governor Mario Draghi.

Top K	Best models	Accuracy
baseline	Transformer, TF-IDF + SVM, TF-IDF + LR	87.50%
5	Transformer	87.50%
10	TF-IDF + LR	87.50%
15	Transformer	87.50%
20	Transformer, TF-IDF + SVM	87.50%
30	Doc2Vec + LR	**92.86%**

Table 4.28. Balanced test set accuracies of the best English masked classification models.

Table 4.29 displays the top classification models for the Italian dataset at various masking levels. The highest balanced test set accuracy, 94.66%, was achieved by the fine-tuned Transformer-based model trained on the dataset with the least masking, surpassing the baseline model by approximately 3.5%. Overall, all the masked classification models listed in the table performed better than the baseline. Unlike the results for the English dataset, the TF-IDF models were outclassed by the Doc2Vec and Transformer-based models. These results are not surprising given the nature of Doc2Vec and Transformer-based models, which are better suited to capture deeper linguistic relationships than frequency-based methods like TF-IDF. Doc2Vec, for example, creates document embeddings that not only represent the semantics of individual words but also consider their position and context within the text, reflecting the stylistic elements of the documents to some extent. The strong results obtained even when most content-related words are masked suggest that Doc2Vec models effectively rely on the structural patterns and style of the texts. Similarly, Transformer-based models are expected to capture deeper relationships between words, leveraging their pre-trained knowledge and further knowledge gained through fine-tuning the model on the masked datasets. However, due to their increased

complexity, these models are more prone to overfitting, which can limit their ability to generalize effectively.

Top K	Best models	Accuracy
baseline	Doc2Vec + SVM	91.11%
5	Transformer	**94.66%**
10	Doc2Vec + SVM	93.06%
15	Doc2Vec + LR	93.47%
20	Transformer	92.23%
30	Doc2Vec + LR	92.78%

Table 4.29. Balanced test set accuracies of the best Italian masked classification models.

Table 4.30 shows the best-performing classification models for the combined English-Italian dataset across different masking levels. The highest balanced test set accuracy, 87.92%, was achieved by an SVM model with TF-IDF embeddings, trained on the dataset masked with the top 15 words per topic. As for the English-only and Italian-only cases, the overall best model involved some form of masking. This enforces the assumption that these models can capture differences between Governors' documents even when the most significant topic-related words are replaced by masking tokens. One notable observation is that all models listed in the table utilized TF-IDF embeddings, while Doc2Vec and Transformer-based models were completely outperformed. This result likely arose from challenges in multilingual representation. For instance, in the case of Doc2Vec, semantically similar words in different languages might be mapped to distant areas in the embedding space, leading to less accurate document representations and, consequently, poorer classification performance compared to the monolingual scenario. As our document corpus involves only two languages, we can conclude that it is more effective to separate the documents by language and use monolingual models instead of using a combined multilingual classification approach. On the other hand, when dealing with documents involving many more languages, the multilingual model approach would be the most convenient option.

Top K	Best models	Accuracy
baseline	TF-IDF + LR	86.62%
5	TF-IDF + SVM	84.86%
10	TF-IDF + SVM	87.04%
15	TF-IDF + SVM	**87.92%**
20	TF-IDF + SVM	86.60%
30	TF-IDF + LR	81.74%

Table 4.30. Balanced test set accuracies of the best multilingual masked classification models.

4.3.6 Can We Capture Stylistic Features?

Based on the findings discussed in Section 4.3.5, where we compared baseline models with classifiers trained on varying levels of masking, we can confidently state that in all scenarios—English-only, Italian-only, and the combined English-Italian datasets—at least one masked model performed better than the baseline. We achieved superior results to the original unmasked models by replacing some of the most significant topic-related terms and training the models on these modified documents.

From these considerations, we can make a few assumptions. First, by obscuring the content-heavy words, we probably forced the models to focus on other distinguishing features among the Governors, possibly related to stylistic patterns. Rather than using a generic `<MASK>` token like in Wang et al. (2023), we applied a variation of this approach that involved using 10 distinct tokens representing the part of speech (POS) of the masked words, providing valuable insights into the Governors' writing styles. Second, the strong classification performance of the masked models implies that there are detectable stylistic differences between the Governors, which the models could utilize for classification.

However, these statements are only assumptions because:

- We cannot conclusively identify the specific authorship features that contributed to the classification process or definitively prove that the models based their decisions on stylistic differences.
- It remains difficult, if not impossible, to fully disentangle style from content, especially in an automated manner.
- A substantial portion of the documents remained unmasked, potentially allowing content-based patterns to influence the results. It would be interesting to further investigate this by increasing the proportion of masked words to mimic Wang et al. (2023)'s PertLE Grande approach, which masked approximately half the words in the text.

Nonetheless, we can reasonably infer that models with some degree of masking likely achieved better classification accuracy than the baseline because shared topics between Governors could have confused the unmasked models, resulting in misclassifications. As seen in Section 4.1, the confusion matrices for these unmasked models revealed that many misclassifications often reflected the temporal proximity of the Governors. For instance, both Ciampi and Fazio discussed the monetary implications of Japan's economic crisis in the 1990s and early 2000s. This content overlap could have led to confusion discerning between the two. Instead, by masking topic-related words, the models may have been able to focus on more distinguishing features, such as style, consequently reducing the impact of these content-driven errors and, in turn, improving classification accuracy.

Chapter 5

Conclusions and Future Work

In this thesis, we analyzed a collection of speeches given by the Governors of the Bank of Italy from 1990 to 2023, including those from Ciampi, Fazio, Draghi, and Visco.

The first objective of this study, discussed in Section 4.1, was to classify the Governors based on the full text of their speeches. We tested various combinations of document embedding techniques and classification methods. Specifically, we employed TF-IDF and Doc2Vec to generate document embeddings, which were then used as inputs for SVM and logistic regression classifiers, as well as fine-tuned Transformer-based models. Our experiments were conducted across three distinct scenarios: one using only English documents, one using only Italian documents, and one combining both English and Italian documents. We discovered that classifying English and Italian documents separately resulted in better performance than classifying the entire corpus as a whole.

For the English dataset, the highest balanced test set accuracy of 87.50% was achieved by three models: an SVM model with TF-IDF embeddings, an SVM model with Doc2Vec embeddings, and the fine-tuned RoBERTa-base model. All three models correctly classified all 21 of Visco's documents and misclassified 1 out of 4 of Draghi's documents.

For the Italian dataset, the highest balanced test set accuracy, 91.11%, was achieved by an SVM model with Doc2Vec embeddings. In this case, only 1 out of 6 of Ciampi's documents was misclassified, all 19 of Fazio's documents were correctly classified, 2 out of 15 of Draghi's documents were misclassified, and 34 out of 36 of Visco's documents were classified correctly.

For the combined English-Italian dataset, the highest accuracy of 86.62% was obtained with a logistic regression model using TF-IDF embeddings. When classifying English and Italian documents separately, the overall balanced test set accuracy improved to 91.01% computed as $\frac{1}{4}\left(\frac{5}{6}+\frac{19}{19}+\frac{16}{19}+\frac{55}{57}\right)$, the average recall across all classes. In this case, fine-tuned Transformer-based models were entirely outperformed by other methods.

The second objective, addressed in Section 4.2, was to identify and analyze the underlying topics in the speeches using a state-of-the-art topic modeling technique. To achieve the best results, we ran the BERTopic algorithm separately on the English and Italian documents.

For the English documents, BERTopic identified 28 topics, including monetary policy and inflation, sustainable finance and ESG ratings, instant payments and the digital Euro, cybersecurity, the Covid-19 pandemic, and other socio-demographic trends.

For the Italian documents, BERTopic uncovered 56 topics, which is double the number found in the English documents. While many topics overlapped with those in the English set, such as sustainable finance and climate change, technology and innovation, and digital currency, several new topics emerged. These included discussions on the value of the yen and the Japanese economy, organized crime, the Mafia and corruption, Italy's educational system and financial literacy, and women's roles in the economy, with a particular focus on female employment rates.

In Subsection 4.2.2, we further concentrated on the themes present in the Governors' final consideration speeches and examined their evolution over time. We trained a BERTopic model on the 30 annual reports (one per year, excluding the years 1993, 1994, 1999, and 2001), all delivered in Italian. In this case, BERTopic identified 30 topics, including monetary policy, pensions, education, the Japanese economy, the Covid-19 pandemic, cybersecurity, and sustainable investments.

By leveraging the Dynamic Topic Modeling (DTM) functionality in BERTopic, we could track not only how frequently these topics changed year by year but also how their representations evolved. We focused on four key topics in this analysis: `Macrotopic8Cf`, related to the yen and the Japanese economy; `Macrotopic10Cf`, centered on the Covid-19 pandemic; `Macrotopic12Cf`, covering cybersecurity and cyber risks; and finally, `Macrotopic13Cf`, concerning climate change and sustainable finance. For instance, we observed that `Macrotopic8Cf` was consistently discussed between 1991 and 2005, corresponding to the tenure of Governors Ciampi and Fazio. This period coincided with Japan's "Lost Decade" (1991-2000), which extended into the "Lost 20 Years", following the speculative bubble burst in 1991.

The third and final objective of this thesis, discussed in Section 4.3, was to classify the Governors based on their distinctive stylistic traits. To isolate the stylistic elements of the speeches, we masked the topic-related words identified through the BERTopic models for both the English and Italian documents from Section 4.2. For each topic, we selected the top 5, 10, 15, 20, and 30 most significant content unigrams and bigrams, allowing us to mask varying portions of the speeches. In the English documents, this masking ranged from 3% to 10%, while in the Italian documents, it covered 1% to 7% of the text. Rather than using a generic masking token such as `<MASK>`, we employed 10 distinct masking tokens corresponding to the Part-Of-Speech (POS) tags of the word(s) being masked, preserving some sense of sentence structure. The POS categories included verbs, nouns, adjectives, adverbs, determiners, pronouns, conjunctions, numbers, foreign words, and adpositions.

We then trained the masked models in the same way as the baseline models, i.e. those trained on the original documents. However, while the masked models were evaluated on the same unmasked test set, allowing a fair comparison with the baseline models, they were trained on masked documents.

For the English dataset, the highest balanced test set accuracy of 92.86% was achieved by a logistic regression model with Doc2Vec embeddings trained on the dataset masked with the top 30 most relevant words per topic. In this case, all 4 of Draghi's documents were correctly classified, and 3 out of 21 of Visco's documents

were misclassified. Correctly classifying all of Draghi's documents allowed the balanced test set accuracy to rise above 87.50% despite introducing additional misclassifications in Visco's class.

For the Italian dataset, the highest balanced test set accuracy, 94.66%, was obtained with the Italian version of the RoBERTa-base model fine-tuned on the dataset masked with the least amount of masking. In this case, all 6 of Ciampi's documents were correctly classified, 3 out of 19 of Fazio's documents were misclassified, all 15 of Draghi's documents were classified correctly, and 2 out of 36 of Visco's documents were misclassified. This masked model surpassed the baseline model by approximately 3.5% in terms of balanced accuracy.

For the combined English-Italian dataset, the highest accuracy of 87.92% was achieved by an SVM model with TF-IDF embeddings trained on the dataset masked with the top 15 most significant words per topic. However, when classifying English and Italian documents separately, the overall balanced test set accuracy improved to 93.86% computed as $\frac{1}{4}\left(\frac{6}{6}+\frac{16}{19}+\frac{19}{19}+\frac{52}{57}\right)$.

In summary, the best-performing models for the English-only, Italian-only, and English-Italian datasets are: a logistic regression model with Doc2Vec embeddings trained on the dataset masked with the top 30 tokens per topic (92.86%), the fine-tuned Transformer-based model with the least masking (94.66%), and an SVM model with TF-IDF embeddings trained on the dataset masked with the 15 most significant tokens per topic (87.92%). Overall, by combining the predictions obtained from the English and Italian models separately, we achieved a balanced accuracy of 93.86% on the test set, outperforming the model trained on the entire document corpus by approximately 6%.

By masking certain parts of the training data, we saw a significant improvement in performance. Specifically, masking led to a 5.4% increase for the English dataset and a 3.6% increase for the Italian dataset, resulting in an overall increase of 3% across both document collections (93.86% compared to 91.01%). It seems likely that the masked models outperformed the baseline models because masking content-related words helped reduce confusion caused by overlapping topics between Governors. By masking topic-related words, the models were encouraged to focus more on stylistic features, reducing the impact of content-driven errors and thus enhancing classification accuracy.

Based on these results, we could further probe improvements in Authorship Attribution (AA) by masking a greater portion of the training data, akin to the PertLE Grande approach in Wang et al. (2023), where up to 50% of the text was masked. This approach could help determine whether increasing the amount of masked content leads to further gains in classification performance, showing that style plays a more significant role, or whether it causes a substantial drop in accuracy, suggesting that more subtle masking is optimal. It would also be interesting to experiment with alternative content-masking techniques. For instance, we could replicate the masking strategy proposed by Wang et al. (2023), which involves masking words based on their POS tags. In their PertLE Grande model, all nouns, adjectives, main verbs, and adverbs are masked, while the PertLE Lite model focuses solely on masking nouns.

Furthermore, we could expand the model to include other Governors, such as

those preceding Carlo Azeglio Ciampi or the current Governor of the Bank of Italy, Fabio Panetta, who succeeded Ignazio Visco in 2023.

This study has various applications for further research. One potential application involves utilizing our trained classification models for detecting deepfakes. For instance, suppose that we ask an advanced AI-based chatbot, like ChatGPT, to generate a fake speech on a particular topic but in the style of a specific Governor. Our objective would be to correctly label the text as AI-generated, thus eliminating the possibility of it being an authentic speech given by one of the Governors used to train the models. For this task, models trained on masked documents look more promising than the baseline models trained on the original speeches.

One naive approach to the problem would be to directly exploit the pre-trained classification models to output predictions for these AI-generated speeches. Given the robustness of our models, if a document is misclassified, it could suggest that it is a deepfake speech rather than a legitimate one. Moreover, we could investigate the predicted probabilities to determine whether the model makes confident predictions or is uncertain about classifying the deepfakes.

Instead, a more advanced approach could involve using *classification-based outlier detection* techniques. These techniques operate in two phases: a training phase, where a classifier is trained using the available labeled training data, and a testing phase that classifies a test instance as normal (authentic speech) or an outlier (deepfake speech) using the learned model (Upadhyaya and Singh, 2012).

In addition, a third approach could be to adapt our fine-tuned Transformer-based models to the new task by modifying the classification head to perform binary classification with labels "Real" (for the authentic speeches) and "Fake" (for the deepfake speeches) and re-training the models on the new training data.

A second potential application of this thesis involves utilizing the topics identified though topic modeling to create an economic index that measures the influence of central bank communication on the Italian and global economy, akin to the indices developed in previous studies (Priola et al., 2021; Baumgärtner and Zahner, 2023; Astuti et al., 2022). However, unlike those studies where Latent Dirichlet Allocation (LDA) topic modeling was applied to Bag-of-Words (BoW) document embeddings, we used BERTopic, a topic modeling algorithm that employs Sentence Transformers to generate semantically and contextually meaningful embeddings. This more sophisticated approach to topic modeling could represent a good starting point for developing more accurate economic indices based on central bank communication.

Bibliography

Valerio Astuti, Riccardo De Bonis, Sergio Marroni, Alessandro Vinci, et al. Così parlarono i governatori della banca d'italia: un'analisi del corpus linguistico delle" considerazioni finali". *QUESTIONI DI ECONOMIA E FINANZA*, 2020.

Valerio Astuti, Alessio Ciarlone, and Alberto Coco. The role of central bank communication in inflation-targeting eastern european emerging economies. *Bank of Italy Temi di Discussione (Working Paper) No*, 1381, 2022.

Martin Baumgärtner and Johannes Zahner. Whatever it takes to understand a central banker: Embedding their words using neural networks. Technical report, IMFS Working Paper Series, 2023.

David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

Jean-François Bouscasse, Daniel Kapp, Kedan Danielle, Thomas McGregor, and Julian Schumacher. How words guide markets: measuring monetary policy communication. https://www.ecb.europa.eu/press/blog/date/2023/html/ecb.blog230809~f101598a82.en.html, 2024. Accessed: 2024-09-27.

Vicki Boykis. What are embeddings. https://vickiboykis.com/what_are_embeddings/, 2023. Accessed: 2024-09-23.

Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 160–172. Springer, 2013.

François Chollet et al. Keras. https://keras.io, 2015.

European Commission. Sustainable finance - environmental, social and governance ratings and sustainability risks in credit ratings. https://ec.europa.eu/info/law/better-regulation/have-your-say/initiatives/13330-Sustainable-finance-environmental-social-and-governance-ratings-and-sustainability-risks-in-credit-ratings_en, 2022. Accessed: 2024-09-23.

Alexis Conneau and Guillaume Lample. Cross-lingual language model pretraining. *Advances in neural information processing systems*, 32, 2019.

Corinna Cortes. Support-vector networks. *Machine Learning*, 1995.

Jacob Devlin. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

EpiCentro ISS. National covid-19 vaccination plan. https://www.epicentro.iss.it/en/vaccines/covid-19-vaccination-plan, 2021. Accessed: 2024-09-23.

Evidently AI Team. How to interpret a confusion matrix for a machine learning model. https://www.evidentlyai.com/classification-metrics/confusion-matrix, n.d. Accessed: 2024-09-27.

Maël Fabien, Esaú Villatoro-Tello, Petr Motlicek, and Shantipriya Parida. Bertaa: Bert fine-tuning for authorship attribution. In *Proceedings of the 17th International Conference on Natural Language Processing (ICON)*, pages 127–137, 2020.

Kim Franke-Folstad. What does it mean if the fed is hawkish or dovish? https://www.sofi.com/learn/content/hawkish-vs-dovish-monetary-policy/#:~:text=Hawkish%20monetary%20policy%20focuses%20on,and%20may%20involve%20lowering%20rates, 2024. Accessed: 2024-09-27.

Stefan Gebauer, Thomas McGregor, and Julian Schumacher. How central bank communication affects the economy. https://www.ecb.europa.eu/press/blog/date/2024/html/ecb.blog240731~61a58b1e4a.en.html, 2024. Accessed: 2024-09-27.

Maarten Grootendorst. Topics over time. https://maartengr.github.io/BERTopic/getting_started/topicsovertime/topicsovertime.html, 2024. Accessed: 2024-09-23.

Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

Sarwat Jahan, Ahmed Saber Mahmud, and Chris Papageorgiou. What is keynesian economics. *International Monetary Fund*, 51(3):53–54, 2014.

Hamed Jelodar, Yongli Wang, Chi Yuan, Xia Feng, Xiahui Jiang, Yanchao Li, and Liang Zhao. Latent dirichlet allocation (lda) and topic modeling: models, applications, a survey. *Multimedia tools and applications*, 78:15169–15211, 2019.

Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196. PMLR, 2014.

L'economia per tutti - Banca d'Italia. Sustainable finance. https://economiapertutti.bancaditalia.it/informazioni-di-base/finanza-sostenibile/index.html?com.dotmarketing.htmlpage.language=3&dotcache=refresh&dotcache=refresh, 2022. Accessed: 2024-09-23.

Leland McInnes, John Healy, and Steve Astels. How hdbscan works. https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html, 2016. Accessed: 2024-09-23.

Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

Tomas Mikolov. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

Alex Minnaar. Word2vec tutorial part ii: The continuous bag-of-words model. https://alexminnaar.com/2015/05/18/word2vec-tutorial-continuousbow.html, 2015. Accessed: 2024-09-23.

Eunjeong L Park, Sungzoon Cho, and Pilsung Kang. Supervised paragraph vector: distributed representations of words, documents and class labels. *IEEE Access*, 7: 29051–29064, 2019.

Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 2(11):559–572, 1901.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Pinecone. Tomayto, tomahto, transformer: Multilingual sentence transformers. https://www.pinecone.io/learn/series/nlp/multilingual-transformers/, n.d.a. Accessed: 2024-09-24.

Pinecone. Sentence transformers: Meanings in disguise. https://www.pinecone.io/learn/series/nlp/sentence-embeddings/, n.d.b. Accessed: 2024-09-23.

Maria Paola Priola, Annalisa Molino, Giacomo Tizzanini, et al. The informative value of central banks talks: a topic model application to sentiment analysis. 2021.

Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. http://is.muni.cz/publication/884893/en.

N Reimers. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

Nils Reimers and Iryna Gurevych. Making monolingual sentence embeddings multilingual using knowledge distillation. *arXiv preprint arXiv:2004.09813*, 2020.

Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21, 1972.

Michael Strobl, Jörg Sander, Ricardo JGB Campello, and Osmar Zaïane. Model-based clustering with hdbscan. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2020, Ghent, Belgium, September 14–18, 2020, Proceedings, Part II*, pages 364–379. Springer, 2021.

Shuchita Upadhyaya and Karanjit Singh. Classification based outlier detection techniques. *International Journal of Computer Trends and Technology*, 3(2): 294–298, 2012.

A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

Andrew Wang, Cristina Aggazzotti, Rebecca Kotula, Rafael Rivera Soto, Marcus Bishop, and Nicholas Andrews. Can authorship representation learning capture

stylistic features? *Transactions of the Association for Computational Linguistics*, 11:1416–1431, 2023. URL https://direct.mit.edu/tacl/article/doi/10.1162/tacl_a_00610/118299.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in Neural Information Processing Systems*, 33: 5776–5788, 2020.

Grace Zhang. What is the kernel trick? why is it important? https://medium.com/@zxr.nju/what-is-the-kernel-trick-why-is-it-important-98a98db0961d, 2018. Accessed: 2024-09-25.