POLITECNICO DI TORINO

Master Degree course in Ingegneria Matematica track Statistica e Ottimizzazione su Dati e Reti

Master Degree Thesis

Data Augmentation for Imbalanced Fund-Policy Classification in Financial NLP: A Comprehensive Evaluation

Supervisors
Prof. Riccardo Coppola

Candidate
Daven Loris Speranza

Academic Year 2024-2025

Acknowledgements

Mi prendo questo spazio per ringraziare la mia famiglia, Mamma e Papà, che mi hanno sostenuto in questi cinque anni: sempre presenti ma mai invasivi, lasciandomi libero di gestire le mie scelte senza mai farmi sentire solo; e Smilla per le migliaia di volte in cui è venuta a bussare alla porta interrompendo le mie due ore consecutive di studio in quei lunghi pomeriggi estivi e invernali: continua sempre a farlo. Un grazie speciale va a Chiara che, da otto anni, è al mio fianco e nel bene e nel male mi ha sopportato e supportato, aiutandomi in tanti momenti a capire quale fosse la strada migliore. Infine, un ringraziamento a tutti i miei amici e, soprattutto, ai coffee pomeridiani a Villa che hanno spezzato le giornate di studio: una costante in questi anni.

Abstract

The asset-management industry produces long, jargon-dense policy texts (e.g., KIIDs) that must be classified along several taxonomies (asset class, geographic area, and others) for compliance and analytics. Manual labeling is costly and not always objective. Modern encoders can read these documents from start-to-finish, but long inputs and imbalances makes the problem challenging in practice.

Can textual data augmentation (DA) mitigate class imbalance and improve fund-policy classification without harming label significance? More specifically: (i) how do lightweight local DA methods compare with online paraphrasing; (ii) when do long-context encoders (*Longformer-base*) materially outperform short-context ones (*RoBERTa-base*); and (iii) what gains, if any, derive from applying these augmentations?

We build a supervised model over KIID investment policies and evaluate two encoder families. We test selective/noising DA and controlled paraphrases, with validation checks to verify augmentations (semantic similarity, clustering indices, PCA maps). Models are tuned with a fixed validation protocol and assessed on macro-/weighted-F1, accuracy, and per-class recall; analyses are made over taxonomy and input length.

We will show how DA consistently improves macro-averaged metrics and minority-class recall compared to baselines, with the simplest mixes delivering robust, low-variance gains. LLM paraphrases can add diversity and usually increase DA related gains but require attention to avoid label drift. *Longformer-base* outperforms *RoBERTa-base* on long policies when label features is dispersed across paragraphs; on short curated inputs, *RoBERTa* remains competitive at lower cost. Augmentation validation confirm semantic preservation for local method and identify risky example drift in generative DA.

Textual DA is a reliable, low risk and resource requirement (with respect to increase the real world input example) remedy for imbalance in fund-policy classification, and encoder choice should follow the corpus length profile.

Future work could include cost-sensitive training and calibrated decision rules; stricter filters for generative DA and implementation of new local technique; larger ad hoc pre-training and error analysis to target classes that remain poorly managed.

Contents

Chapter 1

Introduction

1.1 Context and Motivations

In recent years, the amount of textual documentation produced by the asset management industry has grown exponentially: prospectuses, product sheets, management commentaries, and risk reports represent just a few of the sources an analyst must consult on a daily basis. Automating the classification of such texts, for example, by grouping funds by *asset class*, geographic area, or management style, has a direct impact on *compliance* activities, risk monitoring, and, more generally, the quality of information available to advisors and institutional investors. A reliable classification system reduces due diligence time and costs, limits manual errors, and enables large-scale analyses that are difficult to achieve with traditional approaches.

1.2 Problem definition

The analyzed problem is an active managed Funds classification problem, in which, over different characteristics that could describe the Fund one wants to divide them automatically. Until now, the problem requires active involvement of domain experts that reading the Fund's *Key Investor Information Document (KIID)* have to deduct the rightful information and extrapolate from them the right classification for the analyzed category. The point of the project is to create a model that manages to find the features that distinguish each class giving it a text that contains the investment policy of the fund. The main problem found during the early stage of the model construction and implementation was the inequity in the class elements' number; from theory, we know that an imbalanced dataset given to a model could create a wrong statistical representation inside of it and make it learn the representative inequalities instead of the semantical and statistical difference that comes from the different class affiliation.
These critical problems have driven the project towards to find a solution using *Textual Data Augmentation* that is the name that describe what has been done, namely the homogenization of the number of examples in each class so that features can be learned regardless of the probabilistic distribution within the class. The model indeed does not have to downgrade the probability of assigning a class to a text based on the number

of elements of the class present in the dataset that it has used to *learn*. Basically what happen is that the model in absence of a balanced classes tends to give less importance to less represented classes in favor of the one that it *has seen* more and for which the grater number of training data has heavily modified the model parameters, making it more incline in recognizing statistically frequent patterns and giving less importance to errors made within those classes because of the slight decrease in performance that incorrect classification in this context make.
The main focus of the following studies will be dealing whit two conjoint difficulties:

1. **Class Imbalance**: Machine-learning models often exhibit bias and poor generalization on under-represented categories. In real-world datasets, the distribution of samples can be highly skewed-some asset classes may have thousands of KIID documents while niche classes have only a few dozen. This imbalance stems from the varying popularity of markets: widely traded markets generate abundant documentation, whereas specialized or emerging markets produce far fewer samples.

2. **Domain Specificity**: KIID texts contain specialized financial terminology that is under-represented in general-purpose language models. Without targeted domain adaptation or fine-tuning, such models struggle to capture nuanced investment concepts and to replicate the expertise of a human analyst.

In summary, we tackle the problem of automating the classification of actively managed funds from KIID investment policies, in a setting where labels are both imbalanced and expressed through highly specialized financial language. The core objective of this thesis is to assess whether textual data augmentation can alleviate class imbalance without distorting label semantics, and how far this strategy can be pushed when combined with modern transformer encoders. In the following chapters we formalize the text-classification setup, describe the models and augmentation families adopted, and present an empirical evaluation on multiple taxonomies specific of our dataset, with particular attention to minority-class behavior and the practical applicability of the proposed pipeline in real-world asset-management workflows.

Chapter 2

System model

This chapter provides the theoretical and methodological background underlying our automated fund classification pipeline. Our target inputs are *policy texts* extracted from Key Investor Information Documents (KIIDs) and related fund literature, which we cast as natural-language sequences over a financial lexicon. We outline the evolution of the models used to transform such texts into quantitative objects amenable to statistical learning: from sparse, orderless count vectors to dense distributed representations, and then to contextual encoders based on self-attention. Along the way we highlight task-specific challenges (domain terminology, class imbalance, and few-shot regimes) that motivate the architectural choices made in this thesis. We finally focus on the encoder families actually used in our experiments (RoBERTa and Longformer) and discuss why they are well suited for long, domain-specific documents such as KIIDs.

2.1 Text classification theory

We start from classical text representations used as baselines in our pipeline and progressively move to modern contextual encoders; this section establishes the notation and the sequence of models that map raw texts into vectors for downstream classification.

We will arrive at the end to the description of the actual model used in our real-world application.

2.1.1 Learning objective

Let $(X, Y) \sim \mathcal{P}$, where $X \in \mathcal{X}$ is a text and $Y \in \mathcal{Y} = \{1, \ldots, K\}$ is a class label. A classifier f_θ (where θ are the model parameters) maps a text to class probabilities $p_\theta(y \mid x)$. The goal is to minimize the *expected* risk, using 0-1 loss (apply an indicator function that highlights when the classification is incorrect and set to zero when it is correct):

$$\mathcal{R}(f) \;=\; \mathbb{E}_{(X,Y)\sim\mathcal{P}}\big[\mathbf{1}\{\arg\max_{k} p_f(k \mid X) \neq Y\}\big],$$

whose Bayes-optimal decision rule selects the label with highest posterior $\eta_k(x) \doteq \mathbb{P}(Y = k \mid X = x)$. Because 0–1 loss is non-differentiable, in the minimization process a convex surrogate is used:

$$\hat{\mathcal{R}}(\theta) \;=\; \frac{1}{N}\sum_{i=1}^{N} \ell(p_\theta(\cdot \mid x_i), y_i), \quad \ell(p,y) \;=\; -\log p_y,$$

optionally with regularization.

2.1.2 Representation

A key design choice is the feature map $\phi : \mathcal{X} \to \mathbb{R}^d$ that is what will be given in input to the classifier. Classical vector-space models (Bag-of-Words, TF–IDF) encode token frequency but throw away order. Distributed representations (word/subword/document embeddings) capture semantic similarity via dense vectors. Neural networks encoders (CNN, RNN, Transformer) *learn* ϕ end-to-end, producing token states sensitive to the context $h_1, \ldots, h_n$ and a whole document representation $g(h_{1:n})$ (e.g., [CLS]). For policy-style texts where evidence is found in different locations rather then all together across sentences, the preservation of word order and long-range dependencies is particularly valuable.

2.1.3 Document length and aggregation

Given a token sequence $(t_1, \ldots, t_n)$, the classifier ultimately aggregates information across positions:

$$f_\theta(x) \;=\; \mathrm{softmax}\big(W\, g_\theta(h_1, \ldots, h_n) + b\big).$$

Two theoretical constraints emerge: (i) *information coverage*: truncation can remove decisive spans and distort $\eta_k(x)$; (ii) *computational scaling*: full self-attention is $\mathcal{O}(n^2)$, motivating architectures/approximations that preserve the majority of the information but without quadratic cost. These constraints justify using encoders with sufficient effective context to aggregate dispersed cues coherently but not just merely taking the whole text into account as it is.

2.1.4 Imbalance and cost-sensitive risk

In our data some classes are common and others are rare. Let $\pi_k = \mathbb{P}(Y{=}k)$ be the (true) frequency of class k. When π_k are very different, a model trained to minimize average error tends to “not take too much risk” and predict the majority classes too often, just because it is randomly easier to pick that one with respect to the others.

How do we ideally decide a label? For any document x, let

$$\eta_k(x) \;=\; \mathbb{P}(Y{=}k \mid X{=}x)$$

be the model’s estimated probability for class k. If all mistakes cost the same, the best rule is simply

$$\hat{y}(x) \;=\; \arg\max_k \eta_k(x),$$

i.e., pick the class with the highest probability.

Mistakes have different weights. Sometimes a wrong prediction hurts more in one direction than another (e.g. in a different context; is better to check a patient that is resulted positive to a test twice then not check him at all). We encode this with a *cost matrix* $C \in \mathbb{R}_+^{K\times K}$, where $C_{k,j}$ is the cost of predicting j when the true class is k (by definition $C_{k,k}$=0). Now the decision that minimizes the *expected cost* is

$$\hat{y}(x) \;=\; \arg\min_{j} \; \sum_{k=1}^{K} C_{k,j}\,\eta_k(x).$$

Intuition: for each candidate prediction j, we average the costs of being wrong, weighted by how likely each true class k is for this x, and we choose the smallest.

Imbalance is not only about data counts; it is about the *relative harm* of different mistakes, overall accuracy could not be that much affected by errors made in really small classes. Cost-sensitive learning makes this explicit and turns model probabilities into decisions that match the real priorities of our task.

2.1.5 Few-shot regimes and inductive bias

When $N_k \ll N$ for some classes, variance dominates and overfitting is likely. Generalization improves by injecting *prior structure*: strong representations (pretrained language models, of which we will talk better in §2.6), parameter sharing across labels, and explicit regularizer (e.g., early stopping, dropout). From a statistical viewpoint, transfer learning reduces sample complexity by constraining f_θ (the classifier) to a hypothesis class already adapted to language usual characteristics; a careful fine-tuning then specialize it to the task without destroying priors that we have injected.

2.1.6 Domain specificity and shift

If pretraining data and target texts follow different distributions (e.g. pretraining on novel texts to learn language pattern and then classify text with finance jargon, numeric patterns), we face domain shift. A standard view decomposes target error into (i) source error, (ii) a divergence term between source and target feature distributions, and (iii) an irreducible joint error. Mitigations include domain-adaptive pretraining, tokenizer choices that reduce out-of-vocabulary fragmentation, and objectives that align source/-target representations so that a single decision boundary transfers.

The theory above isolates the levers that govern success in text classification: expressive yet stable representations with sufficient context, cost-aware learning under skew, controlled augmentation for variance reduction, and distribution-aware adaptation. These principles guide the methodological choices developed in the remainder of this work, without committing to any specific implementation here.

2.2 Traditional NLP approaches

Textual data have unique difficulties compared to numerical data or other type of inputs such as images that are easier to define inside a matrix. First, natural language is *discrete*: words are symbolic units, and small changes (e.g., synonym substitutions) can drastically alter meaning. Second, language exhibits *high variability and ambiguity*: the same concept can be expressed in many different ways, while the same word may take multiple meanings depending on context in which is inserted. Third, texts are typically *high-dimensional and sparse*: even modest document can generate vocabularies of tens of thousands of unique tokens, leading to extremely large feature spaces. Finally, unlike vectors of fixed size, documents vary in length and contain sequential dependencies that must be captured to preserve semantics. These challenges explain why traditional frequentists representations such as Bag-of-Words and TF-IDF were initially favored: they offer a simple way to map symbolic sequences into numerical form, that is what we like do deal with, but at the cost of discarding word order and contextual information. Before the advent of deep learning and attention-based architectures, natural language processing (NLP) primarily relied on statistical representations of text. In this section, we review the most common traditional approaches, namely Bag-of-Words, TF–IDF, and distributed word embeddings.

2.2.1 Bag-of-Words (BoW)

The Bag-of-Words (BoW) model is one of the earliest and simplest ways to represent documents numerically. The key idea is to discard the order of words and encode each text as a set of tokens. Given a vocabulary $\mathcal{V} = \{w_1, \dots, w_{|\mathcal{V}|}\}$, every document d is mapped to a vector

$$\mathbf{x}_d = (f(w_1, d), f(w_2, d), \dots, f(w_{|\mathcal{V}|}, d)) \in \mathbb{R}^{|\mathcal{V}|},$$

where $f(w_i, d)$ counts the occurrences of word w_i in d. This produces a high-dimensional and sparse vector space, which is often referred to as the *document-term matrix.*

Histogram view. Each document is equivalently described by a histogram of word counts. For example, consider the text: *“Mary had a little lamb, little lamb, little lamb, Mary had a little lamb, its fleece as white as snow.”* The Bag-of-Words (BoW) representation simply counts how many times each vocabulary item occurs and create the couple word-frequency. Importantly, BoW refers to a *fixed* vocabulary and maps any out-of-vocabulary token to a special *unk* symbol (unknown).

Token index	1	2	3	4	5	6	7	8	9	10
Word (vocabulary)	*mary*	*lamb*	*little*	*big*	*fleece*	*white*	*black*	*snow*	*rain*	*unk*
Count in the text	2	4	4	0	1	1	0	1	0	7

Figure 2.1: BoW histogram for the example sentence using a fixed vocabulary of ten items. Tokens not present in the vocabulary (here *had*, *a*, *its*, *as*) are mapped to *unk*, that added make a total of 7 counts.

This approach, although discarding syntax and word order, preserves information about the topic of the text.

Probabilistic interpretation. The BoW representation can be linked to a multinomial distribution over the vocabulary. If a document d of length $n = |d|$ is viewed as n independent draws of tokens from $\mathcal{V}$, then

$$p(\mathbf{x}_d \mid \boldsymbol{\theta}) = \frac{n!}{\prod_{i=1}^{|\mathcal{V}|} x_{d,i}!} \prod_{i=1}^{|\mathcal{V}|} \theta_i^{x_{d,i}},$$

where $\mathbf{x}_d = (x_{d,1}, \ldots, x_{d,|\mathcal{V}|})$ is the count vector and $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_{|\mathcal{V}|})$ are the word probabilities with $\sum_i \theta_i = 1$. A common Bayesian prior on $\boldsymbol{\theta}$ is the Dirichlet distribution, $\boldsymbol{\theta} \sim \mathrm{Dir}(\alpha)$, leading to a Dirichlet-multinomial model of documents [20].

Connection to classification. BoW representations are often used in linear classifiers, especially the Naive Bayes classifier (NBC). Under the assumption of conditional independence of words given the class y, the class-conditional distribution takes the form

$$p(\mathbf{x}_d \mid y = c) = \prod_{i=1}^{|\mathcal{V}|} p(w_i \mid y = c)^{\,x_{d,i}}.$$

Class priors $p(y = c)$ are typically estimated from frequencies, while word likelihoods $p(w_i \mid y = c)$ can be obtained via maximum likelihood or Bayesian estimation with Dirichlet priors; see [17] for a systematic comparison of event models for multinomial Naive Bayes in text.

Example: Spam detection with word frequencies (with BoW + Multinomial NBC. A binary classification labelling is applied to a set of email: $y \in \{\text{spam}, \text{ham}\}$, each email is represented by its BoW histogram $\mathbf{x} = (x_1, \ldots, x_{|\mathcal{V}|})$ with x_i is the count of token w_i.

Taking into account the multinomial NB assumption of conditionally indipendence of words, the class-conditional likelihood is

$$p(\mathbf{x} \mid y = c) \;\propto\; \prod_{i=1}^{|\mathcal{V}|} \phi_{ci}^{x_i}, \qquad \phi_{ci} = p(w_i \mid y = c), \qquad c \in \{\text{spam}, \text{ham}\}.$$

Training (from labeled emails). Let n_{ci} be the total count of word w_i across all training emails of class c, and $N_c = \sum_i n_{ci}$ the total number of tokens in class c. Estimate word probabilities:

$$\hat{\phi}_{ci} \;=\; \frac{n_{ci}}{N_c}, \qquad \hat{\pi}_c \;=\; \frac{\#\{\text{emails in } c\}}{\#\{\text{all emails}\}},$$

where $\hat{\pi}_c$ is the class prior.
Decision rule. Predict *spam* iff the posterior log-odds is positive:

$$\log \frac{p(y = \text{spam} \mid \mathbf{x})}{p(y = \text{ham} \mid \mathbf{x})} \;\propto\; \log \frac{\hat{\pi}_{\text{spam}}}{\hat{\pi}_{\text{ham}}} \;+\; \sum_{i=1}^{|\mathcal{V}|} x_i \underbrace{\log \frac{\hat{\phi}_{\text{spam},i}}{\hat{\phi}_{\text{ham},i}}}_{w_i} \;>\; 0.$$

Words that occur proportionally more often in spam (e.g., `free`, `winner`, `iPhone`) get positive w_i; words typical of ham (e.g., `meeting`, `project`) get negative w_i. The histogram view makes it clear: frequent spam words push the sum up.

Strengths and weaknesses. BoW provides a simple, interpretable, and computationally efficient baseline, especially useful for tasks such as spam detection or topic classification. However, it suffers from severe limitations: (i) dimensionality grows with vocabulary size; (ii) the representation is sparse; (iii) semantic similarity between words is ignored; (iv) no word order or context structure is known in vectorized representation. These problems motivated the development of weighted variants (e.g., TF–IDF).

2.2.2 Term Frequency–Inverse Document Frequency (TF–IDF)

To mitigate the dominance of frequent but semantically poor words (e.g., articles, prepositions), the BoW model is often weighted by the inverse document frequency [26]. The TF–IDF weight of word w in document d is defined as

$$\mathrm{tfidf}(w, d) = \mathrm{tf}(w, d) \cdot \mathrm{idf}(w),$$

where

$$\mathrm{tf}(w, d) = \frac{f(w, d)}{\sum_{w' \in d} f(w', d)} \quad \text{and} \quad \mathrm{idf}(w) = \log \frac{N}{1 + |\{d' \in \mathcal{D} : w \in d'\}|}.$$

Here, $N = |\mathcal{D}|$ is the total number of documents in the corpus. The resulting representation still lies in $\mathbb{R}^{|\mathcal{V}|}$ but downweights ubiquitous terms while upweighting words that are more discriminative across documents.

2.2.3 Word embeddings

Both BoW and TF–IDF suffer from two fundamental limitations: (i) they ignore the semantic similarity between words, and (ii) they do not capture word order or contextual dependencies. To overcome this, distributed representations of words were introduced. In this paradigm, each word $w \in \mathcal{V}$ is mapped to a dense vector $\mathbf{v}_w \in \mathbb{R}^k$, with $k \ll |\mathcal{V}|$, such that semantically similar words are close in the embedding space.

The *embedding space* is the k-dimensional vector space $\mathbb{R}^k$ where these representations live. Geometrically, each word corresponds to a point in this space, and semantic relationships are reflected in their relative positions: words that frequently occur in similar contexts (e.g., “equity” and “stock”) have vectors with small Euclidean or cosine distance, while unrelated words (e.g., “equity” and “pasta”) lie far apart; so the idea is that looking in the embedding space one can recognize the meaning of a word just by looking at its neighborhood. This continuous representation allows algebraic operations with vector that are “interpretable”; for example, in many embedding models

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

a phenomenon first highlighted for Word2Vec by [19].

The most used models are Word2Vec and GloVe.

Word2Vec learns embeddings via a shallow neural network trained on a predictive objective [18]. Two variants exist:

- **Continuous Bag-of-Words (CBOW):** predicts a target word w_t given its surrounding context $C_t = \{w_{t-m}, \dots, w_{t+m}\}$.

$$\max_{\theta} \sum_{t=1}^{T} \log p(w_t \mid C_t; \theta).$$

- **Skip-gram:** predicts context words given a target word.

$$\max_{\theta} \sum_{t=1}^{T} \sum_{-m \le j \le m, j \ne 0} \log p(w_{t+j} \mid w_t; \theta).$$

The conditional probabilities are modeled using the softmax function:

$$p(w_o \mid w_i) = \frac{\exp(\mathbf{v}'_{w_o} \cdot \mathbf{v}_{w_i})}{\sum_{w \in \mathcal{V}} \exp(\mathbf{v}'_w \cdot \mathbf{v}_{w_i})},$$

where $\mathbf{v}_{w_i}$ and $\mathbf{v}'_{w_o}$ are input and output embeddings.
GloVe [21], instead, is based on a matrix factorization approach that leverages global co-occurrence statistics. Let X_{ij} denote the number of times word w_j occurs in the context of w_i. GloVe learns embeddings $\mathbf{v}_i, \mathbf{u}_j \in \mathbb{R}^k$ such that

$$\mathbf{v}_i^\top \mathbf{u}_j + b_i + c_j \approx \log X_{ij},$$

minimizing the weighted least squares objective

$$J = \sum_{i,j=1}^{|\mathcal{V}|} f(X_{ij}) \Big(\mathbf{v}_i^\top \mathbf{u}_j + b_i + c_j - \log X_{ij} \Big)^2,$$

where $f(\cdot)$ is a weighting function that limits the influence of very frequent occurrences. Here, b_i and c_j are scalar terms associated with the target word w_i and the context word w_j, respectively. They allow the model to better fit by absorbing global frequency effects that cannot be captured by the dot product of embeddings alone.

2.2.4 Paragraph vector

In some context the vector representation of a single word could be not enough if we are looking for an entire sentence or documents to be vectorized. To address that, a natural extension of Word2Vec is **Doc2Vec** [13], which enlarges it to variable-length texts.

The idea is to associate each document d to a unique vector $\mathbf{d} \in \mathbb{R}^k$ that acts as a global memory of its content.

The used architectures are:

- **Distributed Memory (PV-DM).** Analogous to the CBOW model, PV-DM predicts a target word w_t from both its surrounding context words C_t and the document vector $\mathbf{d}$:

$$\max_{\theta} \sum_{t=1}^{T} \log p(w_t \mid C_t, \mathbf{d}; \theta).$$

  Here, we add the document embedding, that provides additional context, acting as a memory of the topic or its style.

- **Distributed Bag of Words (PV-DBOW).** In this variant, the document vector $\mathbf{d}$ alone is used to predict randomly sampled words from the document, without considering local context:

  $$\max_{\theta} \sum_{w \in d} \log p(w \mid \mathbf{d}; \theta).$$

  This is similar to the Skip-gram model.

Once trained, each document is represented by its learned vector $\mathbf{d}$, which can be used directly for classification, clustering, or retrieval tasks. Doc2Vec embeddings have proven particularly useful when semantic similarity at the document level is required, for example in sentiment analysis, topic classification, or recommendation systems.

2.2.5 Limitations

Despite distributed embeddings significantly improved over BoW and TF-IDF, they remain context-independent so each word has a single vector regardless of its surrounding context. For example, the word "bank" will have the same embedding in "river bank" and "investment bank", which is problematic in every application and especially in our domain of financial texts. This limitation motivated the development of contextual word embeddings so models that know how to read the neighborhood of a word and in this way know how differentiate meaning, later realized through Transformer-based architectures.

2.3 Neural Networks

Before introducing recurrent architectures, we first have to talk about standard feed-forward neural networks (NNs), see Fig.2.2. In supervised learning, that means that examples of correct labeling are given to the model from us; we observe input-target pairs $(\mathbf{x}, \mathbf{y})$, where the input $\mathbf{x} \in \mathbb{R}^d$ is a fixed-length numeric representation of an object (e.g., a document encoded as features) and the target $\mathbf{y}$ encodes the label. A neural network is a *parametric* function obtained by stacking simple modules called *layers*. Each layer takes a vector and applies an affine map followed by a pointwise nonlinearity. If we denote the width of layer l by n_l (with $n_0 = d$), the l-th layer maps $\mathbb{R}^{n_{l-1}} \to \mathbb{R}^{n_l}$ via a weight matrix $W^{(l)} \in \mathbb{R}^{n_l \times n_{l-1}}$ and a bias $\mathbf{b}^{(l)} \in \mathbb{R}^{n_l}$. The bias shifts the activation thresholds and allows the layer to fit functions that do not necessarily, imposed by the non linear layer, pass through the origin. The nonlinearity ϕ (e.g., ReLU (see Fig.2.3) or tanh) is essential: without it, a stack of layers could sum up to a single affine transformation and could only model linear decision boundaries. Depth (using several layers) enables hierarchical feature extraction, where early layers compute generic patterns and later layers could find more specif feature. If we are using mini-batches, inputs are grouped as a matrix $X \in \mathbb{R}^{d \times B}$.

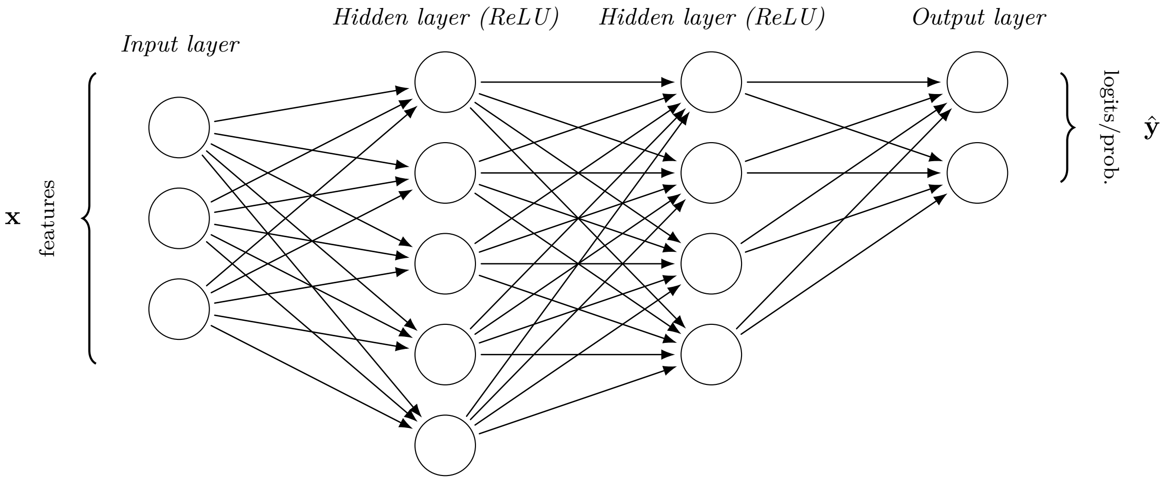


Figure 2.2: Multi-layer perceptron with two ReLU hidden layers. Every neuron receives inputs from *all* units in the previous layer (dense connections). Each hidden layer computes $h = \mathrm{ReLU}(W\mathbf{x} + b)$. On the right, the output layer returns logits $\hat{\mathbf{y}}$ that are converted to class probabilities via softmax.

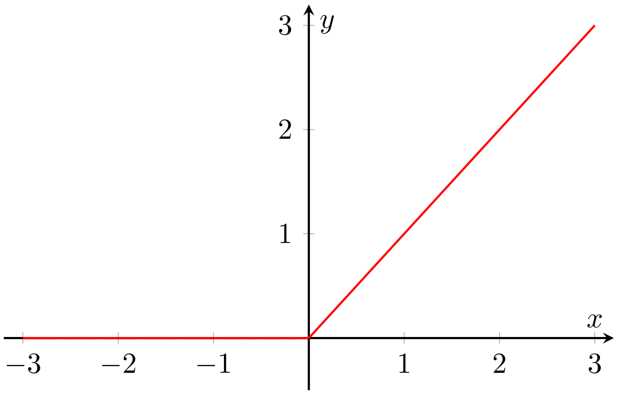


Figure 2.3: $\mathrm{ReLU}(x) = \max(0, x)$. It keeps positive inputs unchanged and sets negatives to zero. The derivative is 0 for $x < 0$ and 1 for $x > 0$ (undefined at $x = 0$, typically set to 0 or 1 in practice). ReLU is cheap to compute, mitigates vanishing gradients, and induces sparse activations; a known caveat is 0 output when the input in negative.

With L layers, the computation reads

$$\begin{aligned}
\mathbf{h}^{(0)} &\doteq \mathbf{x}, \\
\mathbf{a}^{(l)} &\doteq W^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}, \qquad && l = 1, \ldots, L, \\
\mathbf{h}^{(l)} &\doteq \phi^{(l)}(\mathbf{a}^{(l)}), && l = 1, \ldots, L-1, \\
\hat{\mathbf{y}} &\doteq \psi(\mathbf{a}^{(L)}),
\end{aligned}$$

where $W^{(l)}$ and $\mathbf{b}^{(l)}$ are trainable, $\phi^{(l)}$ is a nonlinearity, and ψ is an output activation chosen for the task. For multi–class classification with K classes, ψ is typically the softmax

$$\mathrm{softmax}(\mathbf{z})_k \;=\; \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}}, \quad k = 1, \dots, K,$$

so that $\hat{\mathbf{y}} \in \Delta^{K-1}$ represents class probabilities.

2.3.1 Loss and empirical risk

Given training pairs $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N}$, learning minimizes an empirical risk

$$\mathcal{L}(\theta) \;=\; \frac{1}{N}\sum_{i=1}^{N} \ell(\hat{\mathbf{y}}_i, \mathbf{y}_i) \;+\; \lambda\,\Omega(\theta),$$

where $\theta = \{W^{(l)}, \mathbf{b}^{(l)}\}_{l=1}^{L}$, ℓ is a task–specific loss, and Ω is an optional regularizer (e.g., $\Omega(\theta) = \frac{1}{2}\sum_l \|W^{(l)}\|_F^2$ for ℓ_2 weight decay) with strength $\lambda \geq 0$.

2.3.2 Training by gradient methods

We consider unconstrained minimization $\min_\theta f(\theta)$. The gradient $\nabla f(\theta)$ points in the direction of steepest *increase*; moving in the opposite direction decreases the objective. Gradient descent performs the iteration

$$\theta_{t+1} \;=\; \theta_t - \eta_t \,\nabla f(\theta_t),$$

with step size (learning rate) $\eta_t > 0$. For a *convex* function with L-Lipschitz gradient, a constant step $\eta_t = \eta \leq 1/L$ ensures monotone decrease and $f(\theta_t) - f(\theta^\star) = \mathcal{O}(1/t)$. In practice we use the *stochastic* version, replacing the full gradient with a mini–batch estimate.

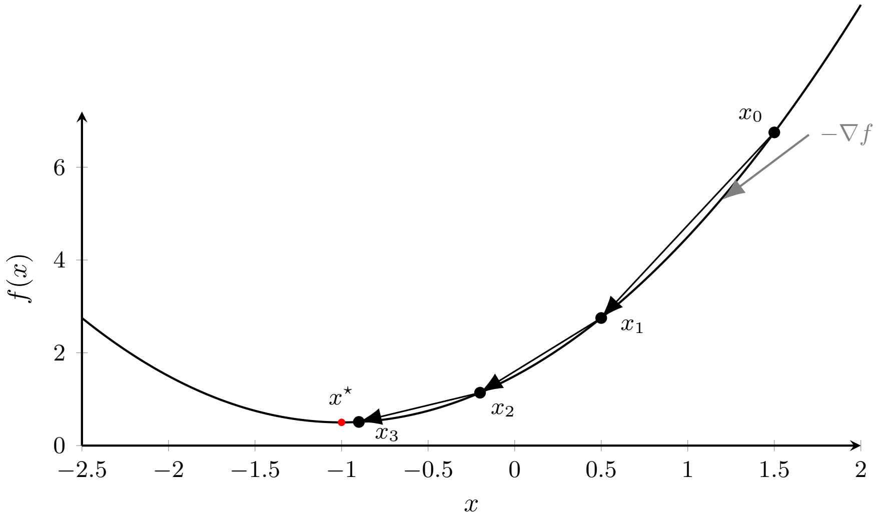


Figure 2.4: Gradient descent on a convex objective $f(x) = (x+1)^2 + 0.5$. Steps move along the negative gradient toward the minimizer $x^\star = -1$ with step size decided using the learning rate.

We train the network by minimizing the empirical loss with (stochastic) gradient methods. Given a mini–batch $\mathcal{B} = \{(x_i, y_i)\}_{i=1}^{B}$, the stochastic gradient is

$$g_t \;\doteq\; \nabla_\theta \left(\frac{1}{B} \sum_{(x_i,y_i)\in\mathcal{B}} \ell(f_\theta(x_i), y_i) \right),$$

and parameters are updated as

$$\theta_{t+1} \;=\; \theta_t \;-\; \eta_t \, g_t,$$

where $\eta_t > 0$ is the learning rate (possibly scheduled over time, that means that we update it during the training steps, is not constant over the entire process). Using mini–batches provides an unbiased, low–variance estimate of the full gradient at a much lower computational cost per step.

Momentum and adaptive methods. Plain SGD can be accelerated with momentum or adaptive preconditioning. With momentum one keeps an exponential moving average (EMA) of past gradients,

$$v_t = \beta \, v_{t-1} + (1-\beta) \, g_t, \qquad \theta_{t+1} = \theta_t - \eta_t \, v_t,$$

where $\beta \in [0,1)$ controls the memory of the velocity. Adaptive methods (Adam/AdamW) keep EMAs of both the gradient and its elementwise square:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t, \quad s_t = \beta_2 s_{t-1} + (1-\beta_2) g_t^{\odot 2},$$

with bias–corrected moments $\hat{m}_t = m_t/(1-\beta_1^t)$ and $\hat{s}_t = s_t/(1-\beta_2^t)$. The Adam update is

$$\theta_{t+1} = \theta_t - \eta_t \, \frac{\hat{m}_t}{\sqrt{\hat{s}_t} + \varepsilon}.$$

Then to improve generalization we use decoupled weight decay (AdamW). Instead of adding ℓ_2 directly to the loss, AdamW applies a separate shrinkage step:

$$\theta_{t+\frac{1}{2}} = \theta_t - \eta_t \, \frac{\hat{m}_t}{\sqrt{\hat{s}_t} + \varepsilon}, \qquad \theta_{t+1} = \theta_{t+\frac{1}{2}} - \eta_t \, \lambda \, \theta_t,$$

where $\lambda \geq 0$ is the weight–decay coefficient. This “decoupling” avoids interactions between adaptive rescaling and regularization.

Practicalities. We use mini–batches, AdamW, and (when needed) global–norm gradient clipping $g_t \leftarrow g_t \cdot \min\big(1, \tau/\|g_t\|_2\big)$ to stabilize training. Learning rates are tuned per model and can follow a schedule (precisely a few steps of warm–up followed by cosine decay in our main application as we will see). All parameters are learned jointly by backpropagation.

2.3.3 Limitation of feed-forward NNs

A feed–forward network processes $\mathbf{x}$ as a whole and produces $\hat{\mathbf{y}}$ in one pass; it does not model order within a sequence. Recurrent Neural Networks (next section) retain the same building blocks (affine maps, nonlinearities, loss minimization by backpropagation), but they *reuse* a layer across positions and maintain a hidden state that evolves in time. This simple change introduces an inductive bias for sequences while preserving the training machinery outlined above.

2.4 Recurrent Neural Networks

A significant step beyond static word embeddings is the *Recurrent Neural Network* (RNN). Unlike feed-forward models, which process each input independently, an RNN maintains a *hidden state* that is updated at every time step. Given a sequence, the state at time t depends on the current input and on the previous state, so the computation at t is context-aware with respect to the tokens $1{:}t-1$. Crucially, the same parameters are reused at each step (weight sharing), and they are learned jointly via backpropagation through time. In this way, RNNs encode order and short- to medium-range dependencies within the sequence, rather than treating tokens in isolation. Given a sequence of tokens $d = (w_1, w_2, \dots, w_T)$, an RNN updates a hidden state $\mathbf{h}_t$ at each step according to the current input and the previous state:

$$\mathbf{h}_t = \sigma(W_h \mathbf{h}_{t-1} + W_x \mathbf{x}_t + \mathbf{b}) \,,$$

where $\mathbf{x}_t$ is the embedding of token w_t, W_h and W_x are trainable weight matrices, $\mathbf{b}$ is a bias term, and $\sigma(\cdot)$ is a nonlinear activation (typically tanh or ReLU). This recursive formulation lets information propagate through time and, in principle, enables the network to capture sequential dependencies.

2.4.1 How to read the recurrence

The equation above defines a dynamical system unrolled over the sequence: at $t{=}1$ the model ingests $\mathbf{x}_1$ and produces $\mathbf{h}_1$; at $t{=}2$ it receives $\mathbf{x}_2$ *and* the summary $\mathbf{h}_1$, and so on. Because the same parameters are reused at every step, the model can handle variable-length inputs and learns generic rules for composing local information into a running summary. In practice, we turn the sequence of hidden states $\{\mathbf{h}_t\}_{t=1}^{T}$ into task-specific outputs in simple ways:

- *Many-to-one* (document/sentence classification): pool the hidden states (last state $\mathbf{h}_T$, mean/max pooling, or a small attention mechanism) and feed the result to a classifier.

- *Many-to-many* (sequence tagging): apply a linear layer (and softmax) to each $\mathbf{h}_t$ to predict a label per token.

- *Bidirectional context* (useful for tagging): run one RNN forward and one backward, then concatenate $[\overrightarrow{\mathbf{h}}_t; \overleftarrow{\mathbf{h}}_t]$ before prediction to use past and future context simultaneously.

Training is performed with *backpropagation through time* (BPTT), i.e., we conceptually “unfold” the recurrence into a chain and apply standard backpropagation on that chain.

2.4.2 What RNNs capture

RNNs are effective because they read the text in order and maintain a running summary of what has been seen so far. This sequential processing lets them encode meanings that

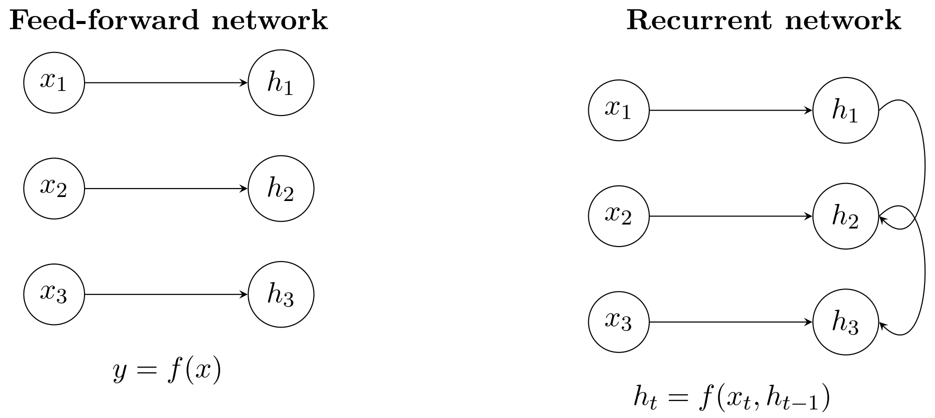


Figure 2.5: Comparison between a feed-forward network (left) and a recurrent neural network (right).

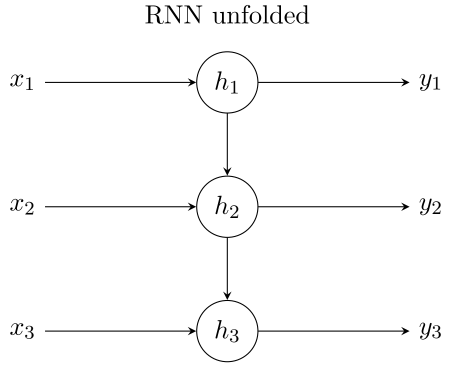


Figure 2.6: Unfolded representation of a recurrent network across three time steps. Shared parameters are reused at each step.

depend on word order (for example, "*may not invest in*" versus "*may invest in*") and track the scope of negation and quantifiers as they appear. While moving through a sentence, the hidden state naturally absorbs short-range syntactic cues—such as determiner–noun or modifier–head relations—that help disambiguate instruments and asset types (e.g., "*government bonds*", "*emerging markets equities*"). Investment policy texts also exhibit highly regular phrase templates (e.g., "*invests primarily in …*", "*at least 70% in …*", "*exposure to … up to …*"); when such patterns occur, the hidden state tends to shift in consistent directions that correlate with the target class. Beyond isolated phrases, the model composes evidence over the sentence: as it encounters asset class, geography, style/sector, and constraints, later hidden states reflect the *combination* of these pieces rather than just their presence. The same applies to quantified patterns—thresholds and percentages like "*at least 50%*" or "*no more than 10%*" carry class information that is captured in context. Finally, the discourse flow typical of policy documents—core allocation rules first, then exceptions—can be mirrored in the evolving state as the text shifts from "main rule" to "conditions".

In summary, by maintaining a state that evolves with the sequence, RNNs can represent *what* was said, *how* it was qualified (negations, thresholds), and *where* it appeared, which is exactly the kind of structure present in investment policy texts.

2.4.3 Vanishing and exploding gradients

In practice, simple RNNs face serious limitations when trained on long sequences. Backpropagation through time (BPTT) propagates gradients across all steps; because each step multiplies by the recurrent Jacobian, the signal can change magnitude exponentially with the number of steps. If the effective gain is slightly below one, gradients shrink rapidly (*vanishing*); if it is above one, they blow up (*exploding*) [2]. This behavior causes the model to only using very recent context, unstable updates, and highly variable training loss indeed deep layers receive very little weight update, and the network struggles to learn long-range dependencies.

The hidden-state update repeatedly applies the same linear map (plus a nonlinearity) to pass information forward; BPTT applies its transpose to pass information backward. Over many steps, these repeated multiplications either dampen or amplify the signal. This is why vanilla RNNs struggle to learn long-range dependencies even if they represent short patterns well.

Consider a vanilla RNN with pointwise nonlinearity σ:

$$a_t \;=\; W_h h_{t-1} + W_x x_t + b, \qquad h_t \;=\; \sigma(a_t), \qquad C \;=\; \sum_{t=1}^{T} \ell(h_t).$$

Define the local gradient $g_t := \frac{\partial \ell(h_t)}{\partial h_t}$ and the backprop signal $\delta_t := \frac{\partial C}{\partial h_t}$, and let

$$J_t \;:=\; \frac{\partial h_t}{\partial h_{t-1}} \;=\; \mathrm{diag}(\sigma'(a_t))\, W_h.$$

Then by the use of chain rule, *backprop through time* gives

$$\delta_t \;=\; g_t \;+\; J_{t+1}^{\top}\, \delta_{t+1}$$

Unrolling from t to T yields

$$\delta_t \;=\; g_t \;+\; \sum_{k=t+1}^{T} \left(J_{t+1}^{\top} J_{t+2}^{\top} \cdots J_{k}^{\top} \right) g_k$$

The transport of gradients from time k back to t is controlled by the *product of recurrent Jacobians* $J_{t+1}^{\top} \cdots J_{k}^{\top}$.
Then, derived using sub-multiplicative matrix norm $\|\cdot\|$ properties,

$$\begin{aligned} \|J_{t+1}^{\top} \cdots J_{k}^{\top}\| &\le \prod_{j=t+1}^{k} \|J_j\| \le \prod_{j=t+1}^{k} \|\mathrm{diag}(\sigma'(a_j))\|\, \|W_h\| \\ &\le \left(c_\sigma \|W_h\|\right)^{k-t}, \qquad c_\sigma \;:=\; \sup_z |\sigma'(z)|. \end{aligned}$$

Hence the magnitude of distant contributions in 2.4.3 is upper-bounded by a *geometric* factor:

$$c_\sigma \|W_h\| < 1 \;\Rightarrow\; \text{vanishing (exponential decay)}, \qquad c_\sigma \|W_h\| > 1 \;\Rightarrow\; \text{exploding (exponential growth)}.$$

Intuitively, each time step multiplies the backprop signal by a gain from the nonlinearity ($\sigma'(a_t)$) and by the recurrent map (W_h). Repeating this $k-t$ times yields a power, hence geometric decay or growth.

Scalar case. In the scalar case $h_t = \sigma(wh_{t-1} + ux_t + b)$,

$$\delta_t = g_t + (\sigma'(a_{t+1})\,w)\delta_{t+1} \;=\; \sum_{k=t}^{T} \Big(\prod_{j=t+1}^{k} \sigma'(a_j)\,w \Big) g_k,$$

and with $|\sigma'(a_j)| \le c_\sigma$ we get $\big|\prod_{j=t+1}^{k} \sigma'(a_j) w\big| \le (|w|\, c_\sigma)^{k-t}$: $|w|\, c_\sigma < 1 \Rightarrow$ vanish, $|w|\, c_\sigma > 1 \Rightarrow$ explode.

2.4.4 Limitations of recurrent models

Despite these improvements, recurrent models still suffer from some drawbacks. They process sequences sequentially, making training and inference slower compared to parallelizable architectures. Exist newer model called LSTMs and GRUs can handle moderately long dependencies but their ability still degrades with very long documents such as policies extracted from KIIDs, motivating the adoption of attention-based models that will be introduced in the next section (§2.5).

2.5 The Transformer

Recurrent networks read a sequence one step at a time, which makes long-range dependencies hard and training slow. Convolutional neural networks (CNNs) instead, are computationally efficient because convolutions can be highly parallelized and parameters are shared across space and while each layer has a limited receptive field, the overall receptive field increases as we stack layers, use pooling, larger kernels, or dilated convolutions. But this inductive bias makes CNNs especially effective in image processing and computer vision, whereas in NLP they have largely been supplanted by Transformers for long-range dependencies. *Transformers* remove recurrence entirely and let each token *attend* to all the others in the same layer via *self–attention.* In practice this gives (i) full parallelism over tokens and (ii) direct access to distant context—two advantages for policy texts where relevant cues may be far apart.

2.5.1 Scaled dot-product self-attention

Self–attention lets each token build a *bespoke summary* (meaning that it is custom-made to meet a specific use case and not being a generic overview) of the whole sequence, focusing more on the parts that matter for that token. It does so by turning similarity scores into a probability distribution over other tokens and then taking a weighted average of their information.

Definition of the three multihead projection. Starting from token embeddings $X \in \mathbb{R}^{n\times d}$, we compute

$$Q = XW_Q, \qquad K = XW_K, \qquad V = XW_V.$$

- Q (queries): what each position is *looking for.*
- K (keys): what each position *offers* as a descriptor, this identifies what position matches the search *described* by Q.
- V (values): the *content* to be aggregated that is what is needed to be imported from those matches.

These are three different linear "views" of the same tokens: the dot product $q_i^\top k_j$ measures how well token j matches the needs of token i.
During training, gradients shape W_Q, W_K so that useful pairs have high $q_i^\top k_j$, and shape W_V so that v_j carries the information that, once mixed, helps the loss.

From similarity to attention weights. We turn all pairwise similarities into *attention weights* with a softmax:

$$A \;=\; \mathrm{softmax}\Big(\frac{QK^\top}{\sqrt{d_k}}\Big) \in \mathbb{R}^{n\times n}, \qquad A_{ij} = \frac{\exp\big(\frac{q_i^\top k_j}{\sqrt{d_k}}\big)}{\sum_{t=1}^{n} \exp\big(\frac{q_i^\top k_t}{\sqrt{d_k}}\big)}.$$

- Each row $A_{i:}$ is a *probability distribution* over positions ($\sum_j A_{ij} = 1$, $A_{ij} \geq 0$): "how much should token i listen to token j?"
- The factor $1/\sqrt{d_k}$ stabilizes the scale of $q_i^\top k_j$ so softmax is neither too flat nor saturated as d_k grows.
- A mask M can be added to zero out illegal positions (e.g., future tokens or padding) by adding $-\infty$ to those logits before softmax, this makes their final weight negligible.

Obtained results. The output representations are

$$Z \;=\; AV, \qquad z_i \;=\; \sum_{j=1}^{n} A_{ij}\, v_j.$$

Thus each new token vector z_i is a *convex combination* of the value vectors $\{v_j\}$, with data–dependent weights. In words: token i rewrites itself as a summary of the most relevant pieces in the sequence.

Example on our domain. Consider the policy sentence

> *"The fund invests **at least** 70% in European **equities** and up to 30% in **bonds**."*

Let the token at position i be the word **equities**. In a self–attention layer, **equities** forms a query q_i that "looks for" information useful to interpret its role (constraints, geography,...). Every other token j forms a key k_j and a value v_j. The raw *relevance scores* for **equities** are the dot products

$$s_{ij} = \frac{q_i^\top k_j}{\sqrt{d_k}}, \qquad \text{and } A_{ij} = \frac{\exp(s_{ij})}{\sum_{t=1}^{n} \exp(s_{it})}.$$

Suppose (for illustration) the largest scores for q_i come from the tokens {"70%", "European", "at least"}; after the softmax we might have that these 3 have the biggest values:

$$A_{i,70\%}, \quad A_{i,\text{European}}, \quad A_{i,\text{at least}}.$$

The new representation of **equities** is the weighted average

$$z_i = A_{i,70\%}\, v_{70\%} + A_{i,\text{European}}\, v_{\text{European}} + A_{i,\text{at least}}\, v_{\text{at least}} + \sum_{j \notin \{\cdot\}} A_{ij}\, v_j.$$

where the extra sum represents the residual attention (the distribution must sum to 1), i.e., the small contribution from the other tokens.

Meaning. The vector z_i now carries a *bound* (from "70%"), a *region* (from "European"), and an *operator* indicating directionality (from "at least"), all tied to the head noun **equities**. In one step the model has stitched together *what* (asset class), *how much* (percentage), and *where* (geo-Europe).

2.5.2 Multi–head specialization

In multi–head attention the model runs several *independent attention mechanisms in parallel*, each with its own projections $(W_Q^{(m)}, W_K^{(m)}, W_V^{(m)})$. Each mechanism is a separate "*head*" that looks at the sequence from its own perspective, each one noticing different details. After each head computes its context vector, the results are concatenated and linearly mixed.
With h heads, the same token builds h complementary summaries:

$$z_i^{(m)} = \sum_{j=1}^{n} A_{ij}^{(m)}\, v_j^{(m)}, \qquad Z_i = \text{Concat}(z_i^{(1)}, \ldots, z_i^{(h)}) W_O.$$

Each head learns its own notion of "relevance". In our domain, one head typically attends to *quantitative cues* (percentages, bounds such as "at least" / "no more than"), another to *entities and geography* (regions, countries, indices), and another to *operations and exceptions* (negations or clauses like "except", "derivatives for hedging"). Then a concatenation is made up to merge this views in a final representation.

2.5.3 The role of [CLS]

For sequence classification, encoder–only Transformers (e.g., BERT/RoBERTa style, we will talk more about in next sections) prep end a special token, commonly called [CLS] (that stands for "classification"). It is a *learned embedding* $e_{\text{CLS}} \in \mathbb{R}^d$ placed at position 0, so the input to the first layer is

$$X_0 \;=\; \big[e_{\text{CLS}} + p_0,\; e_{x_1} + p_1,\; \ldots,\; e_{x_n} + p_n\big],$$

where e_{x_t} are token embeddings and p_t are positional encodings. From that moment on, [CLS] is treated like any other token: it queries all positions and is, in turn, visible to them.

Because [CLS] take part to the whole sequence at every layer, its hidden state after L layers,

$$h^{(L)}_{\text{CLS}} \;=\; F^{(L)}(X_0),$$

acts as a *global summary vector*. Gradients from the classification loss flow *only* through the small linear head on top of $h^{(L)}_{\text{CLS}}$, and back into the encoder, consistently bringing the model to store in $h^{(L)}_{\text{CLS}}$ exactly the information that helps predict the label. Is in some way the entire overview of the sequence that can possibly be store in only one token.

Geometric view. At one attention sublayer, the [CLS] update is a convex mixture of value vectors:

$$z_{\text{CLS}} \;=\; \sum_{j=0}^{n} A_{\text{CLS},j}\, v_j, \qquad A_{\text{CLS},j} \geq 0,\;\; \sum_j A_{\text{CLS},j} = 1.$$

Therefore, z_{CLS} lies in the *convex hull* of the current token values. Across heads (multiple mixtures) and across layers (with residual connections (add stability in deep network) and feed–forward transforms), [CLS] repeatedly repositions itself as a sequence–dependent *barycenter* and then passes through learned affine/nonlinear maps. The effect is to move $h^{(L)}_{\text{CLS}}$ into a region of the space where sequences of the same class form tight clusters and different classes are linearly separable by the classifier head

$$p(y \,|\, x) \;=\; \operatorname{softmax}\!\big(W_{\text{clf}}\, h^{(L)}_{\text{CLS}} + b_{\text{clf}}\big).$$

So it's like looking for the best spot in which every class could as easier as possible not be confused with other.

What [CLS] **tends to capture.** In fund policy texts, [CLS] typically aggregates: (i) the dominant *asset* terms (e.g., equities/bonds/commodities), (ii) the key *constraints* (lower/upper bounds, hedging vs. investment use), (iii) the *geographic* scope, and (iv) *exceptions* that change the meaning of the main clause. Because attention is bidirectional in the encoder, [CLS] can simultaneously "listen" to distant cues that live in separate sentences or sections and fold them into a single, compact representation.

This is preatty different from the easier mean pooling because it just averages all token states uniformly. [CLS], instead, learns a *content–aware* weighting through attention, emphasizing informative spans and down–weighting text filler or useless information. Empirically this yields a crisper decision boundary in the embedding space.

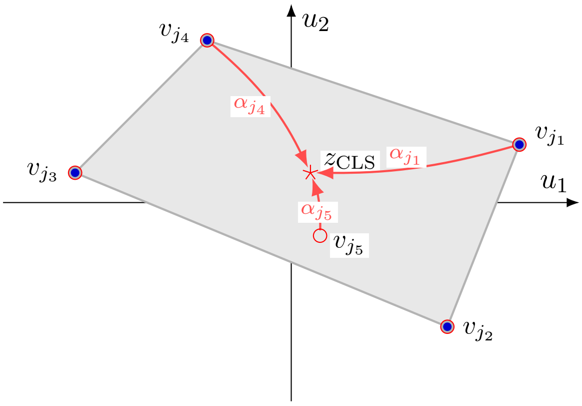


Figure 2.7: Geometric view of a [CLS] update in one attention sublayer. Vectorization in the embedding space (here shrinked to R^2, like what we will see later after applying PCA to the vectorized texts, instead of much higher dimension like R^{768} for RoBERTa-model) of token values v_j (circles) span the convex region $\mathrm{conv}\{v_j\}$ (grey polygon, that is the smaller polygon that can contain all the points). The updated summary z_{CLS} (red star) is a convex combination $z_{\text{CLS}} = \sum_j \alpha_j v_j$ with $\alpha_j \geq 0$ and $\sum_j \alpha_j = 1$, hence it lies *inside* the convex hull (for clarity, only the strongest contributors are highlighted; all tokens contribute with (often tiny) weights (α_j)). Repeating this convex mixing across heads and layers (and applying affine/nonlinear maps) steers [CLS] toward areas of the space where classes are more easily separable.

2.5.4 Why distance does not matter

If "*equities*" appears in paragraph 1 and "*70%*" in paragraph 3, the computation is unchanged:

$$A_{ij} = \mathrm{softmax}(q_i^\top k_j / \sqrt{d_k}),$$

which compares *all* pairs independently of position. Positional encodings P have to be added to X so that q_i and k_j still know *where* tokens sit exactly because of the independency of attention from distance and the cost of connecting distant cues is the same.

2.5.5 Encoder vs. decoder

Think of the model as two cooperating modules with different jobs:

- **Encoder = reader.** It *reads and understands* the whole input at once. Because its self-attention is bidirectional, each word can look left and right to gather the most relevant context. The output is a matrix H_{enc}-a *memory* of the source-where each row is a context-aware vector for one input token.

- **Decoder = writer.** It *writes* the output one token at a time. Its first attention is *masked*, so at step t it can only use what it has already written $(y_{<t})$. Then it performs *encoder–decoder attention*: it asks the encoder's memory *"Which sources are relevant at the moment?"*. This is how alignment emerges (e.g., which source words support the next target word).

Why they difference. Understanding benefits from seeing *all* context at once (bidirectional attention), while generation must be *causal* (no peeking at the future). The encoder therefore builds a good representation of the input; the decoder turns that representation into an output sequence, consulting the input memory when needed.

Information flow.

$$\underbrace{\text{source } x_{1:n}}_{\text{to encoder}} \xrightarrow{\text{Encoder}} \underbrace{H_{\text{enc}}}_{\text{source memory}} \xrightarrow{\text{cross-attn}} \underbrace{h^{(L)}_{\text{dec},t}}_{\text{decoder state at } t} \xrightarrow{\text{Linear+Softmax}} p_\theta(y_t \mid y_{<t}, x).$$

At the next step $t{+}1$, the decoder repeats the loop with the newly generated token in its history. So it will be able to see all the information given by the encoder + its own generation until the past step.

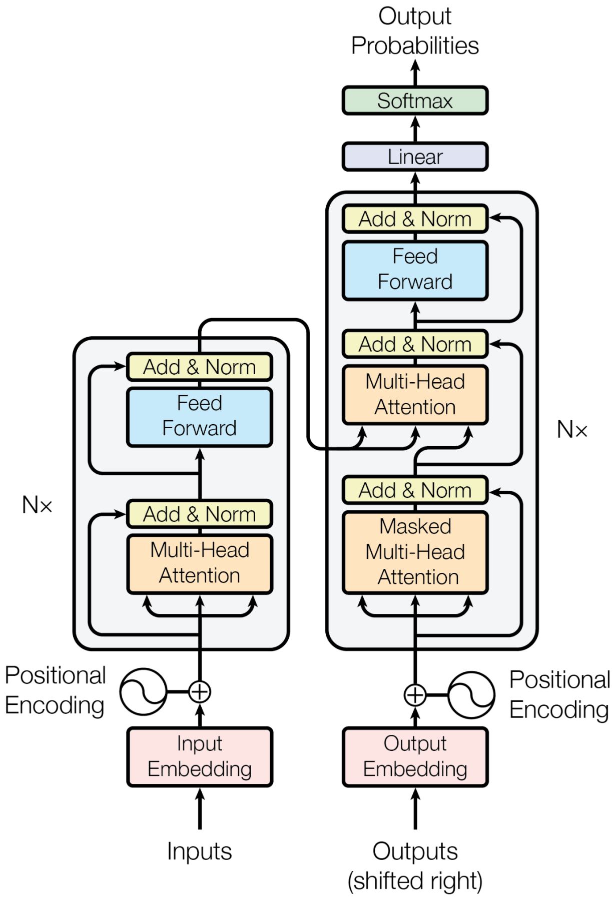


Figure 2.8: The Transformer encoder–decoder architecture (self-attention, masked self-attention, encoder–decoder attention, Add & Norm, Feed Forward). Reproduced from Vaswani *et al.* (2017) [31].

Figure 2.8 summarizes the original encoder–decoder Transformer. On the **left**, the *encoder* reads the entire source sequence in parallel. After token and positional embeddings, each of its N identical blocks applies *bidirectional* self–attention

$$Q = K = V = H^{(l-1)}, \qquad A = \operatorname{softmax}\Big(\frac{QK^\top}{\sqrt{d_k}}\Big), \qquad H^{(l)} = \operatorname{FFN}(AV),$$

so every position can attend to *all* others (no mask). Stacking blocks yields contextual representations

$$H_{\text{enc}} = \operatorname{Encoder}(x_1, \ldots, x_n) \in \mathbb{R}^{n \times d}.$$

On the **right**, the *decoder* generates the target sequence autoregressively. Each decoder block has three sublayers, as shown in Fig. 2.8: (i) *masked* self–attention, enforcing causality via a mask M,

$$A_{\text{mask}} = \operatorname{softmax}\Big(\tfrac{QK^\top + M}{\sqrt{d_k}}\Big), \quad M_{ij} = \begin{cases} 0 & j \le i \\ -\infty & j > i \end{cases};$$

(ii) *encoder–decoder (cross) attention*, which lets the decoder query the encoder outputs,

$$Q = H_{\text{dec}}, \quad K = V = H_{\text{enc}}, \quad A_x = \operatorname{softmax}\Big(\tfrac{QK^\top}{\sqrt{d_k}}\Big), \quad Z = A_x V;$$

(iii) a position–wise feed–forward network. Residual connections and LayerNorm wrap each sublayer (Add & Norm boxes in the figure). The top decoder state is mapped to next–token probabilities with a linear layer and softmax:

$$p_\theta(y_t \mid y_{<t}, x) = \operatorname{softmax}(W\, h^{(L)}_{\text{dec},t} + b).$$

Why the asymmetry? The encoder's role is **understanding** the whole input. The decoder's role is **generation**: at time t it can only use past targets $y_{<t}$ (mask) while consulting the source through cross–attention, as shown in Fig. 2.8.

Training objective (seq2seq). We maximize the conditional likelihood of each target sequence given the source. Equivalently, we minimize

$$\mathcal{L}(\theta) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T_i} \log p_\theta(y_{i,t} \mid y_{i,<t}, x_i),$$

where $p_\theta(\cdot)$ is the decoder softmax in Fig. 2.8 and $y_{i,<t}$ is the shifted target prefix. Seq2seq stands for the family of models that maps an input sequence into an output sequence (e.g. translations, summarization, subtitling).

2.5.6 Computational cost and long documents

Vanilla self–attention compares all token pairs, using time and memory $\mathcal{O}(n^2)$ per layer where \ is the sequence length. This could become a bottleneck for long texts (in our case, KIIDs and policies). Efficient variants reduce the cost by (i) restricting attention to a *sliding window* over nearby tokens (local context) and (ii) adding a small set of *global* tokens (e.g., headings or summary markers) that can summarize what the mask is hiding. Other approaches use sparse patterns or low–rank approximations. These designs preserve global signals while bringing compute and memory close to linear in n, making lengths in the thousands practical and training faster.

2.5.7 Why this helps our problem

Self–attention can connect distant cues, for instance, an asset-class mention and a constraint or percentage many sentences later, without having to shuttle information through many recurrent steps. Full token-level parallelism shortens training iterations and makes fine-tuning on domain documents feasible. In our pipeline, we therefore adopt encoder Transformers for standard inputs, switching to long–sequence variants when the policy extraction make them longer and them can exceeds the usual context window (details in §2.7–§2.8).

2.6 Pre-trained Language Models

Pre-trained language models (PLMs) are large Transformer encoders or decoders trained on massive, generic text to acquire broad linguistic competence before being *adapted* to a specific task with comparatively little supervision, meaning that with only the training of the model weights done using our specific texts it would not be possible to learn the *language* and the shades of it. In practice, this replaces hand-crafted feature engineering with *transfer learning*: first learn general regularities of language (syntax, semantics, long-range dependencies) during *pre-training*, then specialize the model to the downstream objective by attaching a small task head (or updating all weights) on labeled data.

2.6.1 Tokenization

PLMs do not work on raw characters or whole words; they use a *subword* vocabulary V (typically $|V| \approx$ 30k–50k) made of frequent character sequences, they do not necessarily need to be a whole word but it can happen. A tokenizer maps a string x to a sequence of subwords $(s_1, \dots, s_m)$ with $s_i \in V$ whose concatenation reconstructs x, i.e. $x \approx \text{concat}(s_1, \dots, s_m)$. This gives two key benefits: (i) *open vocabulary*-any new word can be spelled from known pieces; (ii) a compact lexicon that keeps embeddings manageable.

How the vocabulary is learned. Two mainstream families:

- **BPE (byte-pair encoding).** Start from characters; repeatedly merge the most frequent adjacent pair to form a new token until reaching size $|V|$. The tokenization of a word is the sequence of the longest merges present in V, to have less token for representing a single word.
- **WordPiece / Unigram.** Choose the token split that maximizes a simple language model over subwords (roughly, the product of token probabilities). Compared to greedy BPE, this can prefer slightly different segmentations when they are more probable under the learned token distribution.

Why this matters for finance. Domain terms and identifiers are often rare or out-of-vocabulary for word-level models. Subwords avoid collapsing them to `[UNK]` and preserve useful structure:

- Acronyms and regulatory terms: `UCITS` → `U ##C ##ITS` (WordPiece-style) or `UC ITS` (BPE); `KIID` stays intact if frequent, or splits into meaningful pieces.

- Tickers/indices: `MSCI-Europe` → `MSCI - Europe`; the model learns that `MSCI` is an entity and `Europe` is a region.

- Numbers and punctuation: `70%` → `70 %`, enabling heads to specialize on numeric cues while keeping the symbol.

What the model actually sees. After tokenization, each subword is mapped to an embedding and combined with a position signal. Special tokens (e.g., `[CLS]`, `[SEP]`) are injected for classification or pairwise inputs. Formally, the encoder processes

$$X = [e_{\mathrm{CLS}} + p_0,\ e_{s_1} + p_1,\ \ldots,\ e_{s_m} + p_m],$$

so even when a domain term is split, its meaning can be composed by attention over its subparts, thanks also to the positional encoding that we are keeping during the process.

Trade-offs. Larger vocabularies yield shorter sequences (fewer subwords per word) but bigger embedding tables; smaller vocabularies do the opposite. In our setting we favor standard vocabularies (e.g., BERT/RoBERTa) because they already encode many finance-adjacent patterns while remaining fully open-vocabulary for rare tickers and policy jargon. There also exists more specialized models (e.g., FinBERT) but those take a loss in generality that is important for future updates of policies and new issuers that could come in.

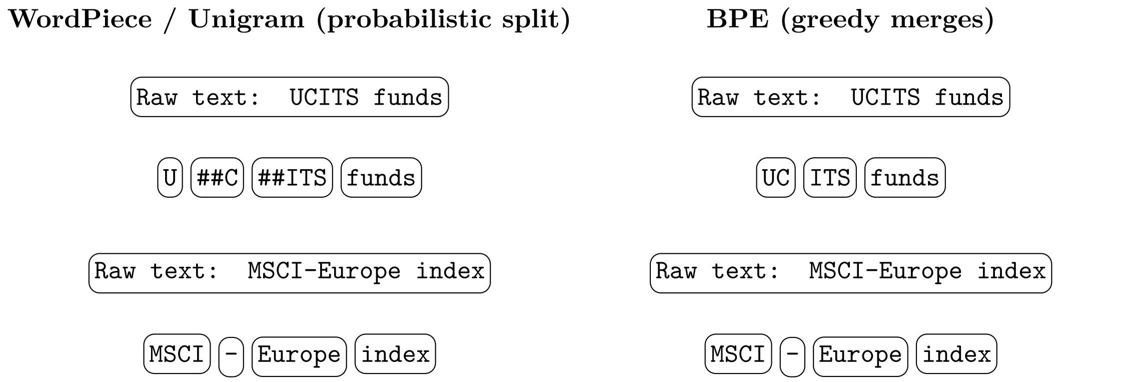


Figure 2.9: Two subword tokenization families used by PLMs. **Left (WordPiece/Unigram):** a probabilistic splitter chooses the segmentation that maximizes the subword language model (often with continuation marker, e.g., `U ##C ##ITS`). **Right (BPE):** greedy pair merges build the vocabulary; words are segmented by longest matches (e.g., `UC ITS`). Both approaches produce subword tokens that avoid `[UNK]` and keep the embedding table compact.

2.6.2 Pre-training objectives

Let $x = (x_1, \ldots, x_n)$ be a tokenized sentence and h_t the contextual hidden state for position t.

- **Masked Language Modeling (MLM).** A random subset $M \subset \{1, \ldots, n\}$ is masked; the model predicts the original tokens. The loss is

$$\mathcal{L}_{\mathrm{MLM}}(\theta) \;=\; -\,\mathbb{E}\Big[\sum_{t \in M} \log p_\theta(x_t \mid x_{\setminus M})\Big],$$

  encouraging bidirectional context use (left *and* right). This is the core of BERT-style encoders, with variations in masking policy and data/optimization in RoBERTa.

- **Causal Language Modeling (CLM).** A decoder-only model predicts the next token given the past:

$$\mathcal{L}_{\mathrm{CLM}}(\theta) \;=\; -\,\mathbb{E}\Big[\sum_{t=1}^{n} \log p_\theta(x_t \mid x_{<t})\Big],$$

  the standard objective behind GPT-style generators.

- **Denoising seq2seq objectives.** An encoder–decoder is trained to reconstruct clean text from a noised version (token/Span masking, shuffling, etc.), yielding a general-purpose text-to-text model. BART and T5 popularize this setup for robust generation and comprehension.

2.6.3 Fine-tuning for classification

For an encoder-only PLM, we obtain contextual states $H_{\mathrm{enc}} \in \mathbb{R}^{n \times d}$ and pool them into a sequence vector (e.g., the [CLS] state or mean pooling), then apply a linear head:

$$\hat{y} = \mathrm{softmax}\big(W_{\mathrm{clf}}\,\mathrm{Pool}(H_{\mathrm{enc}}) + b\big), \qquad \mathcal{L}_{\mathrm{CE}} = -\log \hat{y}_{\,y}.$$

Because the encoder was pre-trained on random texts, fine-tuning typically converges quickly and it needs less labeled data than training from scratch. In our domain, this allows the model to link dispersed cues (asset class, percentages, geographies) even when training sets are modest; that is, in some cases, what we are looking at.

2.6.4 Why pre-train?

Pre-training endows the model with syntax/semantics priors and robust optimization (good initial weights instead of random weights used to initialize each node, that is the usual process before training cycle). We then *adapt* it: (i) by task fine-tuning (as above), and, when helpful, (ii) by *domain-adaptive pre-training* (continued MLM on in-domain unlabeled KIIDs/prospectuses) so that finance-specific terms and style shift the encoder's representations toward our target distribution, that could be useful but not in any situation. In this way the backbone already "knows language", so the head can focus on decision boundaries rather than rediscovering basic patterns that are common in written text of any kind.

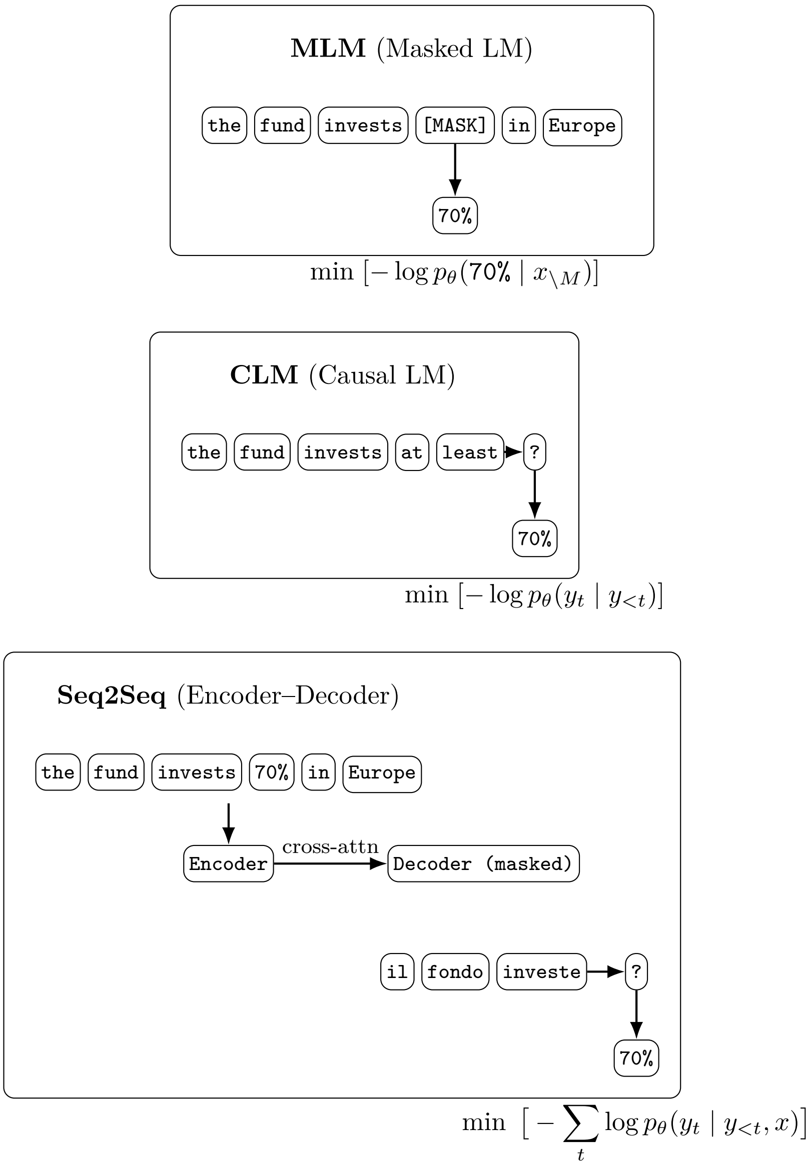

MLM (Masked LM)
the
fund
invests
[MASK]
in
Europe
70%
$\min\,[-\log p_\theta(70\% \mid x_{\setminus M})]$
CLM (Causal LM)
the
fund
invests
at
least
?
70%
$\min\,[-\log p_\theta(y_t \mid y_{<t})]$
Seq2Seq (Encoder–Decoder)
the
fund
invests
70%
in
Europe
Encoder
cross-attn
Decoder (masked)
il
fondo
investe
?
70%
$\min\,\big[-\sum_t \log p_\theta(y_t \mid y_{<t}, x)\big]$

Figure 2.10: Three learning paradigms. **MLM**: an encoder masks a subset of tokens and predicts them using *bidirectional* context. **CLM**: a decoder predicts the next token from the prefix only (causal mask). **Seq2Seq**: an encoder produce the hidden states then a decoder generate target *left-to-right* and at each step predicts the next token using its pasts and info from encoder via cross-attention.

2.6.5 Summary of the families

- **BERT/RoBERTa (encoder-only, MLM).** Strong for classification and retrieval, namely the processing of data to identify the pieces that we are looking for; RoBERTa refines BERT with more data, longer training, and removed next sentence prediction (NSP), showing equal or better downstream performance without it.

- **GPT (decoder-only, CLM).** Autoregressive generators; great at open-ended text but less typical for pure classification unless adapted.
- **BART/T5 (encoder–decoder, denoising).** Versatile for seq2seq tasks (summarization, rewriting) and can be cast as text-to-text for classification.

In the specific case of long document Vanilla PLMs have fixed context windows; for long KIIDs this motivates long-sequence variants (e.g., Longformer) or chunking/pooling strategies. We discuss these choices in §2.8.

2.6.6 Takeaway

Pretrained language models provide a strong, reusable *language prior* (via MLM/CLM/-denoising) that we specialize with a small supervised head. In our case, the labeled corpus contains ∼10,000 fund policies averaging ∼1,000 characters each that translate in ≈10 MB total, data obviously not enough for training a big LLM. By contrast, RoBERTa is pretrained on five large, general-domain corpora: BookCorpus, English Wikipedia, CC-News, OpenWebText, and Stories; totaling >160 GB of text, and uses an MLM-only objective with dynamic masking and longer training. This *orders-of-magnitude* gap in both *scale* and *domain* is exactly where transfer helps: initializing from RoBERTa reduces label needs, improves robustness to financial jargon and rare entities, and supplies the right inductive bias to capture long-range constraints and dispersed mentions within policy texts. Residual class imbalance remains a challenge at our smaller scale; we therefore investigate textual data augmentation to amplify minority classes without altering the encoder architecture.

2.7 RoBERTa-base

RoBERTa (*Robustly Optimized BERT Pretraining Approach*) is a Transformer encoder trained with a masked language modeling (MLM) objective, but with several changes that make pretraining more effective than the original BERT [7, 16]. In particular, RoBERTa removes the Next Sentence Prediction (NSP) loss, uses *dynamic* masking, adopts larger batches and longer training, and relies on a byte-level BPE tokenizer (GPT-2 vocabulary, 50k subwords). Pretraining uses a large and diverse English corpus (CC-News, OpenWebText, Stories, Wikipedia, BookCorpus), totaling roughly 10^{10}–10^{11} tokens.

2.7.1 What "robustly optimized" means in RoBERTa

The phrase in the RoBERTa name denotes a *carefully engineered pretraining recipe* for the BERT encoder that makes the training procedure less brittle and more effective across tasks [16]. The robustness is derived from: (i) an accurate selection of objectives, (ii) a dynamic masking set to avoid overfitting, (iii) a pretraining on a much larger and diverse corpus (reaching $\sim 10^{10}$–10^{11} tokens), (iv) the use of BPE (GPT-2 vocabulary) to avoid out-of-vocabulary tokens [23] and (v) a longer training and with larger *batch size* (the number of training samples used in one iteration of model training) and tuned schedules (AdamW, warmup, cosine or linear decay). In summary, "robustly optimized" is an empirical claim: with these choices, pretraining becomes less sensitive to specific hyperparameters and yields consistently better downstream results than the original BERT. It does not have to be confused with the *formal* robust optimization in *optimization theory*, the branch using worst-case minimization or ambiguity-set risk.

2.7.2 Encoder block

The network remains a BERT-style encoder: same API, same 12/24-layer stacks, same hidden sizes. This means we can reuse infrastructure and heads with minimal code changes and the encoder design is made up of: *post-norm* residual blocks with multi-head self-attention, a two-layer position-wise feed-forward network with GELU, and *absolute* learned positional embeddings. For a layer ℓ, with hidden states $H^{(\ell)} \in \mathbb{R}^{T \times d}$:

$$\begin{aligned}
\mathrm{SA}(H) &= \mathrm{softmax}\Big(\tfrac{QK^\top}{\sqrt{d_k}}\Big)V, \\
\tilde{H}^{(\ell)} &= \mathrm{LN}\big(H^{(\ell)} + \mathrm{Dropout}(\mathrm{SA}(H^{(\ell)}))\big), \\
H^{(\ell+1)} &= \mathrm{LN}\big(\tilde{H}^{(\ell)} + \mathrm{Dropout}(\mathrm{FFN}(\tilde{H}^{(\ell)}))\big),
\end{aligned}$$

where $\mathrm{FFN}(x) = W_2\ \mathrm{GELU}(W_1 x + b_1) + b_2$.
W_1 is a linear map that projects up to a much larger hidden layer; this is useful to express more complex non-linear transformations, then a non linearity is applied and the second linear map W_2 restores the original dimensionality so that residual connections can be applied. This “expand-shrink” pattern is common in Transformer’s design choice.

- **Post-norm vs. pre-norm.** RoBERTa follows the *post-norm* layout (LayerNorm applied *after* the residual addition), as shown in the equations above. This matches the original BERT recipe, by contrast, *pre-norm* variants (LayerNorm before each sublayer) are often used to stabilize *very* deep Transformers but are not part of RoBERTa’s standard checkpoints.
  Residual connections (skip connections). Given a input x and a non-linear sublevel $F(\cdot)$ (e.g. self-attention or FFN) a residual connetion sum the output of the sublevel with the original input: $y = x + \mathrm{Dropout}\big(F(x)\big)$, in Transformer we also apply a normalization. While in the *post-norm* variant (like in BERT/RoBERTa) the block is: $\tilde{x} = \mathrm{LN}\big(x + \mathrm{Dropout}(F(x))\big)$.
  We can notice two effect: (i) mitigation of vanishing/exploding gradient, facilitating the gradient flow and (ii) permit to implement the identity map when $F(x) \approx 0$, “dropping” layers that actually hurt the training accuracy, leaving signal untouched.

- **Absolute positional embeddings.** Positions are learned as vectors p_t and added once at the input: $x_t = e_{w_t} + p_t$. Effective choise but it stops a the maximum training lenght (that is 512 tokens), so for longer contexts we switch to long-sequence encoders (cf. §2.8).

- **Dropout placements.** Two dropouts are used: (i) *attention dropout* on the attention weights and (ii) *residual dropout* on the outputs of SA/FFN before the residual addition; defaults are typically 0.1 during fine-tuning.

Dropout. A dropout layer randomly sets a fraction p of weights to zero during model training (and rescaling the others by $\frac{1}{1-p}$). This acts as stochastic regularization: it reduces overfitting and improves generalization, especially when the labeled dataset is modest; this because it tends to don't focus on specific nodes that are gaining weight and this could start a chain reaction that could bring the model at not spreading learning across many nodes. At inference time, during testing or during real-world application, dropout is disabled (no activations are zeroed).

- **Masks.** Padding tokens are excluded from attention with a padding mask. Causal masks are *not* used in the encoder (bidirectional context), but appear in decoder-only models.
  Padding tokens. They are special placeholders inserted to make all sequences in a mini–batch the same length (e.g., right–padding `the fund invests` → `the fund invests [PAD] [PAD]`). They carry no semantic meaning and *must not* influence attention or the loss.
  Padding mask. Let $v \in \{0,1\}^T$ mark real tokens (v_t=1) vs. padding (v_t=0). Define an additive mask $M \in \mathbb{R}^{T\times T}$ with $M_{ij} = 0$ if v_j=1 and $M_{ij} = -\infty$ if v_j=0. The attention weights are then computed as

$$A = \mathrm{softmax}\Big(\frac{QK^\top}{\sqrt{d_k}} + M\Big),$$

  which forces $A_{i,j} = 0$ whenever position j is padding.

- **FFN sizing and activation.** The position-wise FFN expands the channel dimension from d to $d_{\mathrm{ff}} \approx 4d$ (e.g., 768 → 3072 in `roberta-base`) with a GELU nonlinearity, then projects back to d. GELU combines the benefits of ReLU-like sparsity with smoother gradients, see Fig. 2.12 for its shape.

The encoder layer can be read as: (i) globally reweight information via self-attention, (ii) stabilize with residuals+LayerNorm, and (iii) refine each position independently with a high-capacity FFN. RoBERTa keeps this BERT-compatible block unchanged, differing primarily in its pretraining and tokenizer.

2.7.3 When to prefer RoBERTa

In practice, RoBERTa is not always the default choice; it is preferable under certain conditions that match its design strengths:

- **Medium-length documents (≤ 512 tokens).** RoBERTa processes sequences up to its native *context window* without architectural changes. This makes it well suited for most fund policy snippets, where the extracted text rarely exceeds a few lines. Longer inputs would instead require chunking strategies or long-sequence models such as Longformer.
  Context windows. This is a crucial aspect of Trasformer encoders, that is the maximum number of tokens that the model can process together in a single iteration. For RoBERTa this limit is set to 512 tokens (inherited from BERT), so inputs longer then this needs to be truncated or split and overlapped loosing the entire context written out of the window.

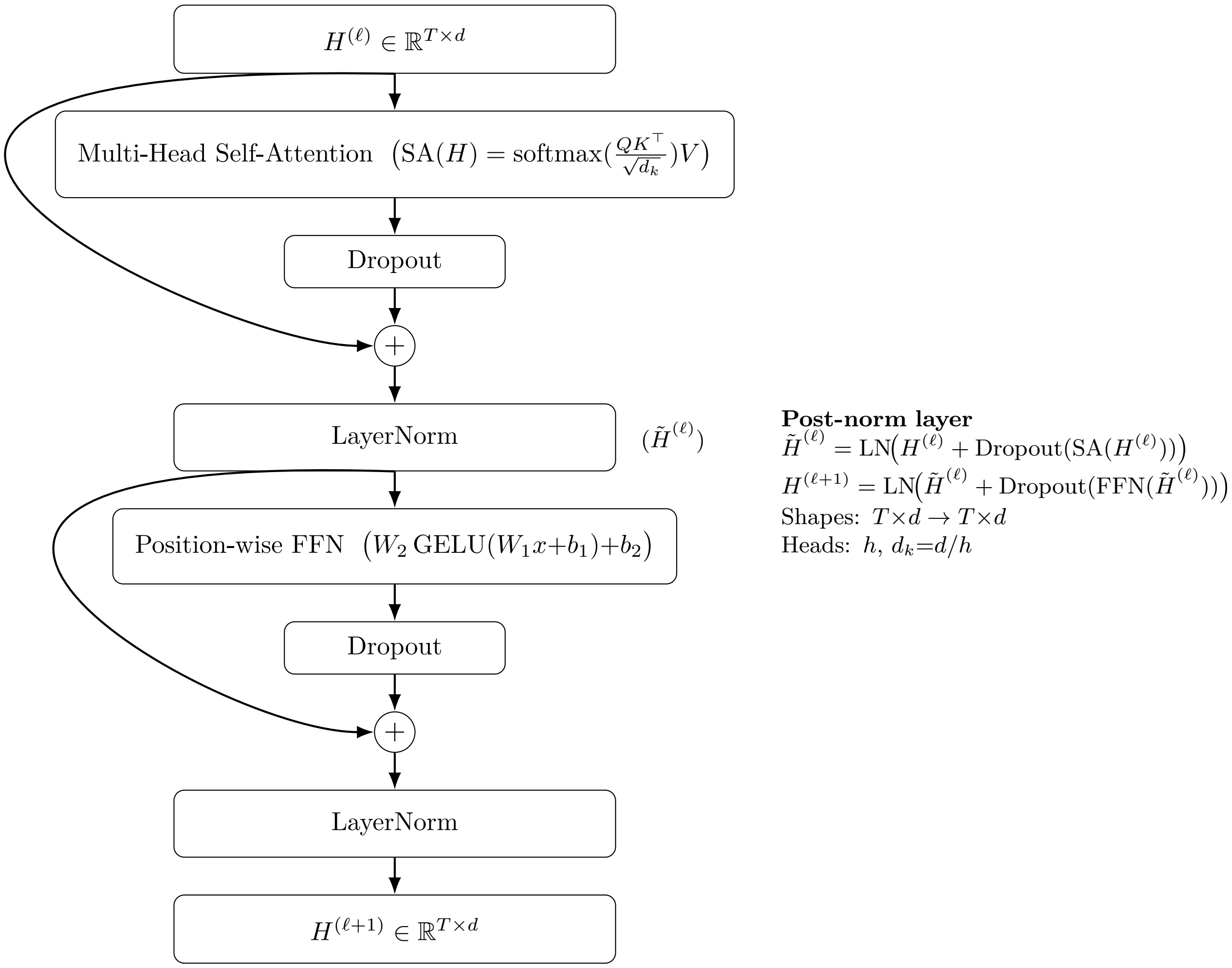

$H^{(\ell)} \in \mathbb{R}^{T\times d}$
Multi-Head Self-Attention $\big(\mathrm{SA}(H) = \mathrm{softmax}(\frac{QK^\top}{\sqrt{d_k}})V\big)$
Dropout
+
LayerNorm
$(\tilde{H}^{(\ell)})$
Position-wise FFN $\big(W_2\,\mathrm{GELU}(W_1x{+}b_1){+}b_2\big)$
Dropout
+
LayerNorm
$H^{(\ell+1)} \in \mathbb{R}^{T\times d}$
Post-norm layer
$\tilde{H}^{(\ell)} = \mathrm{LN}\big(H^{(\ell)} + \mathrm{Dropout}(\mathrm{SA}(H^{(\ell)}))\big)$
$H^{(\ell+1)} = \mathrm{LN}\big(\tilde{H}^{(\ell)} + \mathrm{Dropout}(\mathrm{FFN}(\tilde{H}^{(\ell)}))\big)$
Shapes: $T{\times}d \to T{\times}d$
Heads: h, $d_k{=}d/h$

Figure 2.11: Encoder block schematic: Self-Attention → Dropout → Residual + LayerNorm → FFN → Dropout → Residual + LayerNorm, come nelle tre equazioni del testo.

- **Reliable transfer with modest fine-tuning data.** Because our dataset is relatively small compared to the pretraining corpora, models that transfer stably with limited supervision are crucial. RoBERTa's strong initialization ensures that even when fine-tuned on ∼10k labeled policies, it learns domain-specific distinctions without overfitting or collapse.

- **Handling of domain tokens and numbers.** Byte-level BPE avoids out-of-vocabulary issues for financial acronyms, tickers, or numeric thresholds. This is particularly advantageous in our domain, where rare entities (e.g., "UCITS", "MSCI-Europe", "70%") carry direct class-defining meaning. Alternative models with fixed word-level vocabularies would often fall back to `[UNK]`, losing critical information.

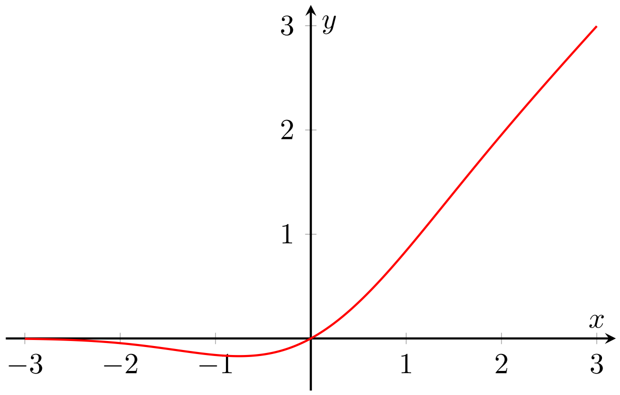


Figure 2.12: **Gaussian Eror Linear Unit** GELU(x) [9]. Smooth nonlinearity used in BERT/RoBERTa. Where the exact function si derived from the standard Gaussian cumulative distribution function: $\mathrm{GELU}(x) \doteq x\,\Phi(x) = x \cdot \frac{1}{2}\left(1+\mathrm{erf}\left(\frac{x}{\sqrt{2}}\right)\right)$, where $\Phi(x) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x} e^{t^2/2}\,dt$. It can be approximated with: $\mathrm{GELU}(x) \approx \frac{x}{2}\left(1+\tanh\left(\sqrt{2/\pi}\,(x+0.044715\,x^3)\right)\right)$, which smoothly suppresses negative inputs and keeps large positives near the identity.

- **Computational efficiency.** RoBERTa offers a good balance between model capacity and inference speed. Compared to more specialized long-sequence models, it requires less memory and simpler infrastructure, which makes large-scale training runs and deployment pipelines more tractable.
- **Limitations.** RoBERTa is less suitable when documents consistently exceed the 512-token limit (favoring Longformer), when parameter-sharing or compression is necessary (favoring ALBERT), or when multilingual coverage is required (favoring XLM-R). These constraints help delineate the boundaries of RoBERTa's usefulness in practice.

2.7.4 Summary

RoBERTa stands out not because it introduces a new architecture, but because its pretraining recipe yields an encoder that is stable, data-efficient, and domain-flexible. For our application, where documents are medium-length, labels are limited, and tokenization of domain-specific strings is critical, RoBERTa provides the most practical backbone among general-purpose PLMs. Its trade-offs are clear: when the task shifts to longer inputs or different languages, other architectures may be more appropriate, but within the scope of this thesis RoBERTa represents the most balanced and effective choice.

2.8 Longformer-base

Many policy texts are longer than the 512–token window used by standard encoders. With a short window we must either truncate the input (and lose information) or split it into pieces and then merge the pieces prediction together. Both choices are brittle for KIIDs, where key cues may sit far from the noun they qualify. Longformer [1] addresses this by changing how self–attention is computed so that long documents can be processed in a single pass (see Fig. 2.13).

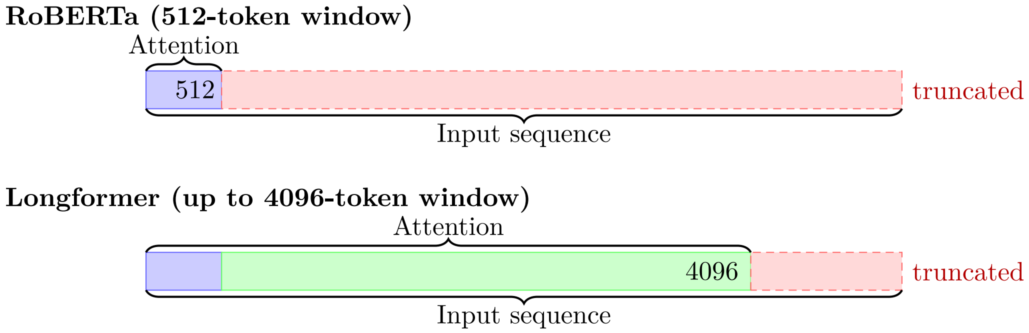
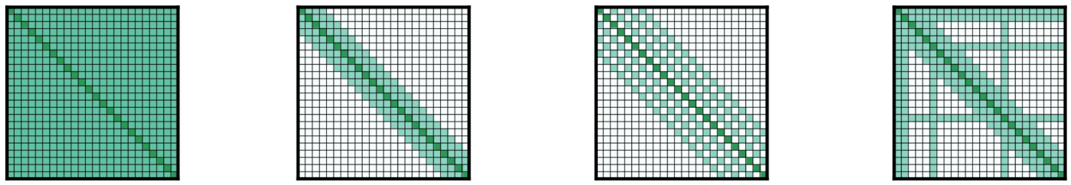


Figure 2.13: Illustration of the *context window*. RoBERTa can only process up to 512 tokens at once; any excess text must be truncated or chunked. Long-sequence models like Longformer extend the window to thousands of tokens, enabling full-document modeling.

2.8.1 Attention pattern

Instead of attending to *all* tokens ($O(n^2)$ cost), Longformer uses:

$$\underbrace{\text{sliding–window local attention}}_{O(n\,w)} + \underbrace{\text{a few \textit{global} tokens}}_{O(n\,g)}$$



(a) Full n^2 attention (b) Sliding window attention (c) Dilated sliding window (d) Global+sliding window

Figure 2.14: Representation of different attention patterns for long sequences. (d) Attention actually used in Longformer: sliding-window plus a few *global* tokens. Reproduced from Beltagy *et al.* (2020) [1].

Here n is the sequence length, w the local window size (a few hundred tokens), and g the number of positions marked as *global*. Each token attends to its neighborhood (the sliding window), which captures short- and mid-range dependencies efficiently. A small set of global tokens can attend to (and be attended by) *all* positions creating document–wide information flow.

How to chose *global* tokens. In Longformer the set of global positions is *not learned by the model*; it is provided as a binary mask together with the input [1]. The choice is task–dependent and kept small to control cost.

Typical rules. For document classification we always mark the special classification token (`[CLS]`) as global so it can aggregate evidence from the whole sequence, this is the most important token when the output needed is classification. For QA we mark the question tokens; for sequence labeling we may mark sentence starts or headings.

Budgeting. We cap the number of global tokens to $G_{\max}$ (e.g., 32 or 64). If candidates exceed the cap, the proposed rule is to keep left–to–right the first occurrences and drop the rest; alternatively, one can rank by TF-IDF over the training corpus and keep the top $G_{\max}$.

2.8.2 Why this helps on KIIDs

Policy documents mix local patterns (e.g., instrument–modifier pairs like "government bonds", "emerging markets equities") with constraints and exceptions that may appear many sentences later. The local window covers the nearby phrasing that drives classification, while global tokens collect and redistribute signals that are spread across the file (percentages, caps, geography, style). In practice we assign global attention at least to the special classification token (e.g., `[CLS]`), so it can aggregate evidence from the whole document before the final prediction.
Longformer is preferred in this specific application, over RoBERTa or similar encoder, because of text length above 512 tokens in a non-negligible fraction of analyzed inputs, case in which a standard encoder is not sufficient to "understand" the whole context. With Longformer we can feed the full text (up to a several thousand tokens) rather then truncating or chunking and so losing long-range cues that affect the class.

2.9 Summary

By combining cheap local attention with a tiny number of global positions, Longformer scales linearly in sequence length, preserves the structure of long KIIDs, and lets the classifier exploit document-wide context without giving up efficiency.

Chapter 3

Theory of *Textual Data Augmentation*

3.1 Motivation of augmenting textual data

Textual data augmentation (DA) creates additional training examples that *preserve the original label* while injecting controlled diversity using different methodology. This helps with (i) general data scarcity, (ii) imbalanced classes and (iii) overfitting to superficial patterns, thereby improving generalization and robustness to distribution shift.

The basic data augmentation theory for NLP (natural language processing) divides the macro categories into three broad families:

- **Paraphrasing-based methods** → Generate new text that keep the original meaning.
  - *How*: (i) synonym or embedding-neighbour substitutions (e.g., WordNet, nearest neighbours in embedding space); (ii) machine-translation paraphrases such as back-translation ($L_0 \to L_p \to L_0$); (iii) seq2seq/LLM paraphrasing (using label or in general promps task-conditioned). [15, 27]
  - *When to use*: we stylistic and syntactic variety is needed beyond small token edits (e.g., reorder clauses, switch active/passive voice) while preserving finance-specific content, not alterating the meaning and the information contents inside the text.
  - *Pros*: high fluency and diversity; closer to real variability in KIID texts.
  - *Cons*: more expensive, for some applycation needed to be done not on a local machine; risk of semantic drift if not filtered.
- **Noising-based methods** → Apply small, local edits to the input, usually locally done.
  - *EDA (Easy Data Augmentation)*: synonym replacement (SR), random insertion (RI), random swap (RS), random deletion (RD). Use a small edit rate $\alpha \in [0.05{,}0.2]$ to balance diversity and label preservation. [32]

  - *AEDA (An Easier Data Augmentation)*: insert only punctuation at random positions; keeps word order and avoids content loss typical of RD/SR. Strong gains in small-data regimes with minimal cost. [11]
  - *STA (Selective Text Augmentation)*: protect high-value ("Gold") tokens; apply insertion/deletion selectively by lexical role to reduce label noise in low-resource settings. [8]
  - *Pros*: cheap, easy to implement, effective with few samples; complements heavier generators.
  - *Cons*: careless swaps/deletions can flip meaning; punctuations insertion can make augmented phrases difficult to read. [15]

- **Sampling / model-based generation** → Use models or feature-space interpolation to produce task-aware synthetic data.
  - *Model generation*: condition pretrained LMs (seq2seq/GPT) on the task or label to sample paraphrases. Typically more *learnable* and task-aware, with higher diversity. [15]
  - *MixUp (representation-space)*: interpolate examples and labels to create new instances, includes also shaded labels:

    $$\tilde{x} = \lambda x_i + (1-\lambda)x_j, \quad \tilde{y} = \lambda y_i + (1-\lambda)y_j, \quad \lambda \sim \mathrm{Beta}(\alpha, \alpha),$$

    applied to sentence embeddings or Transformer hidden states, not interpretable. Works *online* during training and helps minority classes. [30, 33]
  - *Pros*: label-aware, often higher-quality and more diverse than rule-based edits.
  - *Cons*: greater complexity/cost; document-level generation can be harder for current models, so needed to be combined with simpler methods when texts are long.

Empirically, text classification tends to benefit from all three, whereas sequence generation often prefers sampling-based methods, and structured prediction prefers paraphrasing with stricter validity constraints.

Design desiderata. Effective DA have to balance all this things at the same time: (i) *label preservation* (the avoidance of semantic drift); (ii) *diversity* (the introduction of new lexical/syntactic variants); (iii) *fluency* (readability that entails well-formed text also if examples don't necessarily need to be interpretable); and (iv) *task-knowing* (do not break domain-specific terms leading the model to misunderstood the context). Surveys [15] also highlight *filtering* as a practical necessity: discard low-quality samples with automatic controls to prevent error accumulation.

Caveat. Because language is discrete and sensitive to word order, naive edits (e.g., swapping tokens, deleting words) can flip the corpus entailing corrupted labels; even unnoticeable small changes (e.g., *"This is good"* → *"Is this good"*) alter meaning. This

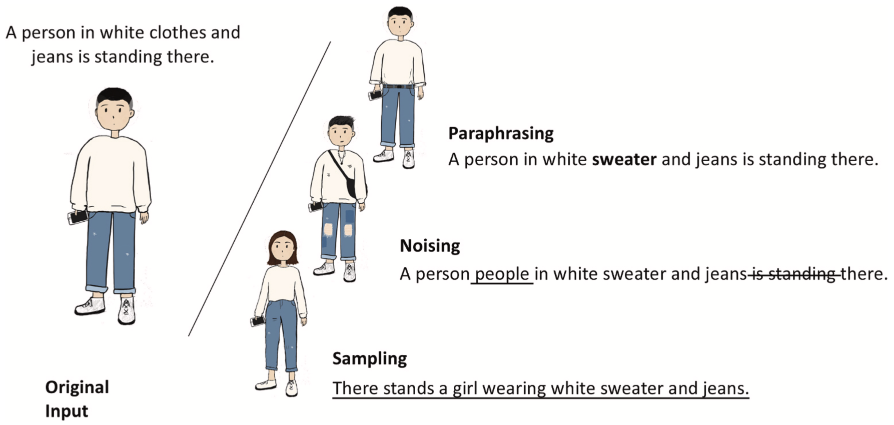

A person in white clothes and jeans is standing there.
Original Input
Paraphrasing
A person in white **sweater** and jeans is standing there.
Noising
A person people in white sweater and jeans ~~is standing~~ there.
Sampling
There stands a girl wearing white sweater and jeans.

Figure 3.1: Representation of different attention patterns for long sequences. (d) Attention actually used in Longformer: sliding-window plus a few *global* tokens. Reproduced from Bohan Li *et al.* (2022) [15].

is why some DA methods include explicit protections or quality gates that needs to be overtaken.

In following parts, we expand each technique we used in our pipeline, connecting practical choices to the literature and noting possible trade-offs in between them.

3.2 AEDA: inserting punctuation only

AEDA (*An Easier DA*) avoids semantic operations and inserts only punctuation marks at random positions. This preserves the order and identity of words (thus the core meaning), while lightly perturbing surface form and giving more strength teaching the model to be unaware to the text composition and instead look at the inside meaning [11].

Procedure used. Let s be the number of tokens in the sentence. We draw

$$q \sim \mathcal{U}[1{,}0.3 \times s], \qquad \text{positions} = \text{sample}(s, q)$$

where s are the world inside a phrase and q is the number of insertion that we want to do. The algorithm insert q symbols sampled uniformly from $PUNCTUATIONS = \{\texttt{.}\,,\texttt{!}\,,\texttt{?}\,,\texttt{;}\,,\texttt{:}\}$ in uniform positions, preserving the word order and so the lexical information. The choice of the number that construct the uniform distribution comes from the need to insert the right number of marks, not too much, that could create negative effect on the model, and at least one insertion; this mirrors the reference implementation and hyperparameters recommended by the authors [11].

When it helps. AEDA consistently improves small-data regimes and avoids the label errors sometimes introduced by deletions/substitutions in EDA (see next section §3.3. Reported gains are largest with few hundred examples and remain positive even with larger sets.

3.3 EDA: controlled lexical perturbations

EDA (*Easy Data Augmentation*) applies simple, stochastic edits at the word level: synonym replacement (SR), random insertion (RI), random swap (RS), and random deletion (RD). Each operation changes roughly an α fraction of tokens ($\alpha \in [0.05{,}0.2]$ is typical), setting a trade between diversity and potential meaning drift [32].

Methods explaination.

- **Synonym replacement (SR)** → choose $n = \max(1, \alpha s)$ non-stopwords, that means words that carry semantic meaning and are not high-frequency ones and replace each with a close synonym (from WordNet), preserving the original meaning as much as possible.

- **Random insertion (RI)** → n times, pick a random non-stopword w and insert a synonym of w at a random position in the sentence; content is kept while adding lexical variation.

- **Random swap (RS)** → perform n random swaps of two token positions; this perturbs word order without changing the bag-of-words content, hardly this could "damage" the inner information.

- **Random deletion (RD)** → delete each token independently with probability $p = \alpha$; use small p (e.g., $\alpha \leq 0.1$) or protect key terms (entities, domain keywords) to avoid semantic drift, this could become problematic if wrong words are deleted.

Operation	**Sentence**
None	A sad, superior human comedy played out on the back roads of life.
SR	A *lamentable*, superior human comedy played out on the *backward* road of life.
RI	A sad, superior human comedy played out on *funniness* the back roads of life.
RS	A sad, superior human comedy played out on *roads* back the of life.
RD	A sad, superior human out on the roads of life.

Table 3.1: Sentences generated using EDA. SR: synonym replacement. RI: random insertion. RS: random swap. RD: random deletion. Reproduced from Wei *et.al* (2019) [32].

Takeaway. EDA is an economic (low hardware resource consumption) DA methodology, it could be used locally without any problem and is applicable to dataset of any dimension. Is preferred the use of low α and insertion/swap over deletion; pair with filtering (Section **??**).

3.4 Back-translation

Translate s from the source language L_0 to a pivot L_p and back to L_0 to obtain a paraphrase $\hat{s}$ that preserves semantics but varies syntax and phrasing due to the different grammatical laws that belong to different linguistic strains. This is a classic and effective DA for NLP tasks that should keep intact fluency and style [29].

Our setting. We generate $\hat{s} = \mathrm{BT}_{L_0 \to L_p \to L_0}(s)$ using Deepl with pivots $L_p \in \{\mathtt{AR}, \mathtt{EL}, \mathtt{RU}\}$ and persist a cache $\mathcal{C} = \{(s, L_p) \mapsto \hat{s}\}$ for cost control and reproducibility. Pivot selection is *typology-aware*: Semitic (AR), Hellenic (EL) and Slavic (RU) differ from English in script, morphology and word order, which empirically yields stronger rephrasing (more difference in between the input and the output after the double translation due to the greater language distance of those three compared with English) than closely related pivots.[1] We avoid multi-pivot chains at scale (e.g., $L_0 \to L_{p_1} \to L_{p_2} \to L_0$): they do not raise variety as expect, increasing costs and drift risk superlinearly; a curated set of single pivots with filtering gave a better cost/quality trade-off in our domain.

Takeaway. Back-translation was originally introduced to exploit monolingual data in NMT (neural machine translation) and is widely adopted as a strong, domain independent augmenter [27]. Quality depends, apart from the choice of the translation model itself, on pivot distance and decoding diversity; combining several pivots can increase variety but raises costs.

3.5 Semantic MixUp in embedding space

MixUp regularizes models by training on convex combinations of examples and labels:

$$\tilde{x} = \lambda x_i + (1 - \lambda)x_j, \qquad \tilde{y} = \lambda y_i + (1 - \lambda)y_j, \qquad \lambda \sim \mathrm{Beta}(\alpha, \alpha).$$

Originally proposed in vision, MixUp encourages local linearity and reduces overconfidence [33].

NLP variants. For text, MixUp is performed in embedding space either on sentence embeddings or inside Transformer hidden states at a randomly chosen layer followed by training with soft labels $\tilde{y}$ which in turn derive from the mixing of the labels of the mixed examples. These *MixUp-Transformer* approaches are especially beneficial in low-resource settings [30].

Our implementation. We apply MixUp to SBERT sentence vectors and to the `[CLS]` representation from layers $L \in \{10,11,12\}$ of a BERT-base encoder, sampling $\lambda \sim \mathcal{U}(0.3,0.8)$ and generating up to five synthetic points per input, using different λ.

[1] Greater typological distance generally increases paraphrase diversity but also the risk of mild semantic drift; to mitigate this we keep a *single* pivot per instance and pass candidates through our similarity/overlap filters during first testing phases

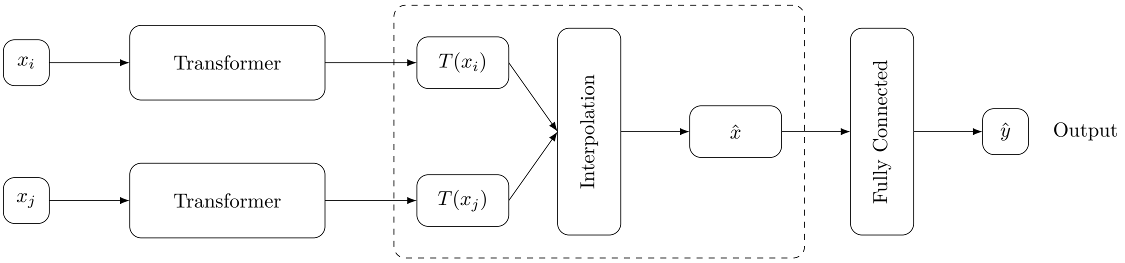


Figure 3.2: Representation of *MixUp-Transformer.* x_i and x_j are two different sentences respectively that are fed to the same Transformer. $T(x_i)$ and $T(x_j)$ are the input representation made from the Transformer. $\hat{x}$ and $\hat{y}$ are respectively the interpolated representation and label. Reproduced from Lichao Sun *et al.* (2020).

3.6 Role-aware augmentation (STA)

Selective Text Augmentation (STA) assigns each token two scores, its *semantic similarity to the label* and its *statistical correlation with the label*, then places words into four roles. Augmentations (selective replacement/insertion/deletion and *positive selection* of Gold words) are then applied with constraints that protect core semantics [8].

How the roles are computed. Given a text x with class c, STA computes:

- **Label–similarity** $ls(w, c)$: cosine similarity between the embedding of word w and a label prototype for c (e.g., the average of sentence embeddings from class-c documents). Higher means w is semantically close to c.

  Let $\mathbf{v}_w$ be the static word embedding of token w and $\mathbf{v}_c$ the embedding for class c (either the label token itself or the average of the tokens in its textual description). The semantic similarity is the cosine:

  $$ls(w, c) = \cos(\mathbf{v}_w, \mathbf{v}_c) = \frac{\mathbf{v}_w^\top \mathbf{v}_c}{\|\mathbf{v}_w\| \, \|\mathbf{v}_c\|}.$$

  In STA *Word2Vec*/GloVe is used to be computational efficient, not using Transformer model in this step.

- **Label–correlation** $lr(w, c)$: a class specific score indicating how selectively w appears in c versus other classes, a statistical data that depend on the frequency on which the word co-occurs with a specif class c_j while not with other classes.

  Statistical correlation is measure using *Weighted Log-Likelihood Ratio* (WLLR):

  $$lr(w, c) = \mathrm{WLLR}(w, c) = p(w \mid c) \, \log \frac{p(w \mid c)}{p(w \mid \bar{c})},$$

  where $p(w \mid c) = \frac{\mathrm{freq}(w,c)}{\sum_{w'} \mathrm{freq}(w',c)}$ and $p(w \mid \bar{c}) = \frac{\mathrm{freq}(w,\bar{c})}{\sum_{w'} \mathrm{freq}(w',\bar{c})}$ have been estimated using the occurrence frequencies of instances in the class c and any other class $\bar{c}$.

After having calculated scores for each word, threshold are set to divide the words in the four groups:

$$C_h = \{w : lr(w,c) \geq \tau_{lr}\}, \quad C_l = \{w : lr(w,c) < \tau_{lr}\},$$

$$S_h = \{w : ls(w,c) \geq \tau_{ls}\}, \quad S_l = \{w : ls(w,c) < \tau_{ls}\}.$$

The *word roles* are:

$$W_G = C_h \cap S_h, \qquad W_V = C_h \cap S_l, \qquad W_B = C_l \cap S_h, \qquad W_T = C_l \cap S_l,$$

i.e. **Gold**, **Venture**, **Bonus**, **Trivial**. [8]

Words are plotted in the 2D plane (ls, lr) and split into four quadrants/roles:

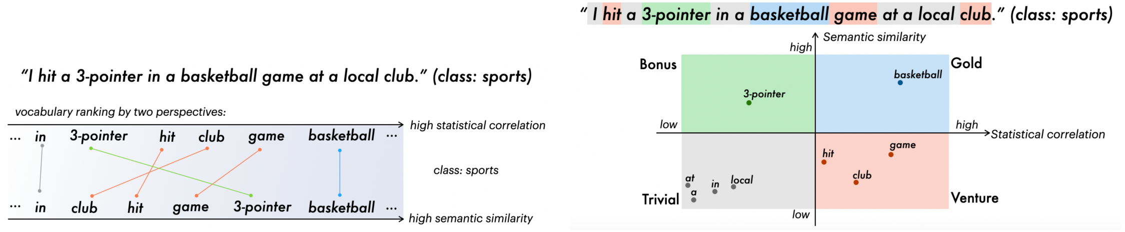
Gold: high ls and high lr (class-indicative core terms),

Bonus: high ls but low lr (semantically relevant but generic),

Venture: low ls but high lr (statistically specific, narrow semantics),

Trivial: low ls and low lr (function words/noise).

This role map guides which tokens to edit and which to protect when applying the already seen text augmentation technique. The map is shown in Fig. 3.3.


"I hit a 3-pointer in a basketball game at a local club." (class: sports)
vocabulary ranking by two perspectives:
... in 3-pointer hit club game basketball ...
high statistical correlation
class: sports
... in club hit game 3-pointer basketball ...
high semantic similarity
" I hit a 3-pointer in a basketball game at a local club." (class: sports)
Semantic similarity
high
Bonus
Gold
3-pointer
basketball
low
high
Statistical correlation
hit
game
club
at
a
in
local
Trivial
Venture
low

Vocabulary ranked by correlation vs. similarity.   Quadrants: Gold/Bonus/Venture/Trivial.

Figure 3.3: STA intuition on the sentence “I hit a 3-pointer in a basketball game at a local club". Showing how the word are classified and based on what each different technique is applied on different portion of text. Reproduced from Biyang Guo *et al.* (2022).

Selective operations. STA applies standard EDA-style edits but *role-conditioned*, so the technique used are the same but applied to everything:

- **Protect Gold** (no deletion, rarely replaced); optionally *positive selection* (copy-/retain/phrasal emphasis) so class cues are preserved.
- **Bonus words**: safe targets for *synonym replacement* or light *insertion* to add lexical variety without shifting meaning.
- **Venture words**: avoid *insertion* (can overfit to false markers); allow *replacement* to reduce dataset bias.
- **Trivial words**: allow *deletion/swap* to improve robustness to local noise.

Our domain. In investment-policy descriptions, candidate **Gold** terms include asset classes (*equity, bond, money market*), regions (*Europe Developed, Latin America*), styles (*value, growth*), and regulation-linked terminology (*UCITS, benchmark*). STA's positive selection keeps these anchors intact, so it's very useful in not losing the context while upgrading the number of samples in our minority classes; it edits peripheral word yielding to a diverse yet label-faithful variants that help these classes.

3.7 Paraphrasing with large language models (LLMs)

Rule-based noising (EDA/AEDA) mostly creates local perturbations; back-translation varies phrasing, but may be conservative. This could improve the robustness to little typographical error but they leave the macro-structure of the document intact. LLMs can restructure sentences (e.g. they can alternate the voice tone from passive to active or go from a single period to a bullet list), switch voice, and replace finance jargon with synonyms that match the outer context by teaching them about the external environment, better match cross-provider style variability found in real KIID texts. Conceptually, in this way, the model will focus more on the information contained in the text rather than on the editorial style.

Conditioned generation. A simple but effective recipe is to *prepend the class label or task instruction* to condition GPT models, then sample paraphrases. In low-resource classification, label-conditioned seq2seq models often outperform other DA baselines [12].

Our design. In our study we applied LLM-based augmentation through a label-conditioned, few-shot paraphrasing pipeline build around KIID investment policies. We use a specialized prompt to give the operational information to GPT-4o, and showing the model n policies we ask $n/2$ policies in output.

The input batch $(x_1, \dots, x_n)$ already share a target label y, we prepend y and a compact instruction block that frame the context and then ask the model to produce $(x_1{}^\star, \dots, x_{n/2}{}^\star)$ that (i) keep the label-defining facts intact, (ii) vary sentence structure and lexis, and (iii) mimic the cross-provider stylistic variability seen in real KIIDs.

Prompting & guardrails. To make generation reliable at scale, some request are needed to be filled, adding on-prompt guardrails with post-generation filters. It should also be noted that the model for sampling uses **temperature** [2], set to 0.8 is good for balancing lexical variety and factual consistency.

[2] **Temperature.** During text generation, the *temperature* parameter T controls the randomness of sampling from the model's output distribution. Given the logits z_i for each token, probabilities are computed as

$$P(w_i) = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}.$$

Lower values ($T < 1$) make the distribution sharper and outputs more deterministic, while higher values ($T > 1$) flatten it, producing more diverse but less predictable text.

3.8 Summary

Therefore, the information deducted from the literature up to this point are:

- **Imbalanced and small-data regimes gain the most.** Light, local DA such as EDA and AEDA delivers the clearest gains when only a few hundred labeled examples are available; benefits taper as datasets grow, and aggressive edits can introduce label noise. In high-stakes domains like finance, it is safer to use low edit rates and punctuation-only perturbations (AEDA or constrained EDA) rather than heavy rewrites. [11, 28, 32]

- **Match DA families to task and input format.** For text classification over relatively short inputs, most DA families (noising, paraphrasing, sampling) are viable; for long or free-form documents, sentence-/paragraph-level paraphrasing (BT, LLMs) is often more effective than token-level noise, while structured outputs (e.g. slot labels, fixed templates) require paraphrases that explicitly preserve format constraints. [15]

- **Prioritize semantic safety and domain anchors.** Effective DA must trade off *label preservation*, *diversity*, *fluency*, and *task-awareness*. Role-aware methods such as STA, conservative EDA settings and similarity/overlap filters all aim to protect class-defining terms while editing peripheral tokens, thereby limiting label drift and reducing noise amplification. [8, 15]

- **Generative methods as strong but costly baselines.** Back-translation remains a robust, domain-agnostic paraphrasing baseline; multiple pivots can increase lexical and syntactic variety but with diminishing returns and higher cost, making an archive almost mandatory in production. Label-conditioned LLM paraphrasing further increases diversity and can better mimic real-world stylistic variability, provided that prompts and filters enforce semantic faithfulness. [15, 27]

Chapter 4

Project Definition

4.1 Goals

Problem and setting. This project starts from a practical need: build a local NLP classifier that runs on two GPUs available to us at the HPC4AI center (UniTO). Given the *investment policy*[1], the classifier must assign each fund to one category among different taxonomies, for example: (i) the main asset class (e.g., Equity vs. Bond), (ii) the geographical area (e.g., Global, Europe), and (iii) the sector (e.g., Financials, Technology, Natural Resources). In short: *read the policy, output the right category for the chosen scheme.*

Models and context length. We first used RoBERTa-base with a 512-token window, which already covered $> 90\%$ of policies without truncation in our corpus. To reduce residual truncation and to keep more context for long policies, we then switched to a long-context encoder (Longformer), with an attention window up to 4096 tokens.

Main challenge: class imbalance. Our labels follow a complex but not so rare distribution: a few categories have many examples, while several others have very few, this is due to the real world imbalance distribution with which asset managers select the elements that compose their investment instruments. This has two practical consequences:

1. **In training.** With standard cross-entropy, the total loss is a sum over samples. If class A appears $100\times$ more than class B, gradients from A dominate the updates. The model therefore learns a strong "majority prior" and tends to predict the frequent classes more often, even when the text hints at a rare class and only apart from the cases in which feature from class B are extremely dominant over the feature of class A. In mini-batches, the same effect repeats: batches often contain many majority samples and few (or zero) minority ones, so minority patterns are seen less and memorized poorly.

[1] Policy text extracted by analysts from the KIID and related fund documents.

2. **In evaluation.** Overall (micro) accuracy may look high because the model is correct on majority classes and those weight heavier on the weighted sum, but *macro* metrics (e.g., macro-F1, macro-recall) expose the problem: minority classes get low recall and are often misclassified as a popular neighbor class.

We classify by different categories. Each taxonomy is imbalanced. Business-wise, we need reliable predictions across categories, not only for the common ones. Missing a minority label (e.g., a specific sector or a less common asset type) may propagate a wrong tag downstream. Technically, imbalance also limits representation quality: rare categories have fewer examples to teach the model the relevant vocabulary, collocations, and decision boundaries.

If a class is *well represented*:

- the model observes many variations of how that class is expressed (different wordings, contexts, document lengths);
- its embeddings and attention patterns for class-indicative phrases become stable;
- the decision boundary around that class becomes sharper and more confident.

If a class is *under-represented*:

- the model only sees a handful of phrasings and may overfit to those exact sentences; specifically, it can learn only the way in which the fund (the one that invests in these minority classes, because usually just a handful of fund invest in some specific "sector") in question writes down its own policy and not understand how recognize them if they are written in another way;
- it confuses the class with semantically close majority classes;
- in calibration, the model is overconfident on majority predictions and underconfident on rare ones.

Why we use textual data augmentation. DA increases the *effective* sample size of minority classes by creating extra, label-preserving variants of their texts. For us, this serves three goals:

1. **Reduce majority bias in gradients.** By adding minority examples, batch updates include more minority signals, so the model adjusts its boundary instead of defaulting to the prior. Apart from the quality of the new samples, the model is now aware that these classes also appear clearly within the dataset and they not seem like outlier.
2. **Improve coverage of minority language.** Paraphrases and light edits expose the model to more wording diversity around the same label (synonyms, syntactic changes, different discourse order), improving generalization and robustness to errors and differences in writing techniques.

3. **Lift macro metrics** The target is higher minority recall and macro-F1, without compromising performance in more common classes.

Because investment policies use precise financial language, we prefer *conservative* DA that does not change the underlying facts or the label: (i) small lexical/noising edits that do not alter meaning (e.g., punctuation-based AEDA), (ii) back-translation or controlled paraphrases that rephrase sentences while preserving technical terms and (iii) do this changes having a selective decision to protect class-indicative tokens. This keeps label drift low while still adding useful variation, we have to find the right trade-off between increasing cluster's size and their "density".

Project goals

G1: Deliver a working local classifier.

- Build an end-to-end pipeline that reads policy text and assigns a category in each target taxonomy.
- Run inference and fine-tuning *locally* (two GPUs at HPC4AI), respecting compute and memory limits.
- Decide when to use short or long-context encoding if helpful (fallback from 512 tokens to 4096 tokens) to avoid losing decisive sentences and on the other hand to avoid wasting available machine time.

G2: Study textual DA for imbalanced classes.

- Evaluate whether DA improves *minority-class* recall and macro-averaged metrics (e.g., macro-F1) without harming majority classes.
- Compare simple, robust DA families that are suitable for financial text: (a) *noising/lexical* edits such as EDA/AEDA; (b) *paraphrasing* (incl. back-translation and model-based paraphrases); (c) *role-aware* edits that avoid altering class-indicative terms (STA). DA acts as regularization and can strengthen local decision boundaries; surveys and prior work motivate these choices.
- Favor low-risk methods for label preservation. For example, AEDA inserts punctuation (keeps all words, low drift risk), while role-aware STA edits *non-core* words and protects class-indicative tokens.
- Decide if and what is better performing in our specific domain of application.

G3: Verify the suitability of compact PLMs in a real setting.

- Once the DA is set up and the problem of undersized classes has been already investigated and so we are in the best possible working condition.
- Test whether RoBERTa/Longformer (which have the constraint of being small enough to run locally) are adequate for KIID-style language under our realistic GPU limits.

- Quantify the value of longer context for the subset of long policies (trade-off: speed/memory vs. fewer truncation errors) and verify if the more complete dataset with all the long policy saved gives performance improvement.

4.2 Research Objectives

In this chapter we will turn the project goals into *actual* research questions and define the quantitative criteria we will use to answer them. We will study noising edits that keep all words (e.g., AEDA) and *role-aware* edits that protect class-indicative tokens (STA) mixed with paraphrasing using external LLM and what we will use to analyze the outputs. These choices are supported by the prior theoretical work on textual data augmentation for classification task.

4.2.1 Research questions (RQs)

RQ1 (Model adequacy under local constraints). *Are compact PLMs (RoBERTa-base; Longformer-base) sufficient for our KIID-policy classification tasks under two GPUs?*
Hypothesis H1: With careful fine-tuning and context management, small encoders achieve operationally useful performance to be implemented and start producing.

RQ2 (Effect of an imbalance dataset). *Does great imbalance in between the categories to be classified in make a big problem that solved could definitely improve classifier performance?*
H2: The imbalance between classes certainly affects the quality of a classifier of this type, so a solution must be sought, and there is a need for an analysis to be conducted that examines whether the trade-off between improvement in this respect and non-improvement is worthwhile.

RQ3 (How DA works when applyed to our domain). *What behavior DA will show once it will be applied on our policies text?*
H3: DA could show an increase in clusters dimensions, without, however, having a centroid drift, so as to maintain consistency but generate greater controlled diversity.

RQ4 (Effect of DA on imbalance). *Does textual DA increase minority-class recall and macro-F1 without hurting majority classes?*
H4: Conservative DA improves macro-recall and macro-F1, and keeps micro-accuracy stable (Δ not negative).

RQ5 (Which DA works best in finance). *Among easy/noising DA (AEDA), back-translation/GPT-paraphrasing, and role-aware STA, which gives the best trade-off between label preservation and generalization on policy texts?*
H5: A mix of different methods could donate the final model the right amount of

robustness to noise and the correct number of actual policies to be able to capture information.

RQ6 (Long context value). *When policies are long, does a 4096-token window (Longformer) improve performance over a 512-token window (RoBERTa)?*
H6: On documents that would be truncated at 512 tokens, longer context increases macro-F1 and reduces "near-miss" confusions, giving the model the opportunity to look at information that are widely scattered throughout the text.

4.2.2 Model classification metrics

There are two type of results that we want to analyze, per-class results and across-class results.

Per-class analysis

For this analysis of results we compute standard, widely used metrics: **Precision**, **Recall**, and **F1-score**.

- **Precision:** measures the model's ability to avoid labeling a negative as positive. It is defined as the ratio between true positives (TP) and all predicted positives ($TP + FP$):

$$\text{Precision} = \frac{TP}{TP + FP}$$

  A high Precision means that, among the examples predicted as positive, most are actually correct; it tells how many correct positive predictions we obtain out of all positive predictions.

- **Recall:** measures the model's ability to correctly find all positive cases. It is computed as the ratio between true positives and all actual positives ($TP + FN$):

$$\text{Recall} = \frac{TP}{TP + FN}$$

  A high Recall indicates that the model misses few positive examples (few false negatives); out of all truly positive cases, it shows what fraction are correctly classified as positive.

- **F1-score:** is the harmonic mean[2] of Precision and Recall, providing a single balanced value when both precision and completeness matter. It is defined as:

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

[2] The harmonic mean is type of average that gives more weight to smaller values. It is defined as the reciprocal of the arithmetic mean of the reciprocals of the values. For two positive numbers a and b, the harmonic mean is: $H = \frac{2ab}{a+b}$. In the context of the F1-score, this property ensures that both *Precision* and *Recall* must be high for the F1 to be high: if either one is low, the harmonic mean decreases sharply. This makes it a more balanced measure than the arithmetic mean when combining rates or ratios.

The F1-score is particularly useful with imbalanced classes, since it strongly penalizes models that have a large mismatch between Precision and Recall.

Across-class analysis

When we want to look forward to have a complete picture of what is happening we have to analyze the overall metrics computed over the entire dataset and not only on one class at a time. We look at metrics that *aggregate over classes* and at tools that explain *where* errors happen.

Accuracy and balanced accuracy. **Accuracy** is the fraction of correctly classified examples in the whole dataset:

$$\text{Accuracy} = \frac{\sum_{k=1}^{K} TP_k}{N},$$

where N is the total number of examples. It is easy to read but can be misleading when classes are imbalanced (majority classes dominate the score, each class is weighted as its number of examples over the total number of them).

When labels are skewed we also report **balanced accuracy**, the mean true positive rate across classes:

$$\text{BalancedAccuracy} = \frac{1}{K}\sum_{k=1}^{K} \text{TPR}_k = \frac{1}{K}\sum_{k=1}^{K} \text{Recall}_k,$$

which coincides with *Macro-Recall*, where *Macro* indicates that, indeed, we are using equally weighted classes, so minority classes are not washed out by the majority.

The concept of equally balance may in turn lead to misunderstanding because, because, while it is true that in this way we can read the actual behavior of the classifier on a random object of any class, it is also true that the real world (given that the sample is not artificial but derives from a homogeneous extraction of it) behaves, with high probability, just like the weighted case. Therefore, even the weighted metric (i.e., cases in which each class retains its real weight, proportional to the number of examples) has its own effective interpretation from which it is possible to extract useful information that sometimes, even if theoretically it should not be so, more accurately reflects the behavior when applied to a homogeneous dataset.

Averaging schemes over classes. Besides per-class values, we summarize performance with standard averages. Each one answers a different question and should be read accordingly.

- **Macro average (all classes equally important).** Compute the metric *per class* and take the simple mean:

$$\text{Precision}_{\text{macro}} = \frac{1}{K}\sum_{k=1}^{K} \text{Precision}_k, \quad \text{Recall}_{\text{macro}} = \frac{1}{K}\sum_{k=1}^{K} \text{Recall}_k, \quad \text{F1}_{\text{macro}} = \frac{1}{K}\sum_{k=1}^{K} F1_k.$$

Every class counts the same, no matter how many examples it has. Macro averages tell you: "On average, how well do we do per class?"
Use under class imbalance or when minority classes matter. A low Macro-F$_1$ usually means one or more classes are neglected.

- **Weighted average (respect dataset frequency).** Weight per-class metrics by the class support n_k (number of true examples):

  $$\text{Precision}_{\text{weighted}} = \frac{\sum_{k=1}^{K} n_k \,\text{Precision}_k}{\sum_{k=1}^{K} n_k},$$

  and analogously for Recall and F1.

  Frequent classes influence the score more; very small classes have little impact.
  Use when you want a single number that reflects the dataset composition (e.g., "typical overall user experience").

- **Micro average (global, instance-level view).** First sum contingency counts over classes, then compute the metric:

  $$\text{Precision}_{\text{micro}} = \frac{\sum_k TP_k}{\sum_k (TP_k + FP_k)}, \qquad \text{Recall}_{\text{micro}} = \frac{\sum_k TP_k}{\sum_k (TP_k + FN_k)}.$$

  Treats every *example* equally (in multi-label: every (*example*,*label*) pair), is prioritize the global equilibrium between true positive and false positive independently from the class. In single-label multi-class, Micro-F1 = Accuracy.
  Use when overall correctness is the goal and class frequency should drive the score.

- **Samples average (only for multi-label).** Compute the metric per *example* (using its entire label vector), then average:

  $$\text{F1}_{\text{samples}} = \frac{1}{N} \sum_{i=1}^{N} \text{F1}_i.$$

  This metric doesn't work in pure multi-class case (like our, in which for every example just one class is the right one, I can not have more than one right answer simultaneously), it wokr in multi-label situation. Each document counts once, no matter how many labels it has.
  Use when fairness across documents is important (documents may carry many vs. few labels).

Confusion matrices. To inspect *which* classes are confused, we use the $K \times K$ confusion matrix M, where $M_{i,j}$ counts examples with true class i predicted as j. Two normalized views are especially helpful:

$$\tilde{M}^{\text{row}}_{i,j} = \frac{M_{i,j}}{\sum_{j'} M_{i,j'}} \quad \text{(rows sum to 1, shows per-class } \textit{recall} \text{ distribution)},$$

$$\tilde{M}^{\text{col}}_{i,j} = \frac{M_{i,j}}{\sum_{i'} M_{i',j}} \quad \text{(columns sum to 1, shows per-class } \textit{precision}\text{)}.$$

Row-normalization highlights where each true class goes (near-miss vs. far errors), while column-normalization highlights how "pure" each predicted class is.

Table 4.1: Binary confusion matrix.

	Predicted condition	
Actual condition	**Positive (PP)**	**Negative (PN)**
Positive (P)	True positive (TP)	False negative (FN)
Negative (N)	False positive (FP)	True negative (TN)

Total population $= P + N$. TP/FP/FN/TN are the standard counts used to derive all the explained metrics.

Error hierarchy. Not all errors are equally severe in our domain. From our analysis of KID/KIID policy texts, some misclassifications stem from *label noise* rather than from the model itself. In practice, when analysts extract policies, missing or ambiguous information may lead to generic tags such as *"Undefined"*, *"Not Applicable"*, or *"Not Significant"*. These practices inject noise in the training set and make sharp decision boundaries harder to learn. To interpret results responsibly, we therefore distinguish:

- **Near-miss errors**: predictions that fall into a semantically adjacent class (e.g., *Long term* vs. *Mid/Long term* as maturity class), often reflecting ambiguity in the source text or inconsistent labeling. For this type of misclassifications a *soft* labeling will be introduced in which a mismatch in classification made from the model for really close classes will be considered not a model mistake. This come from the fact that since the policies are manually assigned to specific classes by analysts, there may be minor errors or ambiguous classifications which, even for a different analyst, might result in assignment to one of the adjacent classes. In these soft cases, we consider the model as a different analyst.

- **Far errors**: predictions that contradict the dominant signal in the policy (e.g., *Equity* vs. *Bond* as main asset class), these are the error that any human analyst will ever do on the same task.

We primarily report the standard metrics above; qualitatively, we read the confusion matrix with this *error hierarchy* in mind to separate boundary confusions from truly off-target predictions.

4.2.3 DA performance metrics

To verify that textual DA methods inject *useful variation* without distorting meaning nor collapsing the distance between classes, we implemented few *personalized* metrics that could give us an idea of where the new generations are located within the *embedding space*.

For calculating them we have to vectorize each text after having a few augmentation of it in each one of the possible methods chosen, then with the generated vector of 768 dimensions (the input vector in RoBERTa-model) we could find the similarity $s(\cdot,\cdot)$ using **cosine similarity**[3] and induced cosine distance $d(\mathbf{u},\mathbf{v})=1-s(\mathbf{u},\mathbf{v})$.

From the numbers displayed by these metrics we are able to understand if DA is: (i) preserving label, (ii) creating diversity and coverage of the area around the centroid and (iii) protect jargon and meaning.

The four metric families are summarize below. Throughout, we denote by $\mathcal{Y}$ the set of classes and use $s(\mathbf{u},\mathbf{v}) = \cos(\mathbf{u},\mathbf{v})$.

Metric0 - Intra-group *orig vs. aug* similarity

For each original text i we measure how close its augments remain:

$$\text{Metric0}_i = \left[s(\mathbf{e}_i^{\text{orig}}, \mathbf{e}_{i,1}^{\text{aug}}), \ldots, s(\mathbf{e}_i^{\text{orig}}, \mathbf{e}_{i,m}^{\text{aug}})\right].$$

High values (e.g., medians $\gtrsim$ 0.9) indicate strong semantic faithfulness. A very wide spread flags specific augmentation operations that may degrade the signal.

Metric1 - Within-class density

For each class y, we compute average pairwise similarity among *originals* and among *augments*:

$$\text{mean}_{\text{orig}}(y)=\frac{1}{\binom{n_{\text{orig}}}{2}}\sum_{i<j} s(\mathbf{e}_i^{\text{orig}}, \mathbf{e}_j^{\text{orig}}), \quad \text{mean}_{\text{aug}}(y)=\frac{1}{\binom{n_{\text{orig}}m}{2}}\sum_{i<j} s(\mathbf{e}_i^{\text{aug}}, \mathbf{e}_j^{\text{aug}}).$$

A mild increase of mean_{aug} over $\text{mean}_{\text{orig}}$ is desirable: augmented samples cluster near class prototypes while still adding lexical variety.

Metric2 - Inter-class similarity on *originals*

For each class y, we average similarity between its originals and originals from *other* classes:

$$\text{Metric2}(y) = \frac{1}{|\mathcal{Y}\backslash\{y\}|} \sum_{y'\neq y} \frac{1}{n_{\text{orig}}^2} \sum_{i,j} s(\mathbf{e}_{i,y}^{\text{orig}},\, \mathbf{e}_{j,y'}^{\text{orig}}).$$

Lower is better: it indicates better separation between class manifolds.

Sentence-embedding cosine scores are not zero-centered; unrelated pairs often sit around a mid-high baseline, one will never find a zero cosine similarity in between this embedded texts. We therefore compare *relative* levels (vs. Metric1) and *deltas* after DA, rather than imposing an absolute cut-off.

[3] Cosine similarity measures the angular closeness between two embedding vectors $\mathbf{u},\mathbf{v}\in\mathbb{R}^d$: $s(\mathbf{u},\mathbf{v}) = \frac{\mathbf{u}\mathbf{v}}{\|\mathbf{u}\|\,\|\mathbf{v}\|}$. It ranges from -1 (opposite direction) to 1 (identical direction), and is widely used in sentence-embedding spaces because it captures semantic similarity independently of vector magnitude.

Metric3 - Inter-class similarity on *augments*

As Metric2, but computed on class-y *augments* against other classes *augments*.

$$\text{Metric3}(y) = \frac{1}{|\mathcal{Y}\setminus\{y\}|} \sum_{y' \neq y} \frac{1}{n_{\text{aug}}^2} \sum_{i,j} s(\mathbf{e}_{i,y}^{\text{aug}},\, \mathbf{e}_{j,y'}^{\text{aug}}).$$

The goal is to keep Metric3(y) at most comparable to Metric2(y). If Metric3$(y) >$ Metric2(y) systematically, augments are creating "gray zones", where there is more overlapping or proximity between different concepts, that make classes more confusable.

Visualization of possible scenarios

Scenario	Metric0	Metric1	Metric2 vs. Metric3
Ideal	high ↑	≈ or mildly higher ↑	stable: M3 ≤ M2
Too much variation	low ↓	lower ↓	worse: M3 > M2
Redundant augments	very high ↑	much higher ↑↑	unchanged ≈

There is no single universal cosine threshold; sensible ranges depend on the task. Token- or span-level labeling often adopts very *strict* filters around $\theta \approx 0.9$ to guarantee near-identical semantics, whereas sentence-level classification tolerates *looser* filters (e.g., $\theta \approx 0.5$) since labels are stable under broader paraphrases. In our financial-policy setting, where terminology precision matters but we also want lexical diversity, we adopt $\theta = 0.8$ as an empirically reasonable compromise: it is high enough to preserve the semantic core, yet not so restrictive as to suppress diversity. Instead, we sometimes have the opposite problem, since θ is occasionally too high when we need more separation between classes. This choice also keeps inter-class separability stable while expanding the training set with faithful variants.

We then only apply soft transformation for converting similarities to distances when building the compact "drift/compactness/separation/leakage" table in the following chapter.

Metric0–3 aggregations

The table in §5.1.2 reports one row per DA family with *mean / median / p90* taken across classes. We derive the four displayed columns as follows.

Per-class transforms. For each class y:

$$\begin{aligned}
\text{Drift}(y) &= 1 - \overline{\text{Metric0}}(y) \\
\Delta\text{Compactness}(y) &= \left[(1 - \bar{s}^{\text{aug}}(y)) - (1 - \bar{s}^{\text{orig}}(y))\right] = \bar{s}^{\text{orig}}(y) - \bar{s}^{\text{aug}}(y) \\
\text{Centroid dist. proxy}(y) &= 1 - \text{Metric2}(y) \\
\text{Leakage proxy}(y) &= \max\{0,\ \text{Metric3}(y) - \text{Metric2}(y)\}.
\end{aligned}$$

We chosen this aggregation for giving a simpler meaning to the data displayed subsequently, as each one of these has an intrinsic meaning more recognizable then the hard metric as they are evaluated.

- **M1 - Augmentation drift.** This explains how far augments move from their source originals, per class y (faithfulness vs. diversity). For each original i in y with augments $\{\mathbf{a}_{i,1}, \ldots, \mathbf{a}_{i,k}\}$ we compute $d(\mathbf{o}_i, \mathbf{a}_{i,j})$ if the pairing is known and where $\mathbf{o}$ is the original text; otherwise we use the nearest original $d(\mathbf{a}, \mathcal{O}_y) = \min_{\mathbf{o} \in \mathcal{O}_y} d(\mathbf{a}, \mathbf{o})$. We then summarize the per-class distribution with mean/median/p90/p95.

- **M2 - Compactness shift.** It shows what change in within-class cohesion ("density" of the class) after adding augments. We compute for each class y the mean of the upper triangle[4] of the within-class cosine distance matrix on *originals* only ($\overline{d}_{\text{intra}}^{\text{orig}}$) and on *originals+augments* ($\overline{d}_{\text{intra}}^{\text{orig+ aug}}$), here, differently from *Metric1* we also keep the original text when computing the mean adding the augmented texts. A light *decrease* (after minus before) indicates tighter class clouds without collapse; a large decrease may indicate redundant augments while a large increase suggests over-diversification or leakage and could bring to overlapping classes.

- **M3 - Centroid distance matrix.** It give the inter-class margins at the level of class centroids, before and after DA. We compute class centroids $\boldsymbol{\mu}_y$ as the mean embedding per class and form the matrix $D_{y,y'} = d(\boldsymbol{\mu}_y, \boldsymbol{\mu}_{y'})$ (where all diagonal points are 0). When DA is included, we recompute centroids on the expanded (originals embedding included) set to observe any drift. Stable or larger different centroid distance after DA ⇒ preserved/improved margins while shrinking distances warn about class overlap.

- **M4 - Nearest-centroid label consistency.** Is the fraction (%) of augments whose nearest class centroid does *not* belong to their own class. Is found calculating for each augmented vector $\mathbf{a}$ of class y the $\arg\min_{c \in \mathcal{Y}} d(\mathbf{a}, \boldsymbol{\mu}_c)$ with centroids computed on *originals* (so they are not moved around by augments and fit the initial class position in embedding space, that should be coupled with the class meaning) and counting how many assign to $c \neq y$ and dividing by total augments in y. Obviously lower is better; spikes in specific classes can reveal where a DA method flips semantics, so where specific attention is needed.

Aggregation across classes. Given class-wise values $\{v_y\}_{y \in \mathcal{Y}}$ (for any of the four transforms above) we report, in the table, three summaries:

$$\text{mean} = \frac{1}{|\mathcal{Y}|} \sum_y v_y, \qquad \text{median} = \text{median}\{v_y\}, \qquad p90 = \text{quantile}_{0.9}\{v_y\}.$$

We use *uniform* class weighting to prevent head classes from dominating; $p90$ captures worst-case tails (useful to detect risky augments that affect only a few classes).

[4] Only of the upper triangle because the other half is just the repetition as the distance by definition is equal looked both sides.

Additional Metrics

M5 - Global clustering indices. With that we summarize the separation vs. cohesion after DA using already know metrics: we compute Silhouette (with cosine), Davies-Bouldin (DB), and Calinski-Harabasz (CH) on the combined set (originals + augments).

Metrics 1–4 provide *local* evidence (per item, per class) about fidelity, cohesion and cross-class overlap. However, augmentation can be ultimately judged on the *global* geometry of the *combined* embedding set (originals + augments): do classes remain compact and well separated *overall*? M5 complements M1–M4 with three standard, label-aware indices that summarize cohesion vs. separation at dataset scale.

We compute:

$$\text{Silhouette}(i) = \tfrac{b(i)-a(i)}{\max\{a(i),b(i)\}}, \quad \text{DB} = \tfrac{1}{K}\textstyle\sum_{c=1}^{K} \max_{c'\neq c} \tfrac{\sigma_c+\sigma_{c'}}{d(\boldsymbol{\mu}_c,\boldsymbol{\mu}_{c'})}, \quad \text{CH} = \tfrac{\mathrm{tr}(B)/(K-1)}{\mathrm{tr}(W)/(N-K)}.$$

Here $a(i)$ is the mean distance from point i to its own cluster, $b(i)$ the minimum mean distance to any other cluster; $\boldsymbol{\mu}_c$ and σ_c are the centroid and average within-cluster distance from its centroid of class c; B and W are between- and within-cluster scatter (distance) matrices; K is the number of classes and N the number of samples. We use cosine for Silhouette and Euclidean for DB/CH (as in common libraries). With ℓ_2-normalized embeddings, $\|\mathbf{u}-\mathbf{v}\|_2^2 = 2(1-\cos(\mathbf{u},\mathbf{v}))$, so the three indices still move coherently and can be compared.

- **Silhouette** balances cohesion ($a(i)$) and separation ($b(i)$) at the *point level* and is robust to moderate class-size imbalance; it rises when augments tighten classes without pushing them toward others.
- **Davies-Bouldin (DB)** penalizes clusters that are simultaneously *wide* (large σ_c) and *close* to a neighbor (small $d(\mu_c,\mu_{c'})$); lower is better. It is sensitive to "gray zones" that Metric3 is designed to detect.
- **Calinski-Harabasz (CH)** rewards between-class dispersion relative to within-class scatter; it rises when augments expand coverage *inside* classes without inflating overlap.

Together, these indices provide an interpretable, complementary summary: Silhouette (balanced pointwise view), DB (worst-case cluster overlap), CH (global variance ratio). We interpret them *jointly* (Silhouette↑/CH↑/DB↓) and primarily in terms of *deltas vs. baseline* rather than absolute thresholds while comparing for different DA settings.

Relation to previously shown metrics. Linking to the previous M1-4 definition: lower M1 (augmentation drift), i.e., augments staying closer to their sources, and a slightly negative M2 (compactness shift), i.e., modest tightening of within-class distances, typically yield Silhouette↑ and DB↓ (and often CH↑) by shrinking within-class scatter without collapsing it. Conversely, if M3 (centroid distances) *shrink* after DA, class margins contract and we usually observe Silhouette↓, DB↑, CH↓. Finally, higher M4 (nearest-centroid label inconsistency) signals leakage of augments toward foreign classes and shows the same adverse signature—Silhouette$\downarrow$/DB$\uparrow$/CH$\downarrow$—while also pinpointing which classes and DA operators are risky.

The steps in the approach we follow are: (i) embed all samples, (ii) ℓ_2-normalize vectors, (iii) compute Silhouette with cosine, DB/CH with Euclidean and (iv) summarize per DA family as mean/median/$p90$ *across classes or runs.*

Edge cases and safety applied.

- *Pairing vs. nearest-original fallback (M1):* When the number of per-original augments is irregular, `_normalize_aug` reconstructs a best-effort structure; if exact alignment fails, drift is computed to the nearest original. This yields a conservative proxy and avoids discarding data.
- *Small classes:* With < 2 samples, within-class means are undefined; functions return `NaN` (explicit in the dataframes) to prevent biased aggregates.
- *Complexity:* Within-class pairwise computations scale with $O(n_y^2)$ but are restricted to per-class blocks; centroid operations are $O(n)$ and cheap.

Expected results. (i) low-to-moderate drift (M1) without heavy tails, (ii) slightly tighter or similar compactness (M2), (iii) stable or larger centroid separations (M3), (iv) low leakage (M4), and (v) better Silhouette/CH and lower DB (M5). Any metric that degrades systematically across classes is a red flag to adjust filters or reduce that DA's weight is what data suggests.

PCA maps. We include two *example* PCA[5] views (Figure 4.1 and 4.2) to make the geometry concrete and give the reader the possibility to visualize what we are quoting in the section. The complete figure set per method/taxonomy is reported in the *Results* chapter (§5).

This chapter defines the metrics and shows two illustrative plots. *All quantitative results* (full per-class summaries, centroid distance matrices, leakage rates, and global clustering indices), together with the complete PCA panels for each DA method and taxonomy, are presented and analyzed in Chapter 5.

4.3 Research Methodology

This section explains *how* we ran the study: (i) the type and structure of the data we used, (ii) the end–to–end pipeline and the validation protocol that led us to the final configuration, and (iii) the test setup. Results are deferred to Chapter 5.

[5] We visualize sentence embeddings by projecting them to 2D with Principal Component Analysis (PCA), that is a linear dimensionality reduction method. It finds orthogonal directions (principal components) that capture maximal variance in the original high-dimensional data (e.g., 768-d RoBERTa vectors) and projects the data onto the top components so we can plot and inspect clusters seeing them over 2 or 3 dimensions.

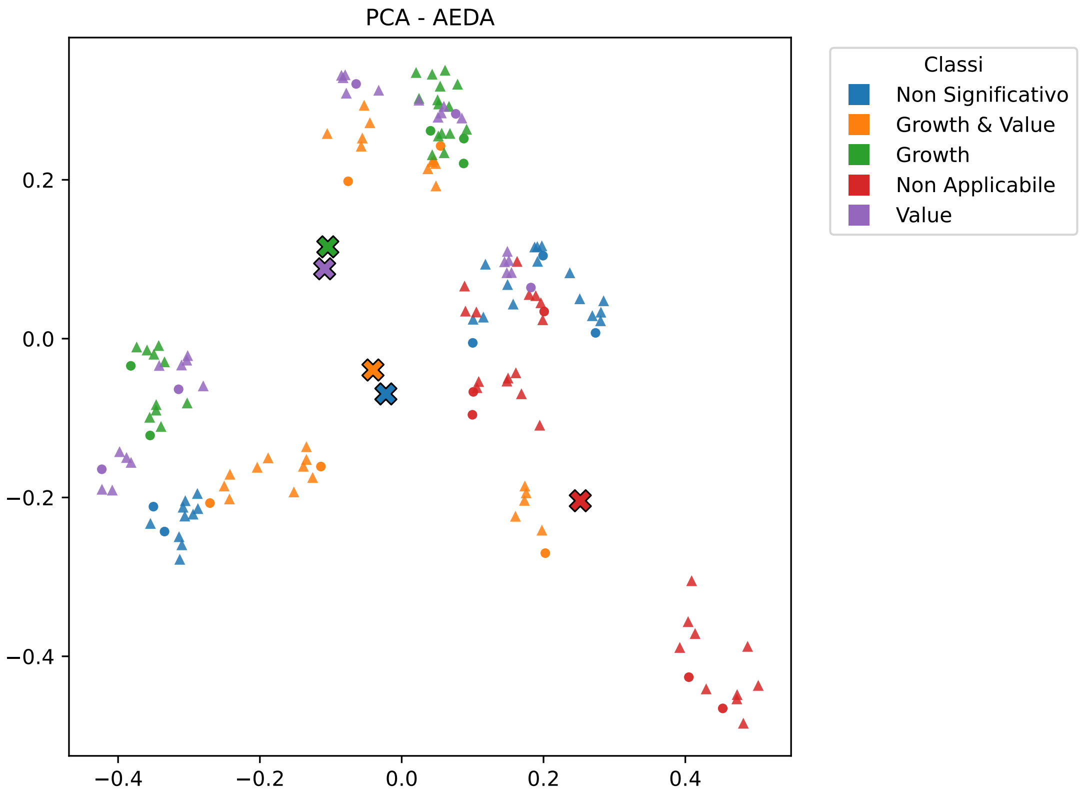


Figure 4.1: PCA scatter (*style*, AEDA). Originals (circles), augments (triangles), centroid (crosses). Desired pattern: local halos around class neighborhoods without bridges across classes.

4.3.1 Type of data and structure

There are two main datasets that have been used: one with a small policy length (the one extracted manually from professional analysts) and the second one that has the same set of classes and categories but consists of longer policies that don't fit the small context window of RoBERTa-base. Apart from the difference in text length and so information contained, the detailed number per each class are the same (e.g. the number of present missing values).

The corpus contains a total of **8,362** fund entries, each corresponding to one fund. Each row is uniquely identified by the numeric keys `ana_code` and `comparto`, while the column `cod_sicav` (910 distinct values) groups funds belonging to a society, we will use it to differentiate train and test entry because usually a company use a specific format to draft this type of informational document and dividing them could help the model avoid overfitting.

Text fields. There is a main textual columns `poli_GB`: the full investment policy in English, available for all 8,362 records, this could come from a translation made prior to the creation of the database or just from a document drafted in english, and we could

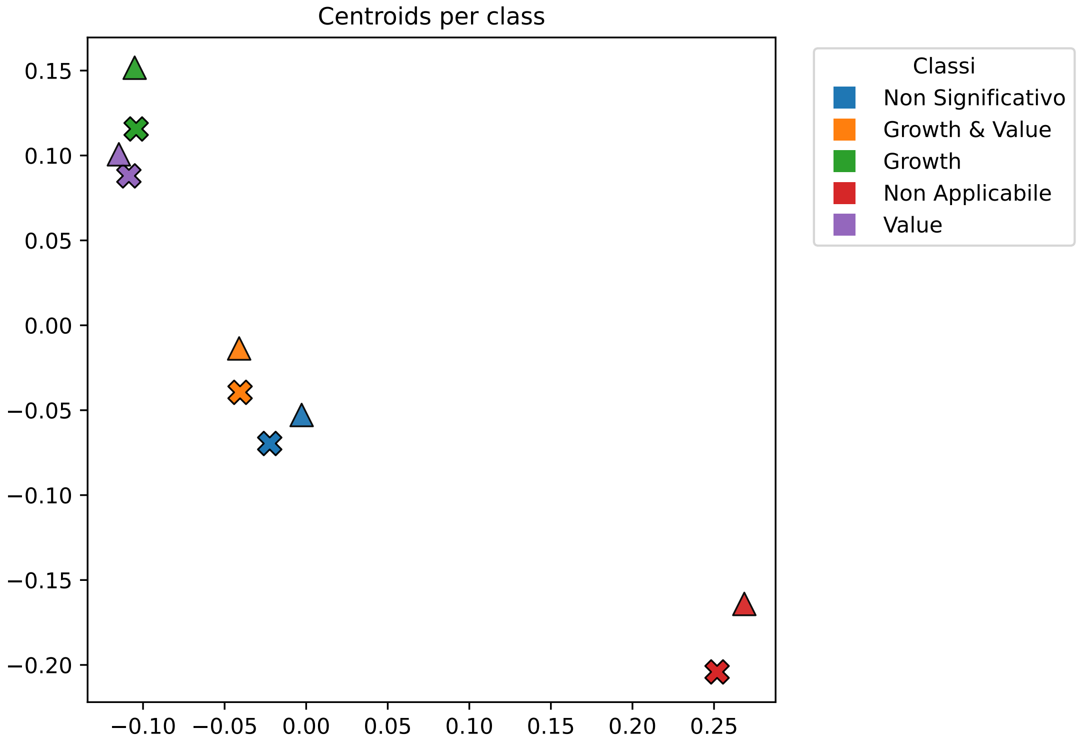


Figure 4.2: Centroid map (*style*, AEDA). Originals (crosses) and augmented (trinagles). We inspect within-class centroid drift and pairwise inter-centroid distances to ensure separability is not eroded by DA.

find the italian version if needed and available; in this case the textual input given to the neural model need to be written in english due to the fact that the PLM is trained on an English dataset and is therefore more likely to understand texts in the same language in terms of the type of syntax and grammar that distinguishes a particular idiom. As previously said the dataset have different policies length: (i) *short policies* one has median of about **156 words**, mean of 161 and maximum of 418, while (ii) *long policies* one has English policies longer and more detailed, with a median of about **413 words**, mean of 434 and maximum 1,340. Duplicated texts appear across funds due to multiple share classes or template reuse, `poli_GB` has 83 duplicates, those will be eliminated in train environment due to the use of few filter.

Label columns. The dataset includes several taxonomies that represent different classification targets (counts refer to the 8,362 rows in this file):

- `asset`: 16 classes; missing 1.6%. Top class *Azionari* (43.5%) followed by *Obbligazionari* (25.9%). Strongly imbalanced toward Equity/Bond; minor groups include Absolute Return variants, Monetary, Commodities, and Convertibles.

- `geo_area`: 6 classes; missing 1.7%. Top class *Globale / Aree multiple* (55.5%) then

Europa (25.0%); very small tails for *America Latina* (0.4%) and *Africa, Area Araba, Medio Oriente* (0.2%). Skewed to Global/Europe.

- `sector`: 27 classes; missing 1.7%. *Settori Multipli* dominates (56.7%), then *Non Significativo* (22.4%), with *Non Applicabile* at 6.3%. Single–sector buckets (e.g., *Materiali*, *Tecnologia*) are comparatively small. Highly skewed toward multi–sector/not-specific.
- `dimension`: 9 classes; missing 1.6%. *All Cap* (28.3%) and *Large & Mid Cap* (27.6%) lead; *Non Significativo* is 25.6% and *Non Applicabile* 9.0%. Moderate spread with many non-specified entries.
- `style`: 5 classes; missing 1.6%. *Growth & Value* (56.0%) dominates; *Non Significativo* 25.6%, *Non Applicabile* 9.0%, and pure *Growth*/*Value* are minorities (3.9% / 3.9%). Strong blend skew.
- `maturity`: 8 classes; missing 1.7%. *Non Applicabile* (47.3%) and *All Maturity* (40.6%) cover most cases; shorter buckets such as *Breve Termine* (3.5%) and *Breve–Medio Termine* (3.6%) form the tail. Mostly relevant to fixed–income funds.
- `rating`: 6 classes; missing 1.7%. *Non Significativo* (44.6%), *Tutte le Classi di Rating (Speculative e Investment Grade)* (31.1%), *Investment Grade* (16.0%), *Speculativa* (4.0%), *Non Applicabile* (2.7%); one record is *Senza Rating*. Often not applicable to non–bond funds.
- `issuer`: 5 classes; missing 1.7%. *Corporate* (55.5%) and *Corporate&Government* (34.1%) dominate; *Government* 6.3%, *Non Applicabile* 2.4%, *Sovranazionali* ≈0.0%. Concentrated in corporate issuance.
- `asset_exposure`: 31 classes; missing 0.7%. *100% Azioni* (43.3%), *100% Obbligazioni* (28.0%), *Flessibile* (11.7%); leveraged patterns (e.g., *300% Azioni*) exist but are rare. Skewed toward pure exposures.
- `type`: 23 classes; missing 1.1%. *Fondi a gestione attiva (Market Fund)* (45.7%), *Flessibili* (25.7%), *Passiva (Low TEV)* (19.3%); remaining strategies (e.g., *Fondi a Scadenza*, *Leverage Bull/Bear*, *Long/Short*) form a long tail. Dominated by active management.

It's easy to extract from the displayed data that the missing values have to be treated but they are not a bottle neck, the number over the total is very small. Instead in each class there are categories that have few examples; for each category, work was carried out together with domain experts, also taking into account the number of funds in the real world that can be labeled in this way (extra quantization is not optimal, as it is being too vague), to arrive at the final configuration shown in Table 4.2. Some of this classes (`asset_exposure` and `type`) have not been taken into account for the final autonomous classification because the number of internal categories were too high to have a decent number of example each (a lot of them were quasi-empty and for that reason very difficult also to be augmented since a baseline is necessary for augmentation models to work fine).

Normalization and Pruning. As we said, toghether with domain expertex few decision has been made to harmonize noisy labels and reduce sparsity, we applied three configuration files (i) authoritative merges and drops, (ii) keywords that can map to an intended label, and (iii) keywords to suppress. We performed:

- **Merges.**
  - `asset`: *Fondi Protetti* + *Fondi Garantiti* → *Fondi Protetti e Garantiti*; *Ritorno Assoluto (Derivati)* + *Ritorno Assoluto (Tassi & Valute)* + *Ritorno Assoluto* → *Ritorno Assoluto DTV*.
  - `dimension`: *Non Significativo* + *All Cap* → *Non Definito*.
  - `issuer`: *Sovranazionali* + *Government* → *Govern*; *Non Applicabile* + *Non Significativo* → *Non Definito*.
  - `sector`: *Non Significativo* + *Non Applicabile* → *Non Definito*.
- **Removals (low support / not reliable).**
  - `asset`: *Obbligazioni Ibride*, *Tassi di Interesse*, *Valute*, *Volatilità* (dropped).
  - `geo_area`: *America Latina*, *Africa, Area Araba, Medio Oriente* (dropped).
  - `rating`: *Senza Rating* (dropped).
  - `maturity`: *Non Significativo* (dropped).
- **Keyword guardrails.** Terms flagged, e.g., *volatility*, *VIX* → *Volatilità*, that were incompatible or belong to a specific class but written in different terms needs to be disambiguate or suppress before applying the final mapping.

Exposure and benchmark fields. Additional descriptive columns include:

- `currency_exposure` (54 patterns, describing hedged/unhedged currency structures);
- `benchmark`, which lists the declared reference index (56.2% non–missing, 2,868 distinct values).

Benchmarks are useful for qualitative validation but too sparse and heterogeneous to be treated as a predictive label and for that they were not counted as values to classify policies in.

Soft classes. As said in the previous section not every error weight equally, sometimes if the model gives an interpretation close to the analyst's one it could be considered not an error, because maybe it derived from an ambiguous policy or an ambiguous expert classification. With domain expert we have analyzed each class and where it was possible we have created a *soft labeling*, as we have called it, that takes into account these possible equivalences.

As shown in Fig. 4.3 adjacent classes are not considered error along with few particular label. Those *side links* are indeed this label that connect with a class that in some case is a general one when is not clearly explained.

Class	Categories kept
asset (10)	Azionari; Convertibili; Diversificati; Fondi Protetti e Garantiti; Materie Prime; Monetari; Obbligazionari; Ritorno Assoluto (Azionari); Ritorno Assoluto (Obbligazionari); Ritorno Assoluto DTV
geo_area (4)	America del Nord; Asia – Pacifico; Europa; Globale / Aree multiple
sector (26)	Abs/Mbs; Beni di consumo complessivi; Beni di prima necessità; Consumi discrezionali; Ecologia e ambiente; Energia / Materie prime; Energia e Utility; Finanziari; Immobiliare; Industria; Informatica; Infrastrutture; Materiali; Metalli preziosi & Minerali; Minerale / Metallurgico; Non Definito; Risorse Naturali; Servizi sanitari; Settori Multipli; Sicurezza; Stile islamico; Tassi di interesse e valute; Tecnologia; Tecnologia / Media / Telecomunicazioni; Telecomunicazioni; Tutti i settori ex Finanziari
dimension (8)	Gigante; Large & Mid Cap; Large Cap; Mid & Small Cap; Mid Cap; Non Applicabile; Non Definito; Small Cap
style (5)	Growth; Growth & Value; Non Applicabile; Non Significativo; Value
maturity (7)	All Maturity; Breve Termine; Breve–Medio Termine; Lungo Termine; Medio Termine; Medio–Lungo Termine; Non Applicabile
rating (5)	Classe di Rating Investment Grade; Classe di Rating Speculativa; Non Applicabile; Non Significativo; Tutte le Classi di Rating (Speculative e Investment Grade)
issuer (4)	Corporate; Corporate&Government; Govern; Non Definito

Table 4.2: Kept categories per class (after normalization).

Summary. There are a few identifier columns (`ana_code`, `comparto`, `cod_sicav`) that are numeric, while textual and categorical fields are strings, we will classify on this textual categories. In short, we work at the *fund* level and use the English policy text (`poli_GB`) as the model input. Target labels come from the curated taxonomies listed above, after normalization and pruning (Table 4.2); two high-cardinality fields (`asset_exposure`, `type`) are excluded from autonomous classification (also `sector` could be considerable as high cardinality and so we still had kept it for having a benchmark on these classes). Duplicates are filtered in training to prevent leakage, and overall missingness is low (∼1–2%). The class distributions are strongly skewed, which motivates the class-aware textual augmentation and the stratified, fund-level splits that underpin the validation and test protocol described next.

4.3.2 Pipeline overview

The complete flow has four phases: *(i) data cleaning and split*, *(ii) label selection and multiclass setup*, *(iii) model training with early stopping*, and *(iv) evaluation and logging.* Data augmentation (DA) is applied only on the *training* fold.

(i) Data cleaning. Starting from the table in §4.3.1, we build the input text by concatenating the fund name and the English policy: `ana_name` || `poli_GB` (renamed internally

Asset
Fondi Protetti e Garantiti
Ritorno Assoluto (Derivati)
Convertibili
Diversificati
Materie Prime
Monetari
Azionari ↔ Ritorno Assoluto (Azionari)
Obbligazionari ↔ Ritorno Assoluto (Obbligazionari)

Dimension
Non Applicabile
Large Cap ↔ Large&Mid Cap ↔ Mid Cap ↔ Mid&Small Cap ↔ Small Cap
Side link: Large&Mid Cap – Non Definito

Style
Non Applicabile
Growth ↔ Growth&Value ↔ Value
Side link: Growth&Value – Non Significativo

Maturity
Non Applicabile
All Maturity
Lungo Termine ↔ Medio/Lungo Termine ↔ Medio Termine ↔ Breve/Medio Termine ↔ Breve Termine

Rating
Non Significativo
Non Applicabile
Classe di rating Investment Grade ↔ Tutte le classi di rating (Speculative e Investment Grade) ↔ Classe di rating Speculativa

Issuer
Non Definito
Govern ↔ Corporate&Government ↔ Corporate
Side link: Govern – Corporate

Legend: ↔ adjacency (distance 1). Standalone = isolated. Side links noted with "–".

Figure 4.3: Soft-class proximity.

to `testo_completo_benchmark`). We apply:

1. Drop rows with missing `ana_name`, text, or label.
2. Merge and rename label values using a configuration table (values to drop or unify into a canonical class), then remove classes with fewer than 4 samples overall.

We avoid heavy normalization to preserve domain cues (tickers, percentages, durations, rating acronyms).

(ii) Multiclass setup. For each taxonomy (e.g., `asset`, `geo_area`, . . .) we train a single multiclass head (cross-entropy over K classes, specifically ti will be a K-way softmax head that will pick the highest final value) on the label set remaining after cleaning and the < 4 support filter; no one-vs-rest[6] reduction is used.

(iii) Train/val/test split and DA. Before the final training loop, we run the model to choose the hyperparameter, for every target class we stratify into *train/validation/test* (65%,10%,25%), maintaining the original positive ratio. DA (Chapter 3) is applied *only* to the *training* portion and never leaks across folds, the testing need to be done in the

[6] One-vs-rest is a different approach where instead of using a single K-way head we unfold the problem in K binary classifier that have to choose per each class: *class k* vs *not k* and at the end the highest *class k* value comparing all K classifier is taken.

real world so without our artifact texts. A median balancing method is then applied to the training split to equalize class sizes before model training (see Section 4.3.2).

(iv) Training loop with early stopping. We fine-tune the encoder (RoBERTa or Longformer) with a linear classification head, using AdamW, cosine LR scheduling, mixed precision and gradient accumulation (only in *Longformer* case) . We monitor the *training loss* (our "penalty" that the model tries to minimize) at the end of every epoch and save the best-checkpoint, we use the hyperparameter found in previous step (iii) so this time no validation set is used, we just train on different combination of input (with or without DA) and test it to compare performance. Early stopping halts when the training loss does not improve for p consecutive epochs (patience[7]), this is useful because could prevent over-training that sometimes is exhausting for the model, indeed, in this cases, even the loss value stops falling and starts to move sideways or even rise again, precisely at those moments we early stop the process. The final model is the one from the last epoch before stopping.

(v) Outputs. For each run we save: (a) a batch-loss curve (PDF); (b) an .xlsx report with a summary sheet, per-class F1, and both raw and row-normalized confusion matrices on the test set; and (c) a JSON file with the same metrics plus class order and confusion matrices. We also export per-sample test predictions with class probabilities. We do not tune a decision threshold on validation; predictions use the argmax over class scores.

Class balancing using median

Before training we normalize class sizes on the *training* split using a median target size. Let n_c be the count of class c in the training set and $m = \text{median}(\{n_c\}_c)$. We set a target size

$$T \;=\; \min\{\, n_c \;:\; n_c \geq m \,\}$$

(i.e., the smallest class size that is at least the median). For each class: (i) if $n_c \geq T$ we *downsample* to T (without replacement); (ii) if $n_c < T$ we *upsize* to T via textual DA. This "median method" keeps classes close to a realistic scale while avoiding over- or under-expansion.

Advantages. Choosing T around the *median* of $\{n_c\}_c$ is robust and cost–effective. In fact, T that minimizes the total number of edits (drops or additions) solves

$$\min_{t \in \mathbb{N}} \sum_c |n_c - t|,$$

whose minimizers are the (discrete) medians. Our definition $T = \min\{n_c : n_c \geq m\}$ picks a feasible class size (one that exists in the data) closest to the median from above, so (i) we

[7] Patience in early stopping is a parameter that specifies how many epochs the training process will wait for a performance metric on a validation set to improve before stopping, minimum number of epochs required.

never "invent" a fractional target, (ii) we avoid unnecessary extra upsampling compared to m, and (iii) we minimize the total number of required operations. Practically, this keeps the *synthetic:real* ratio bounded: at least half of the classes have $n_c \geq T$ (no augmentation, only mild downsampling), and at most half require DA. This reduces variance and the risk that gradients are dominated by synthetic items.

Other possible choices.

- **Mean:** sensitive to long right tails. A few very large classes pull the mean up, forcing heavy upsampling for many small classes and increasing the augmented share, resulting in higher label noise and a raise in overfitting risk).
- **Max:** sets $T = \max_c n_c$, which almost always implies upsampling *every* other class-unrealistic priors and a training set largely synthetic.
- **Min:** sets $T = \min_c n_c$, discarding most available evidence (massive information loss and higher variance, realistically could bring the number of example in training from thousands to tents).
- **Mode:** undefined/unstable for small $|\{n_c\}|$ and offers no robustness guarantees.
- **Other quantiles (e.g., Q_1 or Q_3):** Q_1 behaves like a conservative min (too much downsampling); Q_3 like a softened mean (too much upsampling). The median strikes the balance: limited drops on head classes, limited DA on tail classes.

Finally, we apply balancing *only* on the training split, so evaluation reflects the natural class priors. We also drop labels with fewer than 4 total examples to avoid extremely high-variance classes where any target T would imply unrealistic upsampling, still very few example classes need special attention and a very cautious use of DA.

4.3.3 DA method selection

Guided by the theory in Chapter 3, we selected a small set of augmentation families that balance *label preservation*, *diversity*, and *compute cost*. In short, we favored simple, efficient, highly controllable methods to fix class imbalance and add local variability, complemented by one medium–cost paraphrasing method and two LLM–based generators for extra diversity when needed. This choice follows common DA motifs (strengthening local decision boundaries; preserving meaning and form) and practical guidance from the literature.

Decision criteria. For each candidate DA we evaluated: (i) label safety on finance texts (risk of semantic drift or flipping the label), (ii) domain fit (e.g. keeps tickers, percentages, legal phrasing intact), (iii) diversity gain (how much it moves points in embedding space), (iv) throughput (samples/sec on our hardware), and (v) training stability (early-stopping curves, see Fig. 4.4a and Fig. 4.4b). The literature consistently warns that aggressive lexical edits can harm label preservation in text classification, whereas light noising and controlled paraphrases tend to be safer and this was our first rule to follow and verify in practice.

Selected methods

Therefore, of all the methodologies reviewed based on the characteristics sought, we selected some of them for further numerical evaluation. We created combinations where two or more techniques performed well in different areas and their combination could address the issues that they were unable to resolve individually and and diversify the risk of each one.

- **AEDA + EDA mix.** We use AEDA (punctuation insertion) because it is extremely fast, preserves all tokens, and empirically reduces overfitting; we then mix randomly at 50% classic EDA to inject minimal word–order noise.[8] AEDA is known to outperform pure EDA on several classification tasks while keeping semantics intact and being the easiest and most efficient method with respect to all the analyzed ones.

- **STA + AEDA mix.** We adopt STA's *role–aware* editing (protect "gold" class indicators; edit/inject around less critical words) to increase diversity without touching the core finance terms. In our pipeline we stack STA with AEDA to combine selective edits with ultra–light noising. This combination seems the best looking at the local one, STA itself, once generated the dictionary containing the role for each class using WordNet, it run locally at small cost.

All upcoming methods will be combined with components from one of the shown above that helps to increase or decrease diversification and robustness. The number of examples by paraphrasing is defined by a multiplier cap N of the number of examples in the class created through the use of paraphrasing.

- **Back–translation (BT).** We include a medium–budget paraphrasing route via BT to obtain sentence–level rewrites that keep meaning while changing surface form. This is a standard, label–preserving paraphrase technique for text DA. For this method we have to pay more attention because the translation is made using external software and each one is bounded to a cost. The downside is that nowadays each translation API work really well and create a great difference before and after the translation route is not usual.

- **GPT_1 (single-input).** A lightweight LLM paraphraser that takes *one* policy text and returns a semantically equivalent variant. We use it sparingly to add diversity beyond BT while keeping label safety high (constrained by the input prompt). The LLM knows the policy and the class over which we want to find the right category; this is because some information surpasses others in order of importance when we look for a class rather then another.

- **GPT_2 (multi-input).** A stronger generator that reads *four* in–class inputs and produces *two* synthetic texts that blend their info. We use it to stretch intra–class

[8] We use only some of the EDA available operations to avoid deletions/substitutions that might remove domain cues.

coverage when minority classes are very small (highest diversity, highest cost). This version has the advantage of being able to approach at the paraphrasing knowing more then one example and better understanding the writing approach used by whom draws up this kind of texts. It has also an increased diversity generation due to the fact that tends to learn the form in addition to the specific class features.

Methods we *did not* adopt. We avoided heavy, unconstrained lexical substitutions or aggressive deletions as individual DA, because in our domain they can drop numeric ranges, ticker symbols, or legal caveats that are predictive of the class. We also chose *not* to deploy a MixUp-style augmentation (also if we prototyped both input/embedding MixUp and hidden-state MixUp with soft targets) for four reasons: (i) finance policies are not linearly compositional, convex combinations in embedding space tend to produce off-manifold samples that blur decision boundaries and can wrongly train attention; (ii) MixUp requires a soft-label regime and different calibration/validation logic, making results not directly comparable to our hard-label DA pipeline, it is not generating directly a new example to train the model on; (iii) under Longformer (L=3072) with gradient accumulation, MixUp effectively doubles forward passes and memory pressure, conflicting with our VRAM budget, that is already a bottle-neck that needs to be optimized; and (iv) our goal was to reach the *median* per each class in size with *textual* samples, MixUp does not create usable texts, while other textual DA do. For these reasons we kept MixUp as a possible future work rather than part of the main research even though it is ready to be implemented.

Final balancing strategy. We target a per–class sample size equal to the first class cardinality larger then the *median* (large classes are downsampled to the median; smaller ones are augmented up to it). Within this budget, cheap methods (EDA/AEDA/STA) provide most of the volume; BT and GPT are used selectively for creating the first $N \times n$ (where n is the number of real example of the class while N is the budget allocated to the creation of a multiple of real texts given to the expensive methods) augmented texts and the remaining part is filled with a mix of the cheaper methods. We selected this line of work for 2 reasons: (i) when a lot of examples have to be created, using only paraphrasing method, this could also, by a little amount, insert a drift in the textual meaning and not create the noisy robustness that local noising-based method gives; (ii) not local method, are executed using external tools and creating thousands of samples only by those two could have made a considerable cost. Setting up generation in this way therefore means that classes with few examples are increased in number first by a certain amount of paraphrasing that is a multiple of the number of real examples in the class, and the remainder is filled with examples creating by noising. On the other hand, for classes with a larger number of examples, it is probably possible to cover the number of generations required using only paraphrasing methods, since much of the meaning (which we can see, for example, interpolated in the centroid) is already weighted on real examples and therefore the risk of slipping away from the starting point has decreased.

The overall recipe reflects well-known DA trade-offs between diversity, label preservation, and compute, and mirrors best practices reported in recent surveys.

Training notes. Each DA family received its own stable learning–rate setting; we tuned for smooth validation loss and early–stopping behavior rather than peak speed, consistent with guidance that DA interacts with regularization and schedulers[9]. Indeed training the model with or without the augmentation change the trend of the loss function and the gradient updates request a different LR for each origin of the generations, is not only the number of example that decide the best hyperparameter but it's also where these examples fit the vectorized space once they go through the encoder.
Preview of next chapter. In §5 we will show PCA snapshots of sentence embeddings: light methods (AEDA/STA) expand local neighborhoods; BT and GPT push samples further while remaining inside the class cluster-exactly the behavior we wanted.

4.3.4 Validation protocol and hyperparameter selection

RoBERTa

This subsection covers the short-policy experiments with *RoBERTa-base*, trained *per taxonomy* (e.g., `asset`, `geo_area`,...). Splits are made with grouped hold-out by `cod_sicav` to prevent leakage of the same company across folds. This is because we have found that very often, in almost all cases, a company drafting these documents uses the same investment policy for very similar funds or changes a few words instead of modifying the entire text. This does not benefit the model, which, seeing many similar or identical policies, would tend to understand that a given category is always linked to a given text without interpreting the content[10].

Training settings. We use `AdamW` with no explicit weight decay, a cosine LR scheduler with linear warm-up over 10% of training steps, and a budget of 15 epochs (maximum number after what, also if early stopping doesn't halt the process it will stop anyway) with early stopping (patience = 3). Maximum input length is $L = 512$. [11]

Validation phase (model selection). In the validation script we keep a test fraction of 25% and carve out a validation fraction of 14% from the remaining data, that is, after the validation phase, the train set (grouped by `cod_sicav`). DA is applied *only to the training split.* We scan the following learning-rate grid:

$$\mathrm{lr} \in \{\, 1\cdot 10^{-6},\ 5\cdot 10^{-6},\ 1\cdot 10^{-5},\ 5\cdot 10^{-5},\ 1\cdot 10^{-4}\,\}.$$

For each LR we train up to 15 epochs with early stopping on the validation loss (the "penalty"); the scheduler uses *10% warm-up* of total train steps. We log batch-wise train

[9] The scheduler is mechanism that decide how to update the learning rate during the training process, because it is better to have a dynamic learning rate that changes as the value of the loss function changes in the meanwhile that the gradient loses intensity.

[10] These decisions stem from a process of careful analysis of inputs and results, which showed that the model associated a specific text with a label and, finding it again in the *test set*, classified it correctly only because it had already seen it in the *training set*.

[11] No gradient clipping is applied in the scripts.

loss curves and epoch-wise validation losses and compare runs by: (i) *lowest validation loss* reached, and (ii) qualitative *stability/speed* of the early decrease.

Run-level metrics. Let $\{L_e^{(\mathrm{lr})}\}_{e=0}^{E-1}$ denote the sequence of *validation* losses for a given learning rate lr, evaluated once per epoch e. We define:

$$L_0^{(\mathrm{lr})}\ \text{(initial loss)}, \qquad L_{\min}^{(\mathrm{lr})} = \min_{0\le e<E} L_e^{(\mathrm{lr})}, \qquad e^\star = \arg\min_{0\le e<E} L_e^{(\mathrm{lr})}.$$

To summarize the *early decrease*, we use the average drop per epoch up to the loss minimum:

$$s^{(\mathrm{lr})} = \begin{cases} \dfrac{L_0^{(\mathrm{lr})} - L_{\min}^{(\mathrm{lr})}}{e^\star}, & \text{if } e^\star > 0, \\ +\infty, & \text{if } e^\star = 0 \text{ (minimum at the first epoch).} \end{cases}$$

Decision rule. The decision is to be done per taxonomy and per DA setting, we rank learning rates with the following rule:

$$\widehat{\mathrm{lr}} = \arg\min_{\mathrm{lr}} L_{\min}^{(\mathrm{lr})} \quad \text{(primary criterion).}$$

If multiple candidates are within a small tolerance δ (we used $\delta = 10^{-2}$) of the best $L_{\min}$, we break ties by choosing the one with the largest early–decrease speed:

$$\widehat{\mathrm{lr}} = \arg\max_{\mathrm{lr}\in\mathcal{C}_\delta} s^{(\mathrm{lr})}, \qquad \mathcal{C}_\delta = \{\mathrm{lr} : L_{\min}^{(\mathrm{lr})} \le \min_{\mathrm{lr}'} L_{\min}^{(\mathrm{lr}')} + \delta\}.$$

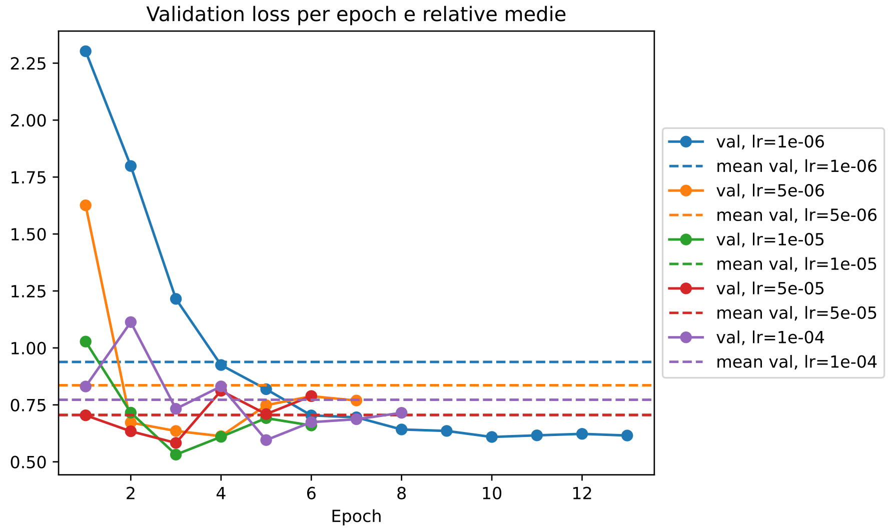
If a further tie occurs, we prefer the smaller LR for stability.

Parallel to the numerical description of the appliance of the LR in the specific contest we always look at the loss descent from the graph, we can see a reported example in 4.4a and 4.4b, that show feature that is impossible to identify from just the numerical view, e.g. if the minimum is reach not during a steady descent but in a oscillation last few gradients updates.

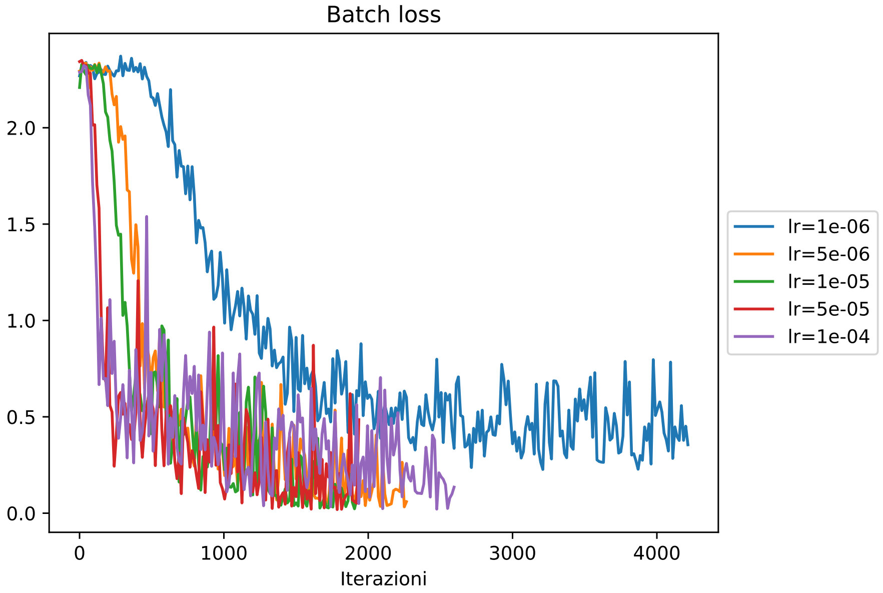
We apply this selection *separately* to the baseline (no DA) and to each DA method, since changes in effective sample size and noise can shift the optimal learning rate.

Final train and test phase. In the final script we keep 25% test and set no validation split (i.e., val = 0%). We reuse the same optimizer and scheduler with 15 epochs, 10% warm-up, and patience = 3, but since there is no validation set, the early stopping monitors the mean training loss per epoch. DA is again applied *only* to the training split, together with the same median balancing to reach the target size T per class. DA method selection were made as explained in Section 4.3.3 and also external from these basic case others where evaluated to test few configurations.

Predictions use *argmax* over class logits. We report standard SKLEARN metrics and, where relevant, a task-specific *soft* metrics derived from label proximities (see Chapter 5).



(a) Validation loss per epoch across learning-rate candidates. Lower is better. Curves show the LR sweep used for model selection; we pick the LR with the minimum validation loss, breaking ties by the fastest early decrease. This is an external check (over data not seen in training set), used to fine tune parameter, loss is evaluated once per epoch. On the graph also the mean loss values per each LR are plotted using the dashed horizontal lines.



(b) Batch-wise training loss for a representative run. Smoother, monotonic decay indicates more stable optimizer dynamics; pronounced oscillations suggest an overly aggressive LR or noise from augmentation/batching. This explain how well the model learns, is plotted to check training stability and optimize dynamic, loss is evaluated every iteration and plotted every 15 of these.

Figure 4.4: Training dynamics for the learning-rate sweep (RoBERTa, short-policy setting).

Longformer

The validation and decision process of the hyperparameters of the *Longformer-base* model we applied, is in many details the same as the process developed for the model with a smaller attention window. In this section, we show the adjustments that were made to the process in order to optimize its performance, guided by the different structure of the two encoder.

Long context and masking. With Longformer we set to encode up to $L = 3072$ tokens to capture dispersed policy evidence, the model could go up to 4096 tokens but measuring our case specific text everyone of them was under this lower threshold. We enable a *global attention* head only on the first token (CLS-like) via a binary `global_attention_mask` whose first position is 1 and all others 0 (all the weight is placed on that token); this lets the model aggregate long-range cues without quadratic cost while still using a 512 token sliding window around the analyzed one (enabling a bigger window is possible but a greater and more performing hardware then our is needed to be able to achieve results in a reasonable time).

Batching under memory constraints. The most important changes is that the per-step batch size is small ($B = 2$ in our runs[12]), so we had to introduce a **gradient accumulation** with $K = 8$ to reach an effective batch $B_{\text{eff}} = B \times K = 16$. Accumulation is implemented by dividing the loss by K and stepping the optimizer/scheduler every K mini-batches.

Scheduler and early stopping. We keep the same optimizer/scheduler family but adapt details to long runs: AdamW with weight decay 0.01 and a cosine schedule with a *warm-up ratio* of 10% of total training steps (as in the previously displayed model). For Longformer we *skip an explicit validation split* due to the longer, with respect to base model, training process, we directly have trained over few LRs and straight from the performance trend we have chosen the right one, in this way we have used less try instead of doing a validation process of 5 LRs per each methodology and each taxonomy that would have cost us unavailable machine time. We then monitor *training* loss, stopping with patience = 3 epochs once the average epoch loss stops improving.

Maximum number of epochs was here upgraded to 20 (never reached by use of early stopping) and as in the previous case, also in this different dataset, training and test split were made not splitting text grouper by `cod_sicav`.

[12] On our GPUs the limits of per-step batch is $B \in \{1,2\}$ even with mixed precision and a small batch means the update uses a high-variance gradient estimate, $\hat{g} = \frac{1}{B}\sum_{i=1}^{B} \nabla \ell_i$, so the parameter increment $\Delta\theta = -\eta\,\hat{g}$ is noisier (and effectively "larger" for a fixed learning rate). To regain the stability of larger batches without extra VRAM, we average K micro-batches before stepping the optimizer (gradient accumulation), which is equivalent to an effective batch $B_{\text{eff}} = B \times K$.

Final configuration and DA selection. The Longformer script executes two DA configurations per label, giving what already learned in the base model, only top performance configuration per DA type were chosen in order to being able to physically compute all the results: (i) STA/AEDA and (ii) a GPT batch generator (4 inputs → 2 outputs), plus a *no-DA* baseline; back-translation is not used in these long runs after seeing that its performance were exceeded. Class balancing follows the *median* rule used elsewhere in the chapter (cut large classes; augment the smallest up to the median size) and is again applied *only* to the training split.

Prediction use always *argmax* over output logits from the decoder.

4.3.5 Test setup

Hardware. Experiments ran on local GPUs (2× NVIDIA T4 (16GB VRAM each one)). RoBERTa used max length $L = 512$ with per-step batch sizes $B \in [8{,}16]$ (depending on class). Longformer used L capped to 3072 with $B \in \{1{,}2\}$ plus gradient accumulation ($K \in \{4{,}8\}$) to reach an effective batch 8-16.

Software. PyTorch ≥ 2.3 with CUDA, HuggingFace `transformers` for models and tokenizers, `datasets` for splits, `scikit-learn` for metrics and stratification. Mixed precision (FP16/bfloat16) was enabled when available to reduce memory.

Evaluation metrics. For validating the DA and being able to choose the best configurations we used the metrics stated in §4.2.3, that was the first analysis needed to be done. Then given imbalance, we report *macro* F1 and per-class F1, in addition to recall, precision and accuracy macro and per-class; to have a better error comprehension we also report confusion matrices. Then printing the loss descend in train is also useful to understand the macro trend of understanding by the model. Decision thresholds are tuned on validation to maximize F1 (then fixed on test).

4.3.6 Final configuration summary

Common settings. We train *one multi-class model per taxonomy* (softmax over all classes in that taxonomy), not one-vs-rest. We use `AdamW` with cosine decay and a linear warm-up of 10% of total steps, and early stopping with patience = 3 *on training loss* (no validation split in the final runs). Predictions are taken by *argmax* over softmax probabilities (no decision-threshold tuning, we just take the most reliable guess of the model). Data augmentation (DA) is applied *only* to the training split. Splits are grouped by `cod_sicav` with a test share of 25%. A fixed seed is used for reproducibility and comparing improvements or deteriorations run by run.

Class balancing. On the training split, we target a reference class size T, defined as the number of examples in the first class whose cardinality exceeds the median class size. If a class has $n_c \geq T$, we downsample it to T; if $n_c < T$, we augment it up to T. For ultra-rare labels, a force drop is applied when $n_c < 4$ (this check is triggered only for the `sector` class and not for other taxonomies).

RoBERTa-base (short policies).

- **Encoder & context:** `roberta-base`, max input length $L = 512$.
- **Optimizer / schedule:** `AdamW` (default weight decay = 0.01); cosine decay with 10% warm-up.
- **Budget & ES:** 15 epochs; early stopping on *training loss* with patience = 3.
- **Learning rates:** chosen after validation set by lowest loss value and *speed*.
- **DA:** evaluated AEDA, EDA, STA, Back-Translation, GPT_1 and GPT_2 singularly and then mixed to obtain optimal configurations by drawing on the qualities of different techniques. Final configurations used and compared: BT/EDA no more used, STA/AEDA mixed up at 50% randomly, each implemented alone and also implemented by joining with GPT_2 paraphrasing augmentation (with batch: 4 inputs → 2 outputs), each one used to reach the median target.
- **Inputs:** fund name is prefixed to the policy text when available, is not given to the augmenter but it is attached to it after the text change/generation (names are carried through the pipeline; STA/AEDA and GPT-augmented rows omit the name in the generation process because some information are summarized in it and losing them could fatal for the classification of the new phrase if is left to the model to invent a new fund name).

Longformer-base (long policies).

- **Encoder & context:** `longformer-base-4096` with $L = 3072$ and global attention on `[CLS]`.
- **Batching under memory limits:** micro-batch B=2 with gradient accumulation K=8 (effective batch 16).
- **Optimizer / schedule:** `AdamW` (weight decay = 0.01); cosine decay with 10% warm-up.
- **Budget & ES:** 20 epochs; early stopping on *training loss* with patience = 3 (typical stop < 20).
- **Learning rates:** chosen after validation set by lowest loss value and *speed*.
- **DA:** as for short-policies used STA/AEDA and the GPT batch method; BT is implemented but not used in these final runs.
- **Inputs:** fund name is prefixed to the policy text but eliminated from the augmentation script input to reattached it after generation (attention mask is applied on `[CLS]` and given in input to global-attention tokens to the model).

Implementation notes (for reproducibility).

- **Seeding & determinism:** fixed global seed for data split, DA operators, and model init; PyTorch/`transformers` deterministic flags enabled where supported.
- **Tokenization & versions:** tokenizer pinned to each checkpoint family; lowercasing and special-token handling follow the default of the chosen encoder.
- **Logging & selection:** we log (i) config (seed, LR, batch, warm-up, weight decay), (ii) split indices, and (iii) per-epoch train/val curves. Best checkpoints are selected by the applicable early-stopping criterion; values are tuned on validation F1 and then fixed for test.
- **Evaluation:** macro-F1 and per-class F1 are reported alongside precision/recall; class support is always shown to contextualize imbalance (in test performance summary cardinality of each class remain untouched).

This chapter has detailed the end-to-end protocol, from data splitting and balancing, through DA, to model-specific training recipes, that we use consistently across taxonomies. In the next chapter we present the quantitative results, including ablations on DA choices and PCA visualizations of embedding shifts induced by our augmentations, with tables and figures summarizing the per-class gains.

Chapter 5

Results

This chapter reports the empirical findings that address the research questions posed in §4.1, using the evaluation metrics defined in §4.2.2 and the data-augmentation (DA) diagnostics introduced in §4.2.3, within the experimental methodology of §4.3. To keep the narrative focused, the main text presents a *representative subset* of taxonomies, chosen from all the ones analyzed, that best illustrate clear improvements, mixed outcomes, and limits. The analysis follows the logic established in Chapter 4: first we verify augmentation behaviour in embedding space, then we establish encoder baselines without DA, afterwards we quantify the impact of each DA family per model on representative taxonomies, and finally we compare encoders, to chose if the trade-off between performance and cost it is worth the upgrade from small to long-context, and consolidate the best configurations per taxonomy defining if the classification task is suitable to the model analyzed and if the textual data augmentation really improves performance on imbalanced real world dataset.

We structure the chapter into four blocks:

- **Augmentation diagnostics:** Metric 1-4 (§4.2.3) and global clustering indices computed on originals+augments, with PCA 2D visualizations for representative taxonomies.

- **Baselines without DA:** Reference performance of *RoBERTa-base* and *Longformer-base* across taxonomies, before any augmentation, this could just demonstrate if a longer-context model truly benefits from it or it just creates “confusion" inside the classifier.

- **DA impact per model:** STA + AEDA and GPT-based paraphrasing are chosen among all the possibility thanks to the analysis that will be made in previous section; report absolute and relative Δ macro-F1 and changes in minority recall against the baselines. Analysis made parallel per each of the two model.

- **Encoder comparison and best configurations:** When long context helps; final per-taxonomy decisions (encoder + DA + key hyperparameters).

5.1 Augmentation diagnostics

We begin by characterizing how each DA family behaves in the sentence-embedding space before using it in training. The analysis uses the same corpus and grouping protocol described in §4.3 (companies kept separated across splits via `cod_sicav`), and it covers all eight target taxonomies (*asset, dimension, geo_area, issuer, maturity, rating, sector, style*). For diagnostics we embed texts with `SentenceTransformer all-MiniLM-L6-v2` that project text in 768 dimensions vector. Per class, we sample n=5 originals and generate k=5 augments for each original for every DA family considered here (AEDA, EDA, back-translation, GPT paraphrases), then study geometry with cosine similarity.

We report the four DA metrics defined in §4.2.3: Metric0 (orig → aug drift), Metric1 (intra-class similarity), Metric2 vs. Metric3 (inter-class similarity on originals vs. augments) from which we derive M1-4, together with global clustering indices computed on the combined set (Silhouette with cosine, Davies–Bouldin, Calinski–Harabasz). In line with Chapter 4, θ=0.8 (or a cutoff of $1-\theta = 0.2$) is the semantic reference for acceptance; here we primarily show the *raw* geometric effects and summarize acceptance rates against θ to connect with later training.

5.1.1 Setup and scope

To keep the narrative compact, the main chapter includes *one* overview figure and *one* summary table: (i) a PCA panel (2D) comparing originals vs. augments for three *representative* taxonomies (e.g., *asset*, *geo_area*, *sector*); (ii) an aggregated table of M1-4 and clustering indices averaged across taxonomies for each DA family. We focus the analysis on these three representative taxonomies, chosen because their very different cardinalities (K=10, K=4, K=26) highlight distinct behaviors under augmentation and classification. The five remaining targets (*dimension*, *issuer*, *maturity*, *rating*, *style*) are are not displayed here but they just replicate all the behaviours that these three representative ones show. The aim is to quantify drift, cohesion, and separability introduced by each DA family, setting expectations for the training results that follow and choosing if one or more of this possible augmentation choices doesn't improve it.

5.1.2 Similarity metrics summary (per class)

Taxonomy 1 (asset)

Table 5.1: Aggregated similarity diagnostics for **asset** (per-class aggregation; entries are *mean / median / p90*). M1: cosine drift ↓ (derived as $1-\overline{\text{sim}}$ from the Metric0 matrix); M2: Δ compactness ↓ (with_aug dist − orig dist; more negative = tighter); M3: inter-class separation ↑; M4: tail risk on augments (lower is better) ↓.

DA family	**M1: drift ↓**	**M2: Δ compactness ↓**	**M3: centroid dist ↑**	**M4: leakage ↓**
AEDA	0.045 / 0.040 / 0.094	−0.059 / −0.062 / −0.046	0.172 / 0.156 / 0.252	5.3% / 1.7% / **17.0%**
EDA	0.088 / 0.090 / 0.155	−0.034 / −0.040 / 0.006	0.164 / 0.152 / 0.234	6.0% / **0.0%** / 23.7%
BT	**0.026** / **0.018** / **0.058**	−0.066 / −0.067 / **−0.057**	**0.179** / **0.161** / **0.263**	**5.0%** / **0.0%** / 20.0%
GPT	0.039 / 0.039 / 0.069	**−0.071** / **−0.071** / −0.051	0.172 / 0.159 / 0.241	**5.0%** / **0.0%** / 21.0%

Takeaways (asset). Among all augmentation families, back-translation (BT) demonstrates the most balanced behaviour, producing the lowest average drift and the strongest tightening of within-class structure. GPT-based paraphrasing performs very similarly, slightly improving class cohesion while maintaining a safe distance from the originals. AEDA offers moderate results on both drift and compactness, showing that it can enhance class density without introducing major deviations. In contrast, EDA reveals the highest level of drift and even a positive *p90* Δ compactness for some classes, indicating occasional class spreading and a higher risk of semantic shift. Leakage patterns confirm this trend, with EDA displaying the heaviest tail, while BT and GPT remain the most controlled. Overall, the centroid separations remain comparable across methods, with BT preserving the largest average inter-class margins, reinforcing its role as the most stable and semantically consistent augmentation strategy.

Taxonomy 2 (geo_area)

Table 5.2: Aggregated similarity diagnostics for **geo_area** (per-class aggregation; entries are *mean / median / p90*). M1: cosine drift (orig→aug); M2: Δ compactness (negative is tighter); M3: centroid separation; M4: leakage (% aug with nearest *foreign* centroid).

DA family	**M1: drift ↓**	**M2: Δ compactness ↓**	**M3: centroid dist ↑**	**M4: leakage ↓**
AEDA	0.038 / 0.042 / 0.085	−0.073 / −0.071 / −0.044	0.199 / 0.202 / 0.217	6.7% / 3.3% / 16.0%
EDA	0.076 / 0.070 / 0.157	−0.059 / −0.061 / −0.035	0.194 / 0.195 / 0.216	7.5% / 5.0% / 17.0%
BT	**0.025** / **0.020** / **0.064**	−0.067 / −0.067 / −0.044	**0.201** / **0.202** / **0.218**	6.7% / 3.3% / 16.0%
GPT	0.044 / 0.036 / 0.200	**−0.078** / **−0.077** / **−0.058**	0.195 / 0.196 / 0.214	**5.0%** / **0.0%** / **14.0%**

Takeaways (geo_area). Back-translation (BT) yields the lowest average drift and small tail (*p90*), confirming a very safe transformation. GPT paraphrasing tightens classes the most (largest negative Δ compactness) and maintains low leakage, though its drift shows a heavier upper tail for one class. AEDA lands in the middle on both drift and compactness. EDA remains the noisiest option (highest drift and leakage). Centroid separations are comparable across methods, with BT preserving the largest average margins, in line with the *asset* trends.

Taxonomy 3 (sector)

Table 5.3: Aggregated similarity diagnostics for **sector** (per-class aggregation; entries are *mean / median / p90*). M1: cosine drift (orig→aug); M2: Δ compactness (negative is tighter); M3: centroid separation; M4: leakage (% aug with nearest *foreign* centroid).

DA family	**M1: drift ↓**	**M2: Δ compactness ↓**	**M3: centroid dist ↑**	**M4: leakage ↓**
AEDA	0.042 / 0.041 / 0.109	−0.049 / −0.056 / −0.012	**0.247** / **0.243** / 0.306	9.5% / 1.7% / 25.0%
EDA	0.081 / 0.076 / 0.169	−0.036 / −0.043 / 0.011	0.238 / 0.230 / 0.299	10.8% / 3.3% / 28.0%
BT	**0.030** / **0.023** / **0.070**	−0.061 / −0.064 / −0.049	0.244 / 0.240 / **0.309**	8.3% / **0.0%** / 23.0%
GPT	0.039 / 0.037 / 0.075	**−0.067** / **−0.069** / **−0.055**	0.242 / 0.239 / 0.307	**7.5%** / **0.0%** / **21.0%**

Takeaways (sector). Back-translation (BT) achieves the lowest drift and strong intra-class tightening, confirming its safety and stability. GPT paraphrasing shows a very similar pattern, with slightly looser but still well-behaved clusters and the lowest leakage overall. AEDA remains moderate on all metrics, while EDA once again exhibits the highest drift and leakage. Centroid distances are largely comparable across methods, with BT maintaining the widest margins. Overall, both BT and GPT stand out as the most consistent and semantically controlled DA families, mirroring the trends observed for *asset* and *geo_area*.

5.1.3 Global clustering indices

Taxonomy 1 (asset)

Table 5.4: Global clustering indices on the combined set for **asset** (originals+augments), asset taxonomy. Silhouette (cosine), DB, CH.

DA family	Silhouette ↑	DB ↓	CH ↑
AEDA	0.109	2.909	12.739
EDA	0.106	3.116	10.839
BT	0.115	2.885	**13.627**
GPT	**0.122**	**2.814**	13.470

Takeaways (asset). Considering the combined set of originals and augmented texts, GPT achieves the best overall cohesion–separation balance, showing the highest Silhouette score and the lowest Davies–Bouldin index. Back-translation (BT) performs very closely, displaying the best Calinski–Harabasz score and confirming its ability to preserve compact yet well-separated clusters. AEDA lies in an intermediate position, maintaining reasonable structure without introducing excessive dispersion, while EDA remains the least stable method across all three indices. These patterns are fully consistent with the similarity diagnostics in Table 5.1: methods that induce lower drift and stronger intra-class cohesion (such as GPT and BT) also produce cleaner, more interpretable clusters. In practical terms, this suggests that safer DA families tend to preserve the semantic integrity of classes, leading to more robust downstream classification performance in the presence of class imbalance.

Taxonomy 2 (geo_area)

Takeaways (geo_area). On the combined set, AEDA attains the best Silhouette/DB, GPT the best CH, and BT sits very close on all three. EDA is worst overall. The picture is consistent with the similarity diagnostics: safer, lower-drift augmentations (BT/GPT) deliver cleaner clusters, while EDA degrades structure the most.

Table 5.5: Global clustering indices on the combined set (originals+augments), **geo_area** taxonomy. Silhouette (cosine), DB, CH.

DA family	Silhouette ↑	DB ↓	CH ↑
AEDA	**0.181**	**2.676**	12.561
EDA	0.163	2.840	11.688
BT	0.177	2.683	12.416
GPT	0.178	2.704	**12.660**

Taxonomy 3 (sector)

Table 5.6: Global clustering indices on the combined set (originals+augments), **sector** taxonomy. Silhouette (cosine), DB, CH.

DA family	Silhouette ↑	DB ↓	CH ↑
AEDA	0.139	2.684	14.703
EDA	0.133	2.857	13.274
BT	0.142	2.650	**15.011**
GPT	**0.147**	**2.627**	14.963

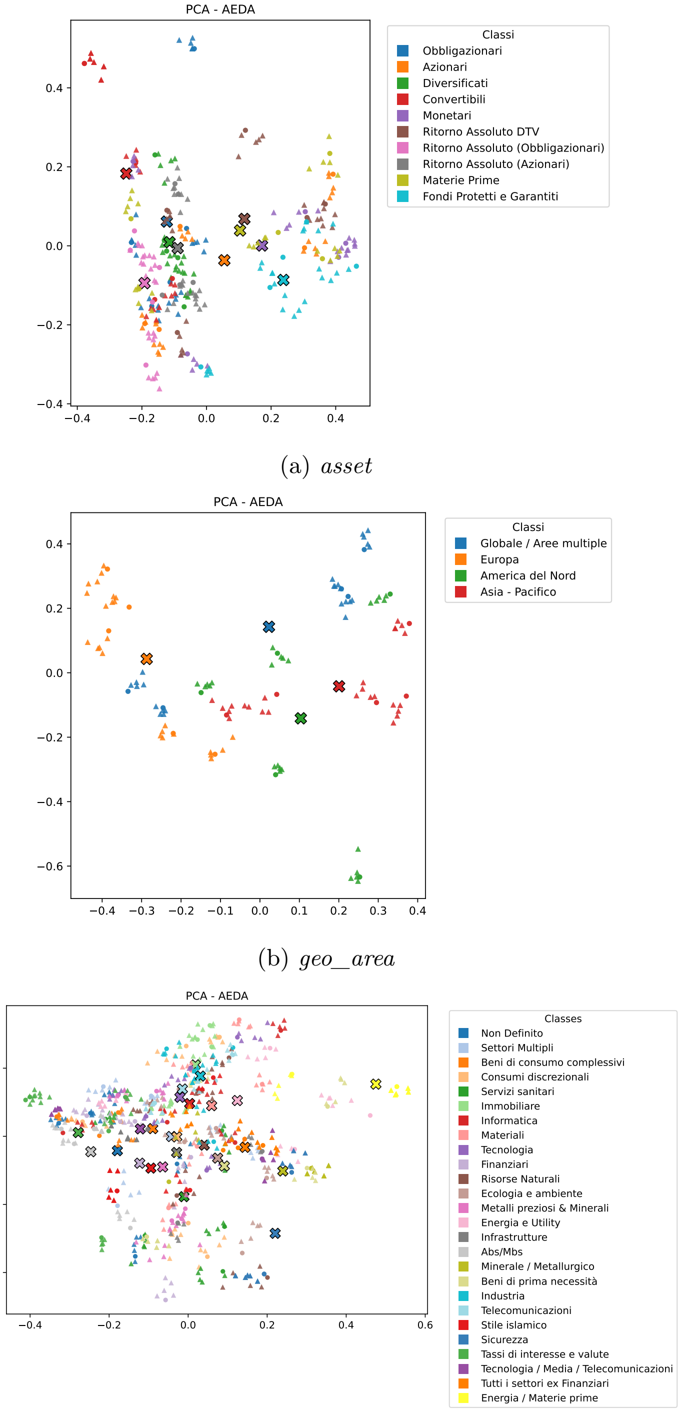
Takeaways (sector). On the combined set, GPT attains the best overall cohesion–separation balance, yielding the highest Silhouette and lowest DB. BT performs nearly identically and achieves the best CH value, confirming its ability to form compact yet distinct clusters. AEDA remains in an intermediate position, while EDA shows the weakest clustering behaviour. These results align with the similarity diagnostics, further supporting BT and GPT as the most reliable and semantically consistent DA strategies.

5.1.4 PCA maps

We visualize how each DA family perturbs the embedding geometry via 2D PCA. For each taxonomy, PCA is fit on the *original* embeddings and then used to project both originals and augments. In the main text we contrast two complementary families: AEDA (lightweight, token-level) and GPT (semantic paraphrasing), on the three representative taxonomies (*asset*, *geo_area*, *sector*).

Scatter panels (*orig+aug*) show whether augments remain near their source clusters and whether within-class spread tightens. Consistently with Tables 5.1–5.3 and §5.1.3, GPT tends to yield tighter, well-aligned clouds with low leakage, while AEDA produces moderate movement that usually preserves class neighborhoods and margins.

AEDA: orig+aug scatter



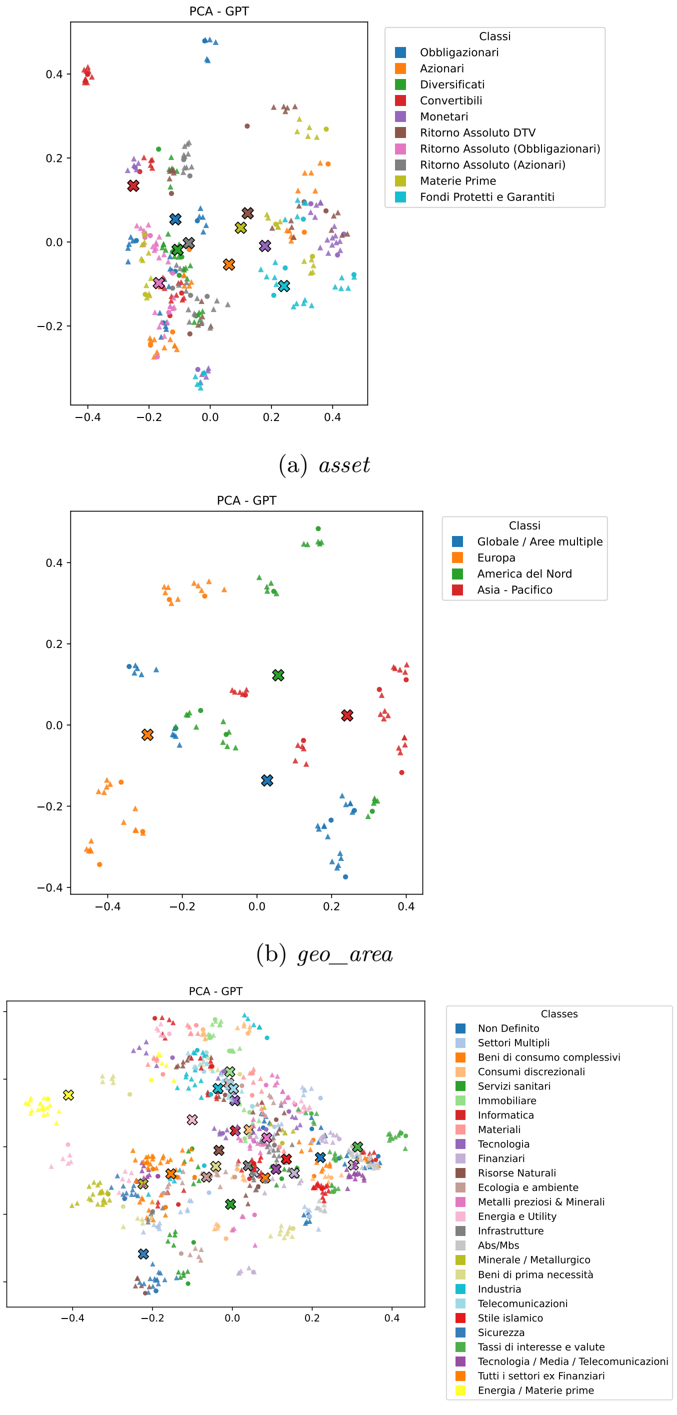
(a) *asset*

(b) *geo_area*

(c) *sector*

Figure 5.1: PCA maps on sentence embeddings with **AEDA** (PCA fit on originals, then applied to originals+augments). Rows show *asset*, *geo_area*, *sector*; ○ are original embeddings, △ augmented texts, and crosses mark class centroids.

GPT paraphrases: orig+aug scatter



(a) *asset*

(b) *geo_area*

(c) *sector*

Figure 5.2: PCA maps on sentence embeddings with **GPT paraphrases** (PCA fit on originals, then applied to originals+augments). Rows show *asset*, *geo_area*, *sector*; ○ are original embeddings, △ augmented texts, and crosses mark class centroids.

Two qualitative patterns emerge from Figs. 5.1 and 5.2. First, as the label space becomes finer-grained (e.g., *sector* vs. *asset*), the margins between class centroids visibly shrink; this anticipates the expected drop in attainable multi-class accuracy as granularity (and number of classes) increases, beyond class-imbalance effects. Second, augmented points (△) concentrate around their source embeddings (○): GPT paraphrases tend to form tighter, small clusters that are roughly round and evenly spread in all directions (see Fig. 5.2b), whereas AEDA induces slightly broader but still local halos (see Fig. 5.1b). These visual trends match the metrics reported in §5.1.2–5.1.3: with GPT/BT, augments stay close to their sources (M1 low), class clouds become slightly tighter without collapsing (M2 mildly negative), the distances between class centroids remain roughly unchanged (M3 small change), and only a few augments are closer to a foreign centroid (M4 low).

Because PCA is a linear 2D projection, absolute distances are not preserved and axes are arbitrary up to rotation/scale, meanwhile point separation on the plot are shrunk and not preserved due to the 766 dimension drop; we therefore treat these plots as *qualitative* evidence and rely on the metric suite in §5.1.2–5.1.3 for validation. The figures offer visual intuition; the metrics provide the faithful, high-dimensional measurement.

5.1.5 Cross-taxonomy recap and DA selection

Across *asset*, *geo_area*, and *sector* there is *no single* DA family that dominates every criterion. GPT paraphrasing consistently yields the tightest classes (most negative Δ compactness) with low leakage and strong global cohesion-separation (best Silhouette/DB in two taxonomies), while BT shows the smallest raw drift and slightly larger centroid margins. AEDA sits between these two: it increases local diversity with moderate drift, usually preserves margins, and is competitive on global indices in *geo_area*. EDA is systematically the weakest, with the largest drift and leakage tails.

This evidence suggests a *complementary* DA policy rather than a single-method choice. In practice we adopt a mixed recipe that exploits the strengths of both GPT paraphrasing (semantic safety, intra-class tightening) and Noising-based methods (lightweight lexical diversity): (i) keep the semantic filter at θ=0.8; (ii) for small classes cap the number of GPT augments (safer, low-leakage samples) to increase the numerosity but also the robustness thanks to *noising methods*; (iii) for majority classes the GPT cap is not filled, a small amount of new example needs to be generated and these must be as similar as possible so as not to affect the margins of the class; (iv) add a filter to the generations and modifications of the original texts through noising techniques by using word-aware method (STA).

In short, while GPT alone often provides the cleanest geometry, pairing it with *noising methods* yields a better balance between *semantic safety* and *useful diversity*. During training we still apply all the different methodology to see if our theoretical forecasts perform well over realistic application.

5.2 Baselines without DA

The goal of this section is to establish the encoder capabilities *without* textual augmentation. Each configuration (baseline vs. DA family) is assigned its own learning-rate via the validation protocol in §4.3.4: we scan a small LR grid, select the run with the minimum validation loss, and break ties by the fastest early decrease to favor stable dynamics.[1] In order to temper the heaviest class skew while preserving information, we cap over-represented classes at the median frequency, leaving the minority tail untouched.

Given the obvious difficulties associated with the high number of classes in *sector* this was fine for analyzing the performance and functioning of the DA, but it is not the most suitable taxonomy for demonstrating how the classifier works, so we will keep the others taxonomies (*asset*,*geo_area*) but we will switch *sector* for *dimension* that also has the soft-view for errors.

As a reminder, under imbalance, weighted-averaged may look high while macro-averaged metrics expose weak minority recall due to the equal weights assigned to each class independently from its numerosity; we therefore report macro-F1 alongside overall accuracy having this caveat in mind.

5.2.1 RoBERTa-base (short context)

Taxonomy 1 (asset)

Table 5.7 reports the baseline on the *asset* taxonomy ("hard" view) together with the *soft_asset* aggregation. In the fine-grained view, overall accuracy is 0.833 and macro-F1 is 0.647; the weighted-F1 reaches 0.842. In the coarser view, macro-F1 rises to 0.707 and micro-F1 to 0.893, with weighted-F1 at 0.895. These values come from the latest asset run with no DA.

It excel where we have high-support, where semantically distinctive classes are strong: *Azionari* (F1 0.966, n=1666), *Materie Prime* (F1 0.957, n=44), *Convertibili* (F1 0.909, n=29). It struggle where we have small or heterogeneous categories, these show lower F1: *Fondi Protetti e Garantiti* (F1 0.143, n=6), *Monetari* (F1 0.441, n=33), and the *Ritorno Assoluto* variants (*Obbligazionari* F1 0.488, n=197; *Azionari* F1 0.404, n=35; *DTV* F1 0.543, n=360). This pattern is coherent with class imbalance: macro-F1 exposes minority-recall weaknesses even when accuracy remains high. Those cited classes have boundary less remarked and can easily cause confusion in the classifier, which will end up obtaining a probability vector for each class that is not biased towards any one in particular but with several peaks across multiple valid options, indeed the *soft* view show a remarkable increase in F1 due to the fact that this type of error are not counted (if the peaks in probability distribution are coherent with classes close to the right one).

Per-class highlights. Table 5.8 collects the key per-class numbers and makes clear *how* performance fails or succeeds: precision (false-positive control), recall (false-negative control), and their harmonic mean (F1). "Support" n indicates stability of the estimate (very small $n \Rightarrow$ high variance).

The support shows the real world fraction, because only in the training set the median cut is applied. In the test set the number of example where the model have to be evaluated is a fraction of the total that maintain the true class proportions. *Precision* measures

[1] See details and decision rule in §4.3.4.

Table 5.7: RoBERTa *asset* baseline. Hard view reports macro-F1, weighted-F1, and accuracy; soft view reports macro-F1, weighted-F1, and micro-F1. All numbers on the test split; N is the number of samples evaluated.

View	N	**Macro-F1 ↑**	**Weighted-F1 ↑**	**Accuracy ↑**	**Micro-F1 ↑**
Asset	4261	0.647	0.842	0.833	–
Soft_asset	4261	0.707	0.895	–	0.893

Table 5.8: RoBERTa *asset* — class-wise precision/recall/F1 (rounded to 3 decimals).

Class	**Support** n	**Precision ↑**	**Recall ↑**	**F1 ↑**
Azionari	1666	0.974	0.958	0.966
Obbligazionari	1415	0.946	0.810	0.873
Diversificati	476	0.685	0.809	0.742
Ritorno Assoluto DTV	360	0.611	0.489	0.543
Ritorno Assoluto (Obbligazionari)	197	0.387	0.660	0.488
Ritorno Assoluto (Azionari)	35	0.322	0.543	0.404
Monetari	33	0.306	0.788	0.441
Convertibili	29	0.962	0.862	0.909
Materie Prime	44	0.917	1.000	0.957
Fondi Protetti e Garantiti	6	0.125	0.167	0.143

purity of the predictions for a class (low precision ⇒ many false positives *into* that class). *Recall* measures coverage of a true class (low recall ⇒ many false negatives, i.e., true items of that class predicted as something else). *F1* balances both; with very small n, F1 can swing widely between runs.

Azionari shows balanced, high precision and recall, indicating clear lexical/semantic cues and well-separated decision boundaries. *Obbligazionari* has very high precision but lower recall: the model is conservative when assigning this label (few false positives) but misses a non-trivial portion of true *Obbligazionari*, which likely get pulled toward neighboring, high-support classes (e.g., *Diversificati*/*RA DTV*). *Materie Prime* and *Convertibili* achieve high F1 despite small n, suggesting distinctive terminology; however, the low support makes these scores volatile across seeds/splits. At the bottom, *Ritorno Assoluto* variants, *Monetari*, and *Fondi Protetti e Garantiti* have low F1s consistent with heterogeneous policies and boundary ambiguity. In these cases the model's probability vector often spreads its mass across adjacent, semantically plausible classes; the *soft* view therefore improves macro-F1 because near-misses within the aggregated family are no longer counted as errors.

Taxonomy 2 (geo_area)

Table 5.9 reports the baseline on the *geo_area* taxonomy. Overall accuracy is 0.904, with macro-F1 at 0.898 and weighted-F1 at 0.905 on the test split (N=4258). No *soft* aggregation is defined for this taxonomy, so micro-F1 is omitted.

Table 5.9: RoBERTa *geo_area* baseline. Macro-F1 and weighted-F1 are reported together with accuracy; micro-F1 is not applicable (no soft view). All numbers refer to the test split.

View	N	**Macro-F1** ↑	**Weighted-F1** ↑	**Accuracy** ↑	**Micro-F1**
Geo_area	4258	0.898	0.905	0.904	–

Per-class highlights. Table 5.10 summarizes precision (false-positive control), recall (false-negative control), and F1 for each region, plus a qualitative "error profile." "Support" n reflects real-world prevalence (the test split preserves true proportions).

Table 5.10: RoBERTa *geo_area* — class-wise precision/recall/F1 and qualitative error profile (rounded to 3 decimals).

Class	**Support** n	**Precision** ↑	**Recall** ↑	**F1** ↑
Globale / Aree multiple	2577	0.942	0.899	0.920
Europa	808	0.808	0.908	0.855
America del Nord	444	0.823	0.867	0.844
Asia – Pacifico	429	0.981	0.960	0.971

With only four, semantically distinct regions, performance is uniformly high. *Asia–Pacifico* is the cleanest class (very high precision and recall), suggesting distinctive lexical cues. *Globale/Aree multiple* shows slightly lower recall than precision, meaning the model sometimes misses these cases, which makes sense because these funds often have broad investment scopes that can also sound like regional ones. For *Europa* and *America del Nord*, recall exceeds precision: these labels are sometimes over-assigned. The narrow macro vs. weighted gap (0.898 vs. 0.905) indicates that strong head-class performance does not mask weaknesses in the smaller regions; the tail classes are also well learned overall and the imbalance is not that much of a problem with so few classes.

Taxonomy 3 (dimension)

Table 5.11 reports the baseline on the *dimension* taxonomy ("hard" view) together with the *soft_dimension* aggregation. In the fine-grained view (N=4492), overall accuracy is 0.634, macro-F1 is 0.442, and weighted-F1 is 0.658. In the soft aggregation, macro-F1 rises to 0.710, weighted-F1 to 0.842, and micro-F1 to 0.829. These values come from the latest *dimension* run with no DA.

It excel in the dominant *Non Definito (ND)* class, that is precise and reasonably well recalled (F1 0.759, n=2883). *Non Applicabile* attains high recall (0.817) although with lower precision (0.395), yielding F1 0.532[2]. It struggle where overlapping classes shows

[2] The two classes *Non Applicabile* and *Non Definito* seems similar but two precisely different concepts, the first one intend that the fund invest in asset that are not definable over their capitalization like bonds, while *Non Definito* regard policies that are not definite in any of the classes from the domain expert that has drafted the classification over which we are training the model, this could be derived from the absence of information inside the policy.

due to really small embedding distances and blurred edges: *Large Cap* (F1 0.236, n=220), *Mid & Small Cap* (F1 0.300, n=84), and *Mid Cap* (F1 0.323, n=10). These categories exhibit boundary ambiguity and small support; the soft view mitigates this by forgiving near-misses within the same family (e.g., *Large* vs. *Large & Mid*).

Table 5.11: RoBERTa *dimension* baseline. Hard view reports macro-F1, weighted-F1, and accuracy; soft view reports macro-F1, weighted-F1, and micro-F1. All numbers on the test split; N is the number of samples evaluated.

View	N	**Macro-F1 ↑**	**Weighted-F1 ↑**	**Accuracy ↑**	**Micro-F1 ↑**
Dimension	4492	0.442	0.658	0.634	–
Soft_dimension	4492	0.710	0.842	–	0.829

Table 5.12: RoBERTa *dimension* — class-wise precision/recall/F1 (rounded to 3 decimals).

Class	**Support** n	**Precision ↑**	**Recall ↑**	**F1 ↑**
Non Definito	2883	0.876	0.670	0.759
Large & Mid Cap	1048	0.481	0.600	0.534
Large Cap	220	0.190	0.314	0.236
Non Applicabile	197	0.395	0.817	0.532
Mid & Small Cap	84	0.238	0.405	0.300
Small Cap	50	0.452	0.380	0.413
Mid Cap	10	0.238	0.500	0.323

Per-class highlights. *ND* is learned with strong precision and fair recall, consistent with broad language and head-class support. Capitalization bucket confusion is the main failure mode: *Large Cap*, *Large & Mid Cap*, *Mid & Small Cap*, and *Mid Cap* share overlapping descriptors, so recall improves at the expense of precision in some buckets (e.g., *Large & Mid Cap* recall 0.600 vs. precision 0.481). In the soft view, near-miss mass is recaptured within the aggregated families: for example, *Mid Cap* achieves F1 0.952, *Large* 0.665, *Mid & Small Cap* 0.545, and *Small Cap* 0.714, while *ND* reaches 0.896 (soft metrics), confirming that most errors are within-family rather than cross-family. Due to the great part of the classes overlap between themselves not counting "first-degree" error makes a big difference, the model succeeds in generally understand the size of the assets in which the fund invests but cannot identify them with extreme precision.

Summary (RoBERTa baseline)

Across the three taxonomies, performance tracks label granularity and boundary clarity: *geo_area* is high and stable (macro-F1 ≈ 0.90), *asset* is mid-range (macro-F1 ≈ 0.63), while *dimension* is the hardest (macro-F1 ≈ 0.44). Weighted-F1 consistently exceeds macro-F1, indicating that head classes are well learned and most of the deficit comes from the tail, this reinforces our idea of implementing DA. Soft aggregations substantially lift scores (*asset*: 0.63→0.70; *dimension*: 0.44→0.71; with micro-F1 of 0.887 and 0.829, respectively), showing that many hard errors are actually *near-misses* within the same semantic family rather than cross-family confusions.

Overall, the encoder captures general semantics reliably but struggles to place fine boundaries under imbalance and label ambiguity. This motivates our strategies in the next sections that preserve semantic fidelity while injecting variety where the baseline exhibits FN-heavy or FP-heavy behavior.[3]

5.2.2 Longformer-base (long context)

Model selection [CLS]

Here in the section we have analyzed the Longformer with active `[CLS]` (global attention on `[CLS]` in the context window) that is because in the DA experiments we compared pooling strategies for Longformer and found that this configuration consistently delivered the best post-augmentation scores in both the hard and soft views. For this reason we adopt Longformer + `[CLS]` in all DA results reported later. Curiously, in the no-DA baseline the same `[CLS]` variant underperformed the base one (despite identical optimization settings); we could not isolate a single cause for this discrepancy and simply record it as an empirical finding (the model should be capable of seeing more information, instead it classify worse).

Taxonomy 1 (asset)

Table 5.13 reports the baseline on the *asset* taxonomy ("hard" view) together with the *soft_asset* aggregation. On the test split of this run (N=1880), overall accuracy is 0.800 and macro-F1 is 0.561; the weighted-F1 reaches 0.818. In the coarser view, macro-F1 rises to 0.635 and micro-F1 to 0.862, with weighted-F1 at 0.866. These values come from the latest *asset* run with no DA (Longformer-base, long-policy setting).

It excels in high-support, where semantically distinctive classes remain strong: *Azionari* (F1 0.940, n=883), *Obbligazionari* (F1 0.839, n=539). Among minority classes, *Convertibili* attains F1 0.750 (n=14) and *Materie Prime* F1 0.667 (n=47) with very high recall (0.957). It instead struggles in heterogeneous/overlapping categories that show lower F1: *Ritorno Assoluto* variants (DTV F1 0.485, n=133; *Obbligazionari* F1 0.361, n=51; *Azionari* F1 0.250, n=20) and *Fondi Protetti e Garantiti* (F1 0.000, n=6, so any of the 6 test example is classified correctly). *Monetari* is FP-heavy (precision 0.488 vs. recall 0.911; F1 0.636), while *Obbligazionari* is FN-heavy (precision 0.948 vs. recall 0.751). Overall, macro-F1 exposes tail weaknesses even when accuracy is high; the *soft* view recovers many near-misses within families.

[3] *FP-heavy*: for a given class, the model produces many false positives—i.e., it predicts the class on items whose true label is different—indicating an overly permissive decision boundary (typically higher recall, lower precision). *FN-heavy*: the model produces many false negatives—i.e., it fails to predict the class on items that truly belong to it—indicating an overly conservative boundary (typically lower recall, higher precision).

Table 5.13: Longformer *asset* baseline. Hard view reports macro-F1, weighted-F1, and accuracy; soft view reports macro-F1, weighted-F1, and micro-F1. All numbers on the test split; N is the number of samples evaluated.

View	N	**Macro-F1** ↑	**Weighted-F1** ↑	**Accuracy** ↑	**Micro-F1** ↑
Asset	1880	0.561	0.818	0.800	–
Soft_asset	1880	0.635	0.866	–	0.862

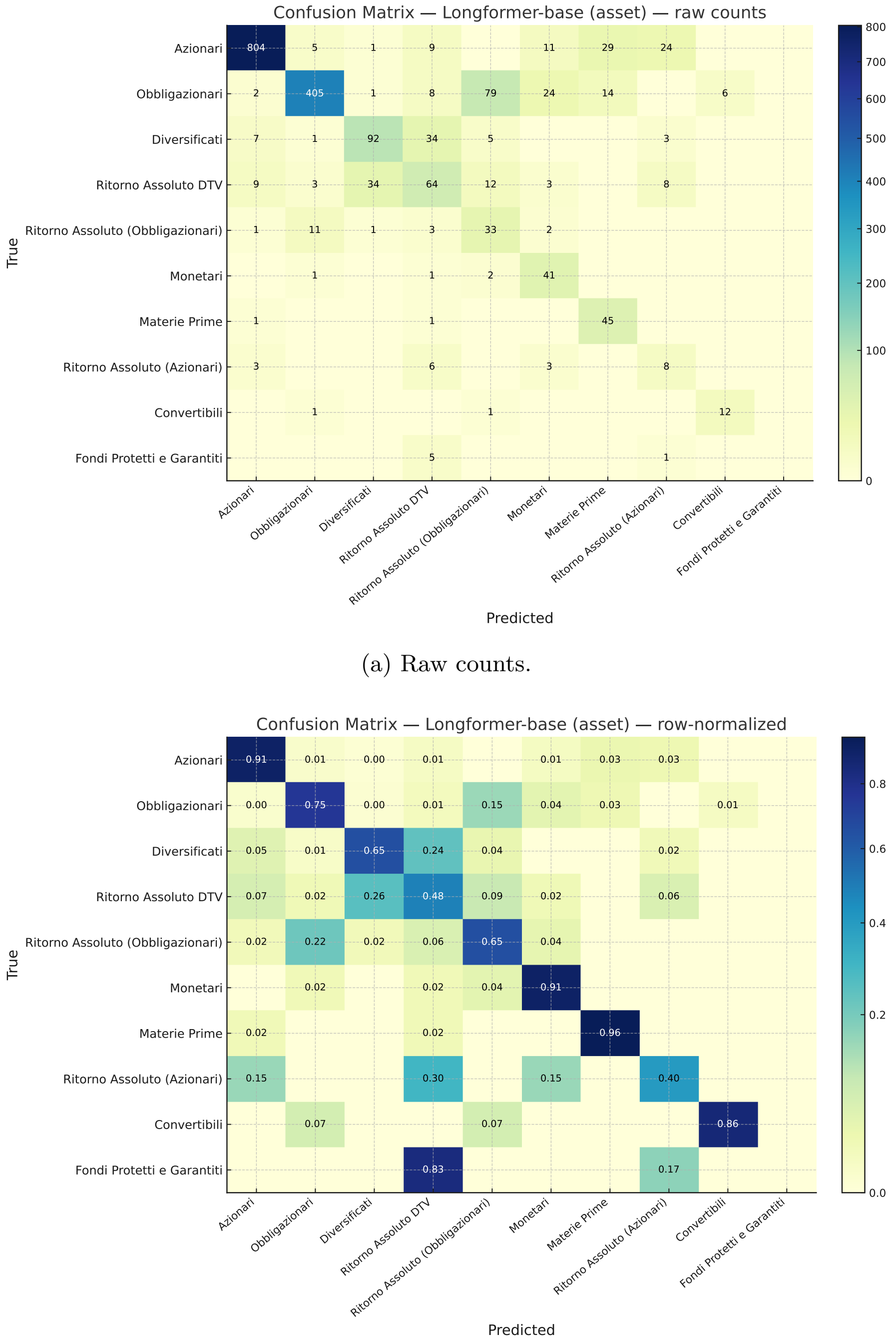
Per-class highlights. Figure 5.3 collects the key per-class numbers and makes clear *how* performance fails or succeeds. "Support" n indicates stability of the estimate, it reflects real-world prevalence (the test split preserves true proportions).

Error analysis via confusion matrix. From the confusion matrix not normalized (Fig. 5.3a) the imbalance effect could be seen clearly, how the low number of errors made in majority classes can, indeed, affect the overall performance of the taxonomy; while the row–normalized CM (Fig. 5.3b) clarifies the class behavior. *Azionari* is clean and well separated: ∼0.91 on the diagonal and its largest off–diagonal goes to *Materie Prime* (∼3.3%), coherently with its high F1 (0.940). *Obbligazionari* shows conservative assignment: only ∼0.75 of true items are kept on–label, with the main loss into *Ritorno Assoluto (Obbligazionari)* (∼14.7%) and smaller flows to *Monetari* (∼4.5%) and *RA DTV* (∼1.5%), matching its high precision / lower recall profile (F1 0.839). The *Diversificati* vs. *RA DTV* pair exhibits the most symmetric near–miss pattern: true *Diversificati* split ∼0.65 on–label and ∼0.24 to *RA DTV*, while true *RA DTV* keep ∼0.48 on–label and send ∼0.26 to *Diversificati*; this explains their mid/low F1s (0.679 and 0.485). *Monetari* attains high recall on–row (∼0.91), but the column collects non–trivial inflow (e.g., from *Obbligazionari*), which depresses precision and yields F1 0.636. Despite small n, *Materie Prime* is almost perfectly recalled (∼0.96 on–row) and only slightly leaks to *Azionari* (∼2.1%); its F1 (0.667) is limited by false positives into the class (column–wise inflow). *Convertibili* remains fairly isolated (recall ∼0.86, minor confusion with *Obbligazionari* ∼7%), consistent with F1 0.750. Among the tails, *RA (Azionari)* is notably unstable: only ∼0.40 stays on–label, with a large drift to *RA DTV* (∼30%), yielding F1 0.250. Finally, *Fondi Protetti e Garantiti* (very small n) collapses into *RA DTV* (∼83%), producing zeros in precision and recall and hurting macro–F1. Overall, the CM confirms that most errors are near–misses; due to that we have it shown on data by looking at the soft aggregation that recovers a substantial share of performance loss of the "hard" analysis (macro–F1 0.561 → 0.635; micro–F1 0.862).

Taxonomy 2 (geo_area)

Table 5.14 reports the baseline on the *geo_area* taxonomy for Longformer-base with no DA. Overall accuracy is 0.916, with macro-F1 0.927 and weighted-F1 **0.916** (N=1868). No *soft* aggregation is defined for this taxonomy, so micro-F1 is omitted.

Error analysis via confusion matrix. Figures 5.5a–5.5b show that the dominant error is a mild *overlap with "Globale/Aree multiple"*: true *Europa* → *Globale* at 7.2%



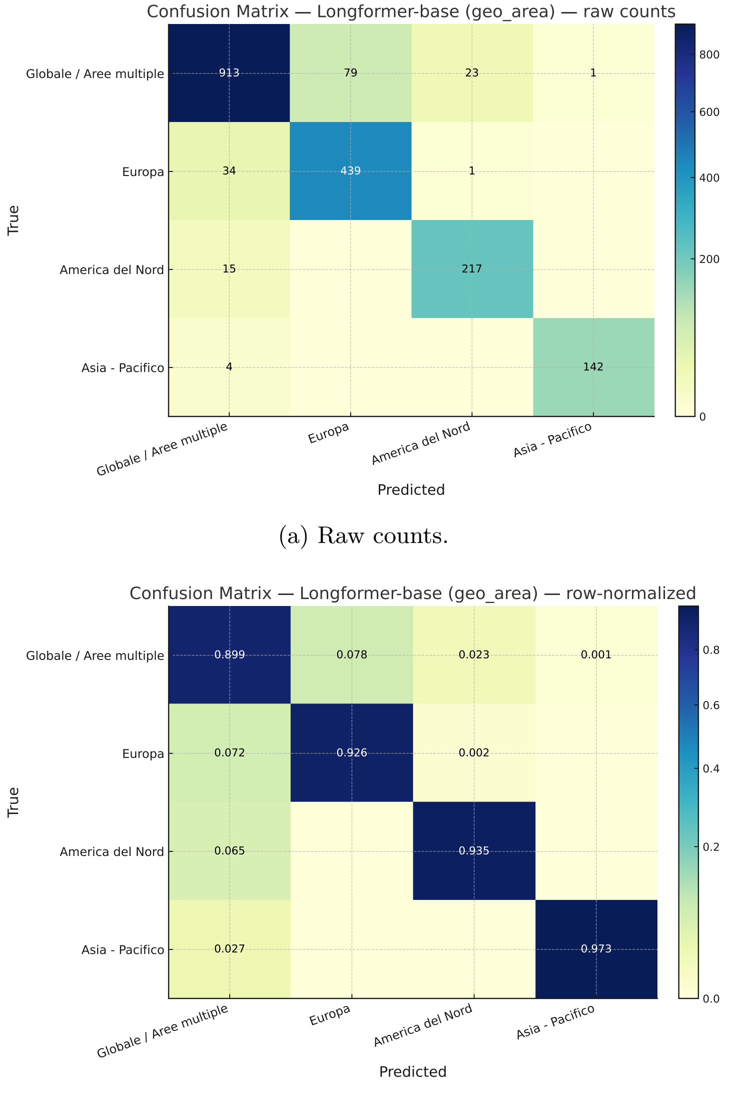
(a) Raw counts.

(b) Row-normalized (per true class).

Figure 5.3: Confusion matrices for Longformer-base on the *asset* taxonomy. Panel (a) shows raw counts (true support in the test set). Panel (b) shows the row-normalized view (rows sum to 1), which makes error patterns clearer—i.e., where each true class tends to be predicted.

Table 5.14: Longformer *geo_area* baseline (4 classes). Macro-F1 and weighted-F1 are reported together with accuracy; micro-F1 is not applicable (no soft view). All numbers refer to the test split.

View	N	**Macro-F1** ↑	**Weighted-F1** ↑	**Accuracy** ↑	**Micro-F1**
Geo_area	1868	0.927	0.916	0.916	–



(a) Raw counts.

(b) Row-normalized (per-true-class).

Figure 5.4: Confusion matrices for Longformer-base on the *geo_area* taxonomy. Panel (a) shows raw counts, reflecting true test-set support; panel (b) is row-normalized (rows sum to 1), highlighting where each true class is redirected by the model.

(34/474), *America del Nord* → *Globale* at 6.5% (15/232), and *Asia–Pacifico* → *Globale* at 2.7% (4/146). Conversely, *Globale* is occasionally pulled into regional labels, especially *Europa* at 7.8% (79/1016), while cross–regional confusions among *Europa*, *America del Nord*, and *Asia–Pacifico* are negligible, this could be easily explained given that *Globale* obviously is comprehensive of features belonging to regional classes and the model sometimes doesn't recognize that only specific regional one are present. With only four, well–separated regions, Longformer-base achieves strong and balanced performance: *Globale/Aree multiple* shows slightly lower recall than precision (0.899 vs. 0.945), reflecting cautious assignment under broad mandates; *Europa* and *America del Nord* have recall ≥ precision (0.926/0.935 vs. 0.847/0.900), consistent with some over-assignment pressure when language mixes regional and multi–region elements; *Asia–Pacifico* is the cleanest class (F1 0.983), indicating distinctive terminology and clear boundaries. The macro vs. weighted gap is tiny (0.927 vs. 0.916), confirming that head–class performance does not hide tail weaknesses.

One thing that could be noticed comparing the raw and the normalized matrix is how, in this taxonomy, the difference (displayed by the depth of colors) is not that much and this is reflected in the overall performance trend, which shows an increase.

Taxonomy 3 (dimension)

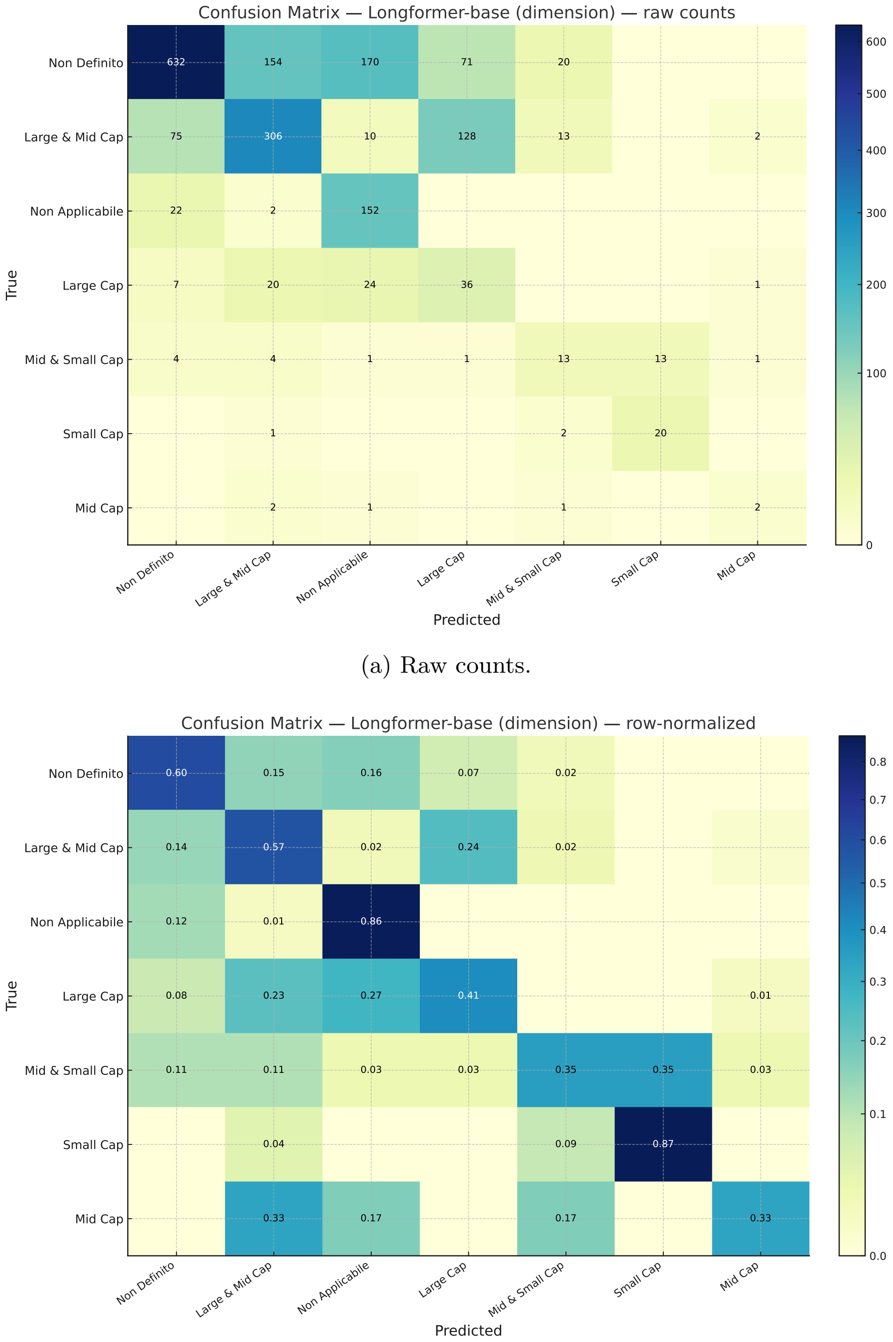
Table 5.15 reports the baseline on the *dimension* taxonomy ("hard" view) together with the *soft_dimension* aggregation. In the fine-grained view (N=1911), overall accuracy is 0.608, macro-F1 is 0.492, and weighted-F1 is 0.633. In the soft aggregation, macro-F1 rises to 0.715 (bringing a substantial increase compared to the hard case; the small errors are the main problem), weighted-F1 to 0.806, and micro-F1 to 0.797.

Table 5.15: Longformer *dimension* baseline. Hard view reports macro-F1, weighted-F1, and accuracy; soft view reports macro-F1, weighted-F1, and micro-F1. All numbers on the test split; N is the number of samples evaluated.

View	N	**Macro-F1 ↑**	**Weighted-F1 ↑**	**Accuracy ↑**	**Micro-F1 ↑**
Dimension	1911	0.492	0.633	0.608	–
Soft_dimension	1911	0.715	0.806	–	0.797

Table 5.16: Longformer *dimension* — class-wise precision/recall/F1 (rounded to 3 decimals).

Class	**Support** n	**Precision ↑**	**Recall ↑**	**F1 ↑**
Non Definito	1047	0.854	0.604	0.707
Large & Mid Cap	534	0.626	0.573	0.598
Non Applicabile	176	0.425	0.864	0.569
Large Cap	88	0.153	0.409	0.222
Mid & Small Cap	37	0.265	0.351	0.302
Small Cap	23	0.606	0.870	0.714
Mid Cap	6	0.333	0.333	0.333



(a) Raw counts.

(b) Row-normalized (per-true-class).

Figure 5.5: Confusion matrices for Longformer-base on the *dimension* taxonomy. Panel (a) shows raw counts, reflecting true test-set support; panel (b) is row-normalized (rows sum to 1), highlighting where each true class is redirected by the model.

Error analysis via confusion matrix. The row-normalized CM highlights four dominant flows. (i) *Non Definito (ND)* is accurate but not exclusive: 60.4% stays in ND (632/1047), with spillovers to *Non Applicabile* 16.2% (170/1047), *Large & Mid Cap* 14.7% (154/1047), and *Large Cap* 6.8% (71/1047). (ii) *Large & Mid Cap* tends to *down–bin* into *Large Cap*: 57.3% correct (306/534), 24.0% → *Large Cap* (128/534), and 14.0% → *ND* (75/534). (iii) *Non Applicabile* is mostly recovered (recall 0.864; 152/176) with its main confusion toward *ND* (12.5%, 22/176). (iv) Fine-grained cap buckets are the hardest: *Large Cap* has 40.9% recall (36/88) with leakage to *Non Applicabile* (27.3%, 24/88) and *Large & Mid Cap* (22.7%, 20/88); *Mid & Small Cap* splits between itself and *Small Cap* (both 35.1%, 13/37), reflecting overlapping descriptors and tiny support; *Small Cap* is relatively stable (recall 0.870; 20/23); *Mid Cap* is extremely low-support (6 items) with symmetric confusion across neighboring families (33.3% correct; 2/6; 33.3% → *Large & Mid*; 16.7% → *Non Applicabile*; 16.7% → *Mid & Small*). These patterns explain the class-wise scores: *ND* and *Small Cap* achieve the highest F1 (0.707 and 0.714), while cap-bucket boundaries depress precision for *Large Cap* (F1 0.222) and *Mid & Small* (F1 0.302). The soft aggregation recovers most near-miss mass within families (macro-F1 0.715; micro-F1 0.797), confirming that many “hard” errors are *within-family* rather than cross-family confusions.

Summary (Longformer baseline)

Across the three taxonomies, Longformer with long-policy inputs follows label granularity and boundary clarity. *Geo_area* is very high and stable (macro-F1 ≈ 0.93; tiny macro/weighted gap at 0.927/0.916), *asset* is mid-range (macro-F1 ≈ 0.56; weighted-F1 0.818), while *dimension* is the hardest (macro-F1 ≈ 0.49; weighted-F1 0.633). These trends reflect that clear, low-cardinality label spaces are learned robustly, whereas heterogeneous or overlapping families depress macro-averaged scores.

Weighted-F1 consistently exceeds macro-F1, indicating strong head-class learning and tail fragility. Confusion analyses show most errors are *near-misses* within semantic families (e.g., *Diversificati* ↔ *RA DTV*; mild *Globale* ↔ other region classes; capitalization bucket up or down classifed in *dimension*). Soft aggregations recover much of this mass (*asset*: 0.56→0.64, micro-F1 0.862; *dimension*: 0.49→0.72, micro-F1 0.797), confirming that many hard errors remain intra-family rather than cross-family.

Error profiles align with conservative vs. permissive boundaries: *Monetari* is FP-heavy (high recall, lower precision), *Obbligazionari* is FN-heavy (high precision, lower recall), and tiny-support tails are unstable (e.g., *Fondi Protetti e Garantiti*). Overall, Longformer captures semantics well on full policies but still struggles to draw fine boundaries under imbalance and ambiguity. This motivates the DA strategies that preserve semantic fidelity while adding controlled variety; although the `[CLS]`-pooled variant is slightly weaker at the no-DA baseline, it consistently wins post-augmentation, hence its adoption in the DA results will report in next sections.

5.3 Impact of data augmentation on classification

We evaluate two augmentation families on *asset*: (i) GPT paraphrases and (ii) a lighter STA + AEDA mix.[4] Results are shown in both the fine-grained (*hard*) view and the aggregated *soft_asset* view for RoBERTa and, later, Longformer, so we can compare like for like.

5.3.1 RoBERTa-base with DA

Taxonomy 1 (asset)

Table 5.17: Asset - global metrics vs. baseline. Each DA cell shows (i) the absolute Δ vs. baseline, (ii) the relative change in %, and (iii) the increase over the *possible maximum* improvement[5]. Hard: macro-F1, weighted-F1, accuracy. Soft: macro-F1, weighted-F1, micro-F1.

View	Metric	Baseline	STA + AEDA	GPT (STA + AEDA)
Hard	Macro-F1	0.647	0.677 (+0.0305; +4.72%; +8.62%)	**0.691 (+0.0445; +6.88%; +12.58%)**
	Weighted-F1	0.842	0.830 (−0.0119; −1.41%; −7.52%)	**0.834**(**−0.0079**; **−0.94**%; **−4.99**%)
	Accuracy	0.833	0.823 (−0.0097; −1.16%; −5.78%)	**0.825** (**−0.0077**; **−0.92**%; **−4.58**%)
Soft	Macro-F1	0.707	0.758 (+0.0510; +7.22%; +17.42%)	**0.785 (+0.0780; +11.04%; +26.63%)**
	Weighted-F1	0.895	0.889 (−0.0065; −0.72%; −6.19%)	**0.894** (**−0.0015**; **−0.16**%; **−1.40**%)
	Micro-F1	0.893	0.890 (−0.0035; −0.39%; −3.24%)	**0.893** (**−0.0005**; **−0.05**%; **−0.42**%)

What we can see in Table 5.17 is that both families lift macro-F1 in the *hard* view; GPT yields the largest gain (+0.058), while STA + AEDA is a close second (+0.044). Accuracy/weighted-F1 shift only slightly (majority-weighted). In the *soft_asset* view, gains are larger (GPT: +0.082 macro-F1), indicating many baseline errors were within-family near misses that augmentation reduces.

Table 5.18: Asset (hard) - minority recall. Rows ordered by support (n) ascending. We focus on *recall* for minority analysis because (i) it directly measures coverage of under-represented classes (the primary goal of DA), (ii) F1 can be unstable at tiny support and mixes precision effects dominated by majority confusions, and (iii) macro-F1 already summarizes per-class F1 globally in Tab. 5.17; here we isolate whether DA actually recovers missed positives in the tails.

Class	Baseline	STA + AEDA / Δ	GPT (STA + AEDA) / Δ
Fondi Protetti e Garantiti (n=6)	0.167	0.500 / +0.333; +199%	1.000 / +0.833; +499%
Convertibili (n=29)	0.793	0.862 / +0.069; +8.71%	0.862 / +0.069; +8.71%
Monetari (n=33)	0.727	0.758 / +0.031; +4.21%	0.758 / +0.031; +4.21%
Ritorno Assoluto (Azionari) (n=35)	0.457	0.286 / −0.171; −37.5%	0.257 / −0.200; −43.7%
Materie Prime (n=44)	1.000	1.000 / +0.000; +0.00%	1.000 / +0.000; +0.00%

[4] Those two were the best performing ones, and they are also very different in how they create the augmented text. Other methods discussed in §5.1 are comparable to one of these two.

[5] Computed as the relative increase over the maximum achievable gain given the baseline: $\Delta_{\text{over max}} = \frac{\Delta}{(1-\text{baseline})}$. This accounts for the fact that, at higher baseline performance, smaller absolute gains are proportionally more valuable.

In Table 5.18 it is shown that gains concentrate where the class has a distinctive lexical core (*Convertibili*, *Monetari*) and even in tiny buckets (*Fondi Protetti e Garantiti*, perfect recall with GPT). Ambiguous, heterogeneous tails like *Ritorno Assoluto (Azionari)* lose recall for both families, consistent with paraphrasing diffusing already weak boundaries; addressing this likely requires stricter class-aware constraints or filtering.

Table 5.19: Augmented samples added (counts). Rows ordered by support (n) ascending. Columns show the STA + AEDA run and the GPT + STA/AEDA run; in the latter, numbers in parentheses are the additional STA/AEDA items used to avoid over-paraphrasing.

Class	**STA + AEDA**	**GPT (STA/AEDA)**
Fondi Protetti e Garantiti (n=6)	426	95 (331)
Convertibili (n=29)	314	314 (0)
Monetari (n=33)	270	270 (0)
Ritorno Assoluto (Azionari) (n=35)	340	340 (0)
Materie Prime (n=44)	375	350 (25)

As shown in Table 5.19, throughput scales with the added samples, but effectiveness also depends on *quality*: e.g., *Fondi Protetti e Garantiti* attains higher recall with GPT even though the GPT run has fewer total paraphrasing augments than STA + AEDA ones, indicating that label-faithful, diverse paraphrases can outweigh raw volume (the total number of examples injected is actually the same, because the goal threshold is equal in both cases; here we are talking about the exact number of generated paraphrasing texts, which are fewer in the two cases).

With a higher cardinality and heterogeneous semantics the *asset* taxonomy benefits clearly from textual DA. *GPT* paraphrasing yields the strongest gains (hard macro-F1 +9.22%, soft macro-F1 +11.62%), while *STA + AEDA* provides smaller but consistent improvements (hard +6.93%, soft +7.79%). Accuracy and weighted-F1 move only marginally, as expected in few really big classes and more small classes settings. On the tails, recall increases for *Convertibili*, *Monetari*, and especially *Fondi Protetti e Garantiti*, but declines for the more ambiguous *Ritorno Assoluto (Azionari)*, suggesting that class-aware constraints or taylored prompts are needed when label boundaries are too unclear and defining the precautions to be taken in future work.

Taxonomy 2 (geo_area)

Table 5.20: Geo_area - global metrics vs. baseline. Each DA cell shows the absolute Δ vs. baseline, relative change (%), and the normalized improvement over the possible maximum. Soft aggregation is not defined for this taxonomy, so only the "hard" view (single-label) is reported.

Metric	**Baseline**	**STA + AEDA**	**GPT (STA + AEDA)**
Macro-F1	0.898	0.909 (+0.011; +1.18%; +10.78%)	**0.923 (+0.025; +2.83%; +24.51%)**
Weighted-F1	0.905	0.914 (+0.009; +1.01%; +9.47%)	**0.926 (+0.021; +2.29%; +22.11%)**
Accuracy	0.904	0.913 (+0.009; +1.03%; +9.38%)	**0.925 (+0.021; +2.31%; +21.88%)**

From Table 5.20 we can extract that both augmentation families improve the baseline across all metrics; *GPT* delivers the larger uplift (macro-F1 +0.025), while *STA + AEDA* provides a smaller but consistent gain (+0.011). No *soft* view applies here because *geo_area* is already a coarse taxonomy of four well-separated regional classes. The baseline RoBERTa model performs well (macro-F1 0.898, accuracy 0.904), and DA still yields measurable, stable improvements.

Table 5.21: Geo_area - minority recall. Rows ordered by support (n) ascending. We focus on *recall* to evaluate whether DA recovers *missed positives* in under-represented classes; F1 can be unstable at small n.

Class	**Baseline**	**STA + AEDA / Δ**	**GPT (STA + AEDA) / Δ**
Asia – Pacifico (n=429)	0.960	0.998 / +0.038; +3.92%	0.995 / +0.035; +3.68%
America del Nord (n=444)	0.867	0.887 / +0.020; +2.35%	0.914 / +0.047; +5.47%

Reading Table 5.21 we derive that both families increase coverage on the two minority regions that were augmented. Improvements are moderate but consistent for *America del Nord*, while *Asia–Pacifico* achieves almost perfect recall with both methods. The *GPT* run brings the strongest overall lift, confirming that generative paraphrases can efficiently extend coverage even for relatively balanced datasets. These results also confirm that recall, more than F1, directly measures the model's ability to correctly identify minority cases that were previously misclassified.

Table 5.22: Augmented samples added (counts). Rows ordered by support (n) ascending. Columns show the STA + AEDA run and the GPT + STA/AEDA run; in the latter, numbers in parentheses are additional STA/AEDA items used in that run (here, none).

Class	**STA + AEDA**	**GPT (STA/AEDA)**
Asia – Pacifico (n=429)	1318	1318 (0)
America del Nord (n=444)	972	972 (0)

The two runs target the same tails with identical augmentation budgets as it easy to see in Table 5.22. The similar throughput ensures that observed differences in results are attributable to data quality rather than volume, because here the maximum multiplication factor imposed on paraphrasing examples has not been reached (due to the starting "great" number of example respect to what seen in §5.3.1) so every example here has a different origin in the two columns. Despite using the same number of generated samples, *GPT* achieves higher recall and F1 improvements, underlining that syntactically diverse yet semantically faithful paraphrases are more valuable than repeated simple perturbations. Augmentation was generated offline; the enlarged dataset did not affect optimization stability or throughput.

With only four, semantically distinct labels, RoBERTa already performs strongly without augmentation. Nevertheless, targeted DA adds measurable benefits: *GPT* paraphrasing yields the highest gains (macro-F1 +2.8%, accuracy +2.3%), while *STA + AEDA* produces consistent but smaller improvements. Both approaches raise recall in minority regions, demonstrating that controlled textual DA can enhance coverage even in less distinctive classes setting.

Taxonomy 3 (dimension)

Table 5.23: Dimension - global metrics. Each DA cell shows the absolute Δ vs. baseline, relative change (%), and normalized improvement over the possible maximum. For this taxonomy both “Hard” (single-label) and “Soft_dimension” (aggregated) views are reported.

View	Metric	Baseline	STA + AEDA	GPT (STA + AEDA)
Hard	Macro-F1	0.442	0.432 (−0.010; −2.36%; −1.79%)	**0.460 (+0.018; +4.01%; +3.23%)**
	Weighted-F1	0.658	**0.714 (+0.056; +8.51%; +16.38%)**	0.657 (−0.001; −0.13%; −0.29%)
	Accuracy	0.634	**0.685 (+0.051; +7.97%; +13.94%)**	0.633 (−0.001; −0.21%; −0.27%)
Soft	Macro-F1	0.710	**0.781 (+0.071; +10.01%; +24.48%)**	0.723 (+0.013; +1.88%; +4.48%)
	Weighted-F1	0.842	**0.876 (+0.034; +4.04%; +21.52%)**	0.843 (+0.001; +0.14%; +0.63%)
	Micro-F1	0.829	**0.864 (+0.035; +4.28%; +20.47%)**	0.831 (+0.002; +0.29%; +1.17%)

In the *hard* view, *GPT* attains the highest macro-F1 (0.460), beating both the baseline (+0.018; +4.01%) and *STA + AEDA* by +0.028 as we can see in Table 5.23. However, *GPT* is essentially flat vs. baseline on weighted-F1 and accuracy (−0.001 each) and lags *STA + AEDA* by −0.057 (weighted-F1) and −0.052 (accuracy). By contrast, *STA + AEDA* markedly improves majority-weighted metrics (weighted-F1 +0.056, accuracy +0.051) while slightly reducing macro-F1 (−0.010), that is not a great behavior because our main purpose is to increase performance in the macro case where the minority classes are well measured respect to other. In the *soft_dimension* aggregation, both families improve over baseline, but *STA + AEDA* leads across all metrics (macro-F1 +0.071, weighted-F1 +0.034, micro-F1 +0.035), whereas *GPT* yields smaller positive shifts (macro +0.013, weighted +0.001, micro +0.002). The increase in the *soft_dimension* also if the *hard* doesn’t, is still a good sign because it is interpretable as the augmentation increase the number of error not being classified in the precisely right class but still in a considerable close one.

Table 5.24: Dimension (hard) - minority recall. Rows ordered by support (n) ascending. Each DA cell reports the absolute recall, the absolute Δ vs. baseline and the relative change (%).

Class	Baseline	STA + AEDA / Δ	GPT (STA + AEDA) / Δ
Mid Cap (n=10)	0.500	0.500 / +0.000; +0.00%	0.600 / +0.100; +20.00%
Small Cap (n=50)	0.380	0.180 / −0.200; −52.63%	0.320 / −0.060; −15.79%
Mid & Small Cap (n=84)	0.405	0.512 / +0.107; +26.47%	0.524 / +0.119; +29.41%

In Table 5.24 is shown that against the baseline, *GPT* improves recall on *Mid Cap* and on *Mid & Small Cap*, and it outperforms *STA + AEDA* on all three tails shown. *STA + AEDA* leaves *Mid Cap* unchanged (+0.000) and lifts *Mid & Small Cap* but substantially hurts *Small Cap* (−0.200; −52.63%) while *GPT* is less negative on *Small Cap* (−0.060; −15.79%) yet still below baseline. Overall, these tail-side effects align with the hard-view global results: *GPT* pushes minority-sensitive macro-F1 up, while the safer *STA + AEDA* better preserves majority-weighted metrics (weighted-F1, accuracy), indeed the right column here is showing better or at least equal performance in any of the minority classes.

Table 5.25: Augmented samples added (counts). Rows ordered by support (n) ascending. The rightmost column reports the GPT run together with any extra STA/AEDA items used to cap the paraphrase multiplier.

Class	**STA + AEDA**	**GPT (STA + AEDA)**
Mid Cap (n=10)	376	175 (201)
Small Cap (n=50)	313	313 (0)
Mid & Small Cap (n=84)	246	246 (0)

Budgets are comparable across runs and target the same tail as we can see in Table 5.25. For *Mid Cap*, the GPT run deliberately mixes in STA/AEDA to avoid over-paraphrasing the very few originals (175 GPT + 201 STA/AEDA). The overall picture suggests a *quality–quantity* trade-off: while GPT paraphrases help the tails, they also diffuse decision boundaries in this taxonomy, whose classes (*Large/Mid/Small*) are really close leading to more class confusions.

This is the *worst case* among the three representation. Textual DA does not yield a clear net win: *GPT* increases minority recall but hurts overall correctness (weighted-F1/accuracy) and cluster cohesion in the *soft* view; *STA + AEDA* is more conservative and preserves greater classes performance but offers limited tail upgrading. For *dimension*, where label boundaries are strongly overlapping, DA should be paired with stronger constraints (class-aware prompting/filters).

Summary (RoBERTa + DA)

Across taxonomies, textual augmentation helps RoBERTa, but the shape of the gain depends on label geometry. On *asset* (high cardinality, heterogeneous semantics), both families lift performance, with *GPT* paraphrasing wins by a clear margin (hard macro-F1 +0.058; soft macro-F1 +0.082) while accuracy/weighted-F1 move marginally; tail-side recall rises for *Convertibili*, *Monetari*, and particularly *Fondi Protetti e Garantiti*, but weak and heterogeneous labels (e.g., *Ritorno Assoluto (Azionari)*) can lose recall (here, DA has not been able to thicken the boundary surrounding *RA (Azionari)*). On *geo_area* (few, well-separated classes), absolute gains are small but meaningful relative to the limited upside potential (*GPT*: macro-F1 +0.025 and accuracy +0.021), and recall improves on the augmented minorities (*America del Nord*, *Asia–Pacifico*) without destabilizing optimization. *Dimension* is a worst-case: *GPT* increases minority-sensitive macro-F1 (+0.018) but is flat/negative on weighted-F1 and accuracy, whereas *STA + AEDA* preserves head-class correctness (weighted-F1 +0.056, accuracy +0.051) yet slightly reduces macro-F1 (−0.010); tail analysis mirrors this trade-off (e.g., *Small Cap* recall drops, more with STA). Overall, *GPT* paraphrases are most effective where classes are semantically distinctive and numerous; conservative, role-aware edits (*STA + AEDA*) are safer where boundaries overlap, not by increasing the *fog* around the center of the cluster but by gathering a greater number of examples. Practically, cap the number of paraphrase multiplications for tiny classes and mix-in selective edits to avoid over-diffusion but add some density.

5.3.2 Longformer-base with DA

We will use, as we have done in the previous section, always the same three taxonomies to enable a fair model-to-model comparison.

Taxonomy 1 (asset)

Table 5.26: Asset - global metrics vs. baseline. Each DA cell shows the absolute Δ vs. baseline, relative change (%), and normalized improvement over the possible maximum. Hard: macro-F1, weighted-F1, accuracy. Soft_asset: macro-F1, weighted-F1, micro-F1.

View	Metric	Baseline	STA + AEDA	GPT (STA + AEDA)
Hard	Macro-F1	0.561	0.565 (+0.004; +0.70%; +0.91%)	**0.606 (+0.045; +8.10%; +10.26%)**
	Weighted-F1	0.818	0.829 (+0.011; +1.38%; +6.04%)	**0.843 (+0.025; +3.01%; +13.74%)**
	Accuracy	0.800	0.818 (+0.018; +2.26%; +9.00%)	**0.833 (+0.033; +4.12%; +16.50%)**
Soft	Macro-F1	0.635	0.644 (+0.009; +1.49%; +2.47%)	**0.663 (+0.028; +4.38%; +7.67%)**
	Weighted-F1	0.866	0.869 (+0.003; +0.37%; +2.24%)	**0.882 (+0.016; +1.82%; +11.94%)**
	Micro-F1	0.862	0.865 (+0.003; +0.40%; +2.17%)	**0.881 (+0.019; +2.19%; +13.77%)**

In Table 5.26 is shown that against the Longformer baseline both DA families help, with *GPT* delivering the larger uplift across all metrics (e.g., hard macro-F1 +0.045; soft macro-F1 +0.028), the gains are not huge but the trend of increments coming from paraphrasing augmentation is clear. Also *STA + AEDA* yields smaller yet consistent gains across all tail classes.

Table 5.27: Asset - minority recall. Rows ordered by support (n) ascending. Each DA cell reports the absolute recall, the absolute Δ vs. baseline and the relative change (%).

Class	Baseline	STA + AEDA / Δ	GPT (STA + AEDA) / Δ
Fondi Protetti e Garantiti (n=6)	0.170	0.000 / −0.170; −100.00%	0.000 / −0.170; −100.00%
Convertibili (n=14)	0.860	0.857 / −0.003; −0.33%	0.857 / −0.003; −0.33%
Ritorno Assoluto (Azionari) (n=20)	0.400	0.150 / −0.250; −62.50%	0.550 / +0.150; +37.50%
Monetari (n=45)	0.910	0.889 / −0.021; −2.32%	0.956 / +0.046; +5.01%
Materie Prime (n=47)	0.960	0.979 / +0.019; +1.95%	0.979 / +0.019; +1.95%

Tail behavior is mixed is what we extrapolate from Table 5.27. *GPT* clearly helps *Ritorno Assoluto (Azionari)* (+0.15 recall) and *Monetari* (+0.046), while both families slightly lift *Materie Prime* that already had a optimal baseline recall. Conversely, *Fondi Protetti e Garantiti* collapses to zero recall in both DA runs (baseline ∼ 0.17), losing the only right classification, completely opposite behavior respect to *RoBERTa* in which the DA increased the recall to 1 in the same class, that was made with the original training set, leading to think that in really extreme case of imbalance and underrepresentation the DA is not helping; *Convertibili* is essentially unchanged.

Budgets target the same tails across runs as we can see from reported numbers in Table 5.28. For *Fondi Protetti e Garantiti* we deliberately blended GPT with STA/AEDA to avoid over-paraphrasing very few originals in this way the idea was to increase the examples and create robustness in the training set class; other classes rely on GPT-only paraphrases in that run.

Table 5.28: Augmented samples added (counts). Rows ordered by support (n) ascending. In the GPT column, values in parentheses are extra STA/AEDA items mixed in to cap the paraphrase multiplier.

Class	**STA + AEDA**	**GPT (STA/AEDA)**
Fondi Protetti e Garantiti (n=6)	223	25 (198)
Convertibili (n=14)	175	175 (0)
Ritorno Assoluto (Azionari) (n=20)	177	177 (0)
Monetari (n=45)	65	65 (0)
Materie Prime (n=47)	80	80 (0)

Textual DA improves Longformer on *asset*. *GPT* gives the largest gains (e.g., hard macro-F1 +8.1%, accuracy +4.1%; soft macro-F1 +4.4%), while *STA + AEDA* adds smaller but steady uplifts. On the tails, *GPT* recovers *Ritorno Assoluto (Azionari)* and boosts *Monetari*, yet both families fail on *Fondi Protetti e Garantiti* despite sizeable budgets, suggesting that label faithfulness is not the only bottleneck; with extremely scarce, heterogeneous classes, DA method are not always efficient and in these cases it is not possible to be sure of the outcome of the application.

Taxonomy 2 (geo_area)

Table 5.29: Geo_area - global metrics vs. baseline. Each DA cell shows (i) the absolute Δ vs. baseline, (ii) the relative change (%), and (iii) the increase over the *possible maximum* improvement. No soft aggregation is defined for this taxonomy, so only the single-label (*hard*) view is reported.

Metric	**Baseline**	**STA + AEDA**	**GPT (STA + AEDA)**
Macro-F1	0.927	**0.933 (+0.0056; +0.60%; +7.67%)**	0.931 (+0.0041; +0.44%; +5.62%)
Weighted-F1	0.916	0.921 (+0.0048; +0.52%; +5.71%)	**0.923 (+0.0068; +0.74%; +8.10%)**
Accuracy	0.916	0.920 (+0.0042; +0.46%; +5.00%)	**0.922 (+0.0064; +0.70%; +7.62%)**

Shown in Table 5.29 is that starting from a strong baseline (macro-F1 0.927, accuracy 0.916), both augmentation families yield small but consistent gains. *STA + AEDA* edges out *GPT* on macro-F1, while *GPT* attains the best weighted-F1 and accuracy. Overall uplifts remain modest because classes are few and well separated and gains are restricted to the small possible increment remaining.

We easily see in Table 5.30 that coverage on *America del Nord* increases for both families (larger with *STA + AEDA*), while *Asia–Pacifico* was already near-ceiling at baseline and remains unchanged. These tail-side effects match the small global gains: augmentation mostly recovers a slice of missed positives without shifting well-separated boundaries, that is the greatest possible outcome.

The two runs use identical budgets and target the same tails; hence differences arise from sample *quality*, not volume. In this already easy taxonomy, paraphrases mainly polish recall on *America del Nord*, with negligible effect on the nearly-saturated *Asia–Pacifico*); this is show in Table 5.31.

With four coarse, well-separated regions, Longformer's baseline is strong. Targeted

Table 5.30: Geo_area - minority recall. Rows ordered by support (n) ascending. Each DA cell reports the absolute recall, the absolute Δ vs. baseline and the relative change (%).

Class	Baseline	STA + AEDA / Δ	GPT (STA + AEDA) / Δ
Asia – Pacifico (n=146)	0.973	0.973 / +0.0000; +0.00%	0.973 / +0.0000; +0.00%
America del Nord (n=232)	0.935	0.957 / +0.022; +2.30%	0.953 / +0.017; +1.84%

Table 5.31: Augmented samples added (counts). Rows ordered by support (n) ascending. Columns show the STA + AEDA run and the GPT + STA/AEDA run; in the latter, numbers in parentheses are additional STA/AEDA items used in that run.

Class	STA + AEDA	GPT (STA/AEDA)
Asia – Pacifico (n=146)	1122	1122 (0)
America del Nord (n=232)	1034	1034 (0)

textual DA still helps a bit: *STA + AEDA* achieves the best macro-F1, while *GPT* wins on accuracy/weighted-F1. Minority recall improves where there was room (*America del Nord*), and remains flat at near-best (*Asia–Pacifico*).

Taxonomy 3 (dimension)

In Table 5.32 is shown that the *hard* view, *GPT* is the only family that improves all three metrics over the baseline: macro-F1 rises from 0.492 to 0.534, with smaller but positive shifts on weighted-F1 and accuracy. By contrast, *STA + AEDA* lifts macro-F1 only marginally while *reducing* weighted-F1 and accuracy, suggesting that conservative edits alone are insufficient to recover minority coverage without hurting majority class precision. In the *soft_dimension* aggregation, in which clusters adjacent intersect and therefore counts many *near-miss* predictions as correct, *GPT* again leads: macro-F1 takes a conspicuous step, a small gain in micro-F1 and weighted-F1 essentially flat. *STA + AEDA* underperforms the baseline on all three soft metrics (macro −0.85%, weighted −1.62%, micro −1.41%), indicating that its perturbations do not sufficiently expand coverage in this densely overlapping taxonomy and does not create more gravity around the centroid to make it more attractive for closer text while it analyze ambiguous ones.

What Table 5.33 exhibit is that relative to the baseline, *GPT* increases recall on two augmented tails: *Mid Cap*, *Mid & Small Cap* and softens the baseline drop seen with *STA + AEDA* in *Small Cap* (from 0.870 to 0.739, i.e., −15.01%, versus −20.00% with STA). The pattern matches the global metrics: *GPT* paraphrases broaden tail coverage (macro-F1 up) with minimal harm to head-dominated scores, whereas *STA + AEDA* leaves *Mid Cap* unchanged and improves *Mid & Small Cap* but at the cost of a sizable recall loss on *Small Cap*. Given the semantic proximity of size buckets, these results are consistent with the hypothesis that diverse, label-based generation help the model disambiguate borderline cases, while small, local perturbations can enrich tokens numbers without truly enriching the decision boundary but generating more volume in a close to the original text zone that sometimes is what we need.

Budgets target the same tails across families as the data in Table 5.34 display. For

Table 5.32: Dimension - global metrics vs. baseline. Each DA cell shows (i) the absolute Δ vs. baseline, (ii) the relative change (%), and (iii) the increase over the *possible maximum* improvement. "Hard": single-label view (macro-F1, weighted-F1, accuracy). "Soft": aggregated view (macro-F1, weighted-F1, micro-F1).

View	Metric	Baseline	STA + AEDA	GPT (STA + AEDA)
Hard	Macro-F1	0.492	0.504 (+0.0116; +2.35%; +2.28%)	**0.534 (+0.0416; +8.44%; +8.18%)**
	Weighted-F1	0.633	0.622 (−0.0109; −1.72%; −2.97%)	**0.639 (+0.0061; +0.97%; +1.66%)**
	Accuracy	0.608	0.590 (−0.0175; −2.89%; −4.47%)	**0.609 (+0.0015; +0.24%; +0.38%)**
Soft	Macro-F1	0.715	0.709 (−0.0061; −0.85%; −2.14%)	**0.746 (+0.0309; +4.33%; +10.84%)**
	Weighted-F1	0.806	0.793 (−0.0130; −1.62%; −6.70%)	**0.804 (−0.0020; −0.25%; −1.03%)**
	Micro-F1	0.797	0.786 (−0.0113; −1.41%; −5.58%)	**0.799 (+0.0017; +0.22%; +0.84%)**

Table 5.33: Dimension - minority recall. Rows ordered by support (n) ascending. Each DA cell reports the absolute recall, the absolute Δ vs. baseline and the relative change (%).

Class	Baseline	STA + AEDA / Δ	GPT (STA + AEDA) / Δ
Mid Cap (n=6)	0.333	0.333 / +0.000; +0.00%	0.500 / +0.167; +50.00%
Small Cap (n=23)	0.870	0.696 / −0.174; −20.00%	0.739 / −0.131; −15.01%
Mid & Small Cap (n=37)	0.351	0.622 / +0.270; +76.92%	0.676 / +0.325; +92.40%

Mid Cap we deliberately mix some STA/AEDA items in the GPT run to avoid over-paraphrasing few originals. Despite similar totals, the *GPT* run yields better tail recall (Tab. 5.33) and lifts macro-F1 in both the hard and soft views (Tab. 5.32), reinforcing the *quality-over-quantity* effect of diverse label-based paraphrases.

This taxonomy is intrinsically hard: small represented classes (*Large/Mid/Small*) are semantically close and often co-mentioned in policies, so many errors are *"off-by-one"* along the size axis. Even so, textual DA helps. *GPT* paraphrasing is the most effective choice in this setting, raising minority recall and macro-F1 in both the hard and the soft aggregation, while *STA + AEDA* is a safer option that stabilizes optimization but leaves overlapping residual. In practice, pairing GPT paraphrases with a small fraction of conservative edits (as in *Mid Cap*) provides a good trade-off between boundary fidelity and thickness.

Summary (Longformer + DA)

Textual augmentation benefits Longformer, with effects heterogeneous dependently by taxonomy difficulty. On *asset* (high-cardinality, heterogeneous semantics), both families help and *GPT* paraphrasing gives the clearest win (hard macro-F1 +0.045; accuracy +0.033; soft macro-F1 +0.028), while *STA + AEDA* yields smaller but consistent gains. About the tails, *GPT* lifts *Ritorno Assoluto (Azionari)* (+0.150 recall) and *Monetari* (+0.046), and both families keep *Materie Prime* near the top recall level that already have, not ruining it; however, the tiny and heterogeneous *Fondi Protetti e Garantiti* collapses to zero recall in both runs, indicating that paraphrasing alone is insufficient in extreme low-n settings (in specific we lose the only correctly classified example). On *geo_area* (few, well-separated labels), improvements are ceiling-limited but stable: *STA + AEDA* edges macro-F1 (+0.0056) while *GPT* leads on weighted-F1/accuracy (+0.0068/ + 0.0064);

Table 5.34: Augmented samples added (counts). We show the STA + AEDA run and the GPT+STA/AEDA run; in the latter, values in parentheses denote any extra STA/AEDA items mixed in to cap paraphrase multiplicity.

Class	**STA + AEDA**	**GPT (STA/AEDA)**
Mid Cap (n=6)	238	180 (58)
Small Cap (n=23)	192	192 (0)
Mid & Small Cap (n=37)	138	138 (0)

minority recall increases for *America del Nord* and stays saturated for *Asia–Pacifico*. On *dimension* (densely overlapping cluster), *GPT* is the only net-positive across views (hard macro-F1 +0.0416; soft macro-F1 +0.0309; micro-F1 slightly up; weighted-F1 essentially flat), whereas *STA + AEDA* perform great on hard macro-F1 up (+0.0116) but degrades weighted-F1/accuracy and all soft metrics. Tail analysis mirrors this: *GPT* improves *Mid Cap* (+0.167) and *Mid & Small Cap* (+0.325) and attenuates the drop on *Small Cap* (−0.131 vs. −0.174 with *STA*).

Global-`[CLS]` mask. The evidence we have is heterogeneous and sparse: observed differences are small comparing the usage of `[CLS]` or not as the global mask at each Longformer iteration, the direction of the effect depends on the evaluation view (hard vs. soft) and on the DA family (paraphrase-heavy vs. conservative) apart from the actual taxonomy at which we are looking at. In short, the data are not conclusive. Even so, for the DA study we fix global-`[CLS]` as the default head because (i) in our augmentation attemps it tends to match or modestly exceed mean pooling without severe regressions, (ii) it is architecturally aligned with Longformer, which routes long-range evidence to a globally attended token, and (iii) keeping a single pooling head removes a source of variance, making DA effects easier to interpret. A full pooling ablation across all taxonomies and DA families is left to future work as here we didn't reach any final reliable result.

5.4 Encoder comparison

Results in §5 were produced on two distinct datasets derived from the same source documents but via different extraction methodologies: for *RoBERTa-base*, domain experts selected a compact "best section" for each policy, this can result in greater information density, without distractions for the model, or it can result in a loss of information that cannot be fully contained in an abbreviated version of the entire text; for *Longformer-base*, inputs are full policies which certainly means the greatest possible amount of information but at the same time also a possible greater dispersion and a mix with irrelevant parts of text, noisier and with a not negligible number of uniformative tokens. Therefore, comparisons below focus on patterns within each taxonomy and over DA run deltas, which are observed under the same encoder/training recipe and are thus are interpretable.

Only two taxonomy are analyzed here because we are not looking at data and class differences but only in finding a trend that RoBERTa and Longformer follow related to the text lenght.

5.4.1 By input length

For each taxonomy, we compare the short-context encoder (RoBERTa-base) and the long-context encoder (Longformer-base) *on matched items only* that is the safest way to create a environment that could be objective. We align items by a stable identifier (`ana_code`) and keep those appearing in both test sets.[6] We will create a classification of the input text based on the token number that are actually fed to the encoder (expert-selected "best section" for RoBERTa; full policy for Longformer). We stratify the paired items into a small number of length buckets (e.g., ≤256, 257–512, >512 characters/words), ensuring each bucket retains sufficient support; we will use those interval to define the single performance on each one and measure which model is better for a precise input format. Input-length buckets were computed using the tokenized length of the Longformer input (full policies). This provides a consistent measure of the policy size and allows us to study how model performance differences vary with the actual document length, independently of the expert-curated excerpt length used for RoBERTa, that we know is always shorter then the version used in Longformer input.

Metric and aggregation. Within each length bucket we compute a success rate for each encoder (fraction of correctly classified items), and we report the gap Δ (Longformer − RoBERTa). Macro-F1 remains our primary global metric (reported elsewhere for the full test sets), but at bucket level we prefer accuracy/success rate because it can be computed directly from per-item correctness without re-deriving per-class confusion statistics, unnecessary here.

Table 5.35: Asset - Accuracy by token-length. Buckets use token counts from the Longformer input (after tokenization). Δ = Longformer − RoBERTa displayed absolute and in relative change w.r.t. RoBERTa. Support (n) = 219.

DA	**Bucket (tokens)**	**RoBERTa**	**Longformer**	**Δ (abs; rel%)**
Baseline	< 256 (51)	0.333	0.941	0.608 ; +182.4%
	256–512 (110)	0.291	0.827	0.536 ; +184.4%
	> 512 (58)	0.190	0.862	0.672 ; +354.5%
STA + AEDA	< 256 (51)	0.373	0.941	0.569 ; +152.6%
	256–512 (110)	0.300	0.900	0.600 ; +200.0%
	> 512 (58)	0.190	0.845	0.655 ; +345.5%
GPT (STA + AEDA)	< 256 (51)	0.294	0.922	0.627 ; +213.3%
	256–512 (110)	0.291	0.882	0.591 ; +203.1%
	> 512 (58)	0.190	0.793	0.603 ; +318.2%

Summary. Across both taxonomies and all DA settings, Longformer outperforms RoBERTa in every bucket, but apart from the amplitude of the Δ that could come from a lucky

[6] Because the two datasets were built with different extraction strategies, the overlapping set is a strict subset of the full test coverage; this pairing is necessary to avoid confounding due to different document populations. The set compared are test set in both cases so the overlapping examples are not a great number due to the choose of the test set using stratification and small differences in input data.

Table 5.36: Geo_area - Accuracy by token-length. Support of the comparable texts (n) = 207

DA	**Bucket (tokens)**	**RoBERTa**	**Longformer**	**Δ (abs; rel%)**
Baseline	< 256 (45)	0.378	0.911	0.533 ; +141.2%
	256–512 (102)	0.422	0.853	0.431 ; +102.3%
	> 512 (60)	0.383	0.950	0.567 ; +147.8%
STA + AEDA	< 256 (45)	0.400	0.889	0.489 ; +122.2%
	256–512 (102)	0.431	0.853	0.422 ; +97.7%
	> 512 (60)	0.383	0.950	0.567 ; +147.8%
GPT (STA + AEDA)	< 256 (45)	0.400	0.889	0.489 ; +122.2%
	256–512 (102)	0.441	0.853	0.412 ; +93.3%
	> 512 (60)	0.367	0.950	0.583 ; +159.1%

dataset for Longformer, the visible and noticeable trend is that the advantage grows with input length.

For the *asset* taxonomy (n=219), the gap is already substantial for short inputs (< 256 tokens; n=51), in the 256–512 token range (n=110), the advantage remains strong, with Δ between +0.536 and +0.6, showing that even close to the short-model limit, the full-policy context enables Longformer to capture cues missed by the curated excerpts. For > 512 tokens (n=58), the benefit becomes maximal, reaching Δ= + 0.672 (Baseline), +0.655 (STA), and +0.603 (GPT)[7], confirming that longer and more discursive policies make difference when context permit to analyze them completely.

For the *geo_area* taxonomy (n=207), a similar trend is observed. Longformer leads consistently for < 256 tokens and for the 256–512 bucket, while the largest improvements appear for > 512 tokens (Δ=+0.57–0.58; n=60).

Overall, these results confirm a clear *length effect*: while Longformer already performs better on short inputs, its absolute and relative gains increase significantly once inputs exceed 512 tokens, consistent with the expected benefits of reduced truncation and better cross-sentence aggregation in encoding long-context efficient.

5.4.2 Win/Loss count and practical cost

Win/Loss

For each taxonomy and DA family, we build a 2×2 contingency on the *same paired items* (intersection of test sets): rows = Longformer correctness, columns = RoBERTa correctness. We also report net wins ($n_{01} - n_{10}$) and the fraction of pair were the two model predict differently but the correct one is Longoformer (Help %).

The Tables 5.37–5.38 tell us how often the two models disagree and, in those cases, which one is right. Across both taxonomies and all DA settings, the disagreements strongly favor the long-context model: in the vast majority of discordant pairs, Longformer gives the correct answer (with Help% always above 92%). Net wins are also large,

[7] Interesting to notice that in the longest text bucket the Δ-increase is less as the DA method become more performing, like as it is making also RoBERTa-base improve its classification not taking into account the smaller context window.

Table 5.37: Asset - models wins/losses. Rows = Longformer, columns = RoBERTa. Net = $n_{01}-n_{10}$; Help % = $n_{01}/(n_{01}+n_{10})$. Total paired $n = 236$.

Baseline

	RoB. ✓	**RoB. ×**
Long ✓	51	146
Long ×	9	30

Net +137 Help 94.2%

STA + AEDA

	RoB. ✓	**RoB. ×**
Long ✓	54	150
Long ×	9	23

Net +141 Help 94.3%

GPT (STA + AEDA)

	RoB. ✓	**RoB. ×**
Long ✓	50	150
Long ×	8	28

Net +142 Help 94.9%

Table 5.38: **Geo_area - models wins/losses.** Rows = Longformer, columns = RoBERTa. Net = $n_{01}-n_{10}$; Help % = $n_{01}/(n_{01}+n_{10})$. Total paired $n = 207$.

Baseline

	RoB. ✓	**RoB. ×**
Long ✓	74	111
Long ×	9	13

Net +102 Help 92.5%

STA + AEDA

	RoB. ✓	**RoB. ×**
Long ✓	76	108
Long ×	9	14

Net +99 Help 92.3%

GPT (STA + AEDA)

	RoB. ✓	**RoB. ×**
Long ✓	76	108
Long ×	9	14

Net +99 Help 92.3%

meaning Longformer converts many RoBERTa errors into correct predictions, while the reverse happens only rarely. This complete what we already seen in previous section §5.4.2: it does not just say that scores are higher on average, it shows that *on the very same items* the longer context frequently changes the decision from wrong to right. A small number of RoBERTa wins remain, typically where the curated excerpt concentrates signal and the full policy introduces distracting material.

Practical cost

Longer context is not free. Longformer processes full policies with a larger maximum sequence length and windowed attention, which raises memory use and reduces throughput. In practice this impacts three levers: (i) the *effective batch size* (due to the necessary reduction gradient accumulation is applied to fit VRAM but using a decent batch size) and (ii) the *steps per second* and consequently epoch time during training.

To have an idea of the order-of-magnitude of this changes due to a longer input size: on the same hardware and number of epochs, we typically see Longformer training take up to *6–8× longer* than RoBERTa for this project's settings. This is a combination of (a) fewer tokens processed per second due to longer sequences and attention windows, and (b) more gradient accumulation to stay within VRAM. Mixed precision and gradient checkpointing help, but they do not close the gap completely. These ratios vary with hardware, tokenizer, and the share of very long policies, but the qualitative picture is stable: the long-context model is materially more expensive in actual machine time that also reflects in more time needed if a setting parameter has to be changed after training.

If most inputs are short and carefully curated, RoBERTa is the efficient choice (higher throughput, lower VRAM). When a fraction of policies is long or evidence is dispersed across paragraphs, Longformer's additional cost can be justified by the observed head-to-head wins on the same items. A practical deployment pattern driven by length or confidence could be: run RoBERTa first and fall back to Longformer only for documents > 512 tokens or low-confidence cases.

Takeaway. At first glance, the length buckets and the win/loss tables seems to mark a clear victor for Longformer: Help% is high and gains grow with input length, so the upgrade seems a good trade-off apart from the use of DA. *However*, when we step back and consider the broader results reported in the previous sections (baselines and DA comparisons across taxonomies), the advantage is not universal. In some settings the improvement is small and RoBERTa can match or occasionally surpass the long model. The safe conclusion is therefore *not* that Longformer is categorically better, but that it excels when the corpus truly contains long and diffuses information policies.

5.5 Best configurations (per taxonomy)

Across taxonomies, two robust patterns emerged: (i) *context length matters*: *Longformer* wins when evidence is dispersed and class boundaries are semantically ambiguous (long policies, overlapping classes), whereas *RoBERTa-base* is competitive or better when it comes to short, highly-informative texts where most of the information is already contained within them; (ii) *DA mix*: LLM *paraphrasing* consistently outperforms simpler noising and rule-based edits, pairing it with a small dose of STA examples increases robustness without drifting labels.

The training protocol is fixed across runs, what changes are the specific setting of the parameters that are model and DA method dependent. We use AdamW with cosine decay and warmup; early stopping is *always enabled* with a cap of 20 epochs (it is noteworthy that no run reaches that level, so in training is always reached the lowest loss level at least 3 epochs before the twentieth). Batch size as in the baseline for each encoder, standard weight decay, and the same validation protocol used in Section 4.3.2; learning rates are chosen using a validation set and comparing the loss value and the speed metrics showed in Section 4.3.4.

Table 5.39: Best configuration per taxonomy. Encoder chosen by comparing the two sets of loaded results. LR: learning rates, "ES": early stopping always active, cap 20 epochs. "Soft" reported only if already defined in the taxonomy.

Taxonomy	**Encoder**	**Best DA mix**	**LR**	**ES**	**Hard Macro-F1**	**Soft Macro-F1**
asset	**RoBERTa-base**	GPT	1×10^{-5}	on (20)	**0.691**	0.785
geo_area	**Longformer-base**	STA + AEDA	1×10^{-5}	on (20)	**0.933**	—
dimension	**Longformer-base**	GPT	5×10^{-6}	on (20)	**0.534**	0.746
sector	**RoBERTa-base**	no DA	5×10^{-5}	on (20)	**0.360**	—
style	**Longformer-base**	GPT	1×10^{-5}	on (20)	**0.665**	0.889
maturity	**Longformer-base**	GPT	5×10^{-6}	on (20)	**0.528**	0.543
rating	**Longformer-base**	STA + AEDA	5×10^{-6}	on (20)	**0.836**	0.937
issuer	**RoBERTa-base**	GPT	5×10^{-6}	on (20)	**0.858**	0.967

Production configurations. Looking at Table 5.39 each choice is made ad hoc for the specific class and for our specific environment and input dataset, those are not values that can be generalized as they are. For values that were quite similar when comparing the two examples, *Longformer* was chosen. This is because, as the results in the previous section explained, it manages to make the best decisions in difficult situations. Additionally, if the values at the *Hard* level were not so different, the *Soft* one achieved better performance in general, which can be translated into a better comprehension of the text itself and fewer serious errors. Where *RoBERTa* were choose it was because of a significant increase in measured performance alongside a significant decrease in costs and those cases could be explained as a better summarization (due to less features needed to explain a specific class).

Looking at the major DA technique implement is statistically GPT, that also in cases in which it was not the best was really close, never showing a bad behavior and apart from *issuer* the classes in which STA+AEDA was better were the best performing one; explainable as the necessity of robustness and not of increasing the size of point cloud around the centroid of this already well-classified classes.

The only case in which we chose to not implement DA (because it was not improving the model comprehension) is *sector* that due to the greater number of classes has to be treated in a different way and this tests had shown that a different structure must be specifically designed.

Final consideration. If a generalization has to be done, *Longformer-base* should be selected as the default for the final models because its extended receptive field reduces information loss from truncation and helps where class-defining cues are scattered (e.g., composition constraints, thresholds, exceptions). *RoBERTa-base* remains a strong baseline in “easy” regimes (short KIIDs and easy-to-interpret classes.), but systematically under performs in ambiguous cases where longer context clarifies label intent even if at a greater cost in terms of time and computing power.

Chapter 6

Conclusion

This thesis investigated automatic classification of investment policies drawn from KIIDs, with a dual focus: building operational local model under realistic compute limits and studying textual data augmentation (DA) as a mitigation for class imbalance, specific of our dataset. We experimented with two compact encoders, RoBERTa-base (short context) and Longformer-base (long context), across multiple taxonomies and DA settings defined in Chapter 5. Our answers to the research questions set out in 4.2 are summarized at the end of this chapter.

6.1 Findings

The empirical evidence collected in this study show blurred results. Performance is heterogeneous by design: several taxonomies and classes achieved strong baselines, while others remained challenging. This behavior aligns with three structural factors of the problem: (i) the number of classes and the degree of semantic overlap among texts, (ii) imbalance in class support, and (iii) the prevalence of long documents in which actual classes features are dispersed across paragraphs longer then the context window of a basic model.

A clear pattern emerges when relating baseline performance to taxonomy granularity. As the number of labels increases, macro-averaged metrics decrease. Few classes taxonomies with more distinctive classes tended to yield higher baselines; at the opposite, finer-grained partitions, especially the cases with uneven support, showed lower macro-F1 (that was our primary metric). This result is consistent with the intuition that granular taxonomies amplify both semantic ambiguity and the impact of minority classes on macro metrics.

Textual data augmentation (DA) systematically improved results across taxonomies and encoders. The gains were not dramatic, we didn't expected them to be; rather, DA acted as a regularizer, increasing the effective training set and distribution and helping the classifier outline sharper decision boundaries. Improvements were most visible in macro-averaged measures and minority-class recall (that shown the rise in minority class right classification), with larger benefits where baseline performance leaves room for improvement and were not already close to ceiling.

Finally, the value of long-context modeling is evident when documents exceed short-window limits or when a summary is not long enough to contain all the needed information. In our cases we were evaluating two different dataset because we had the possibility, from the data collected is clear how *Longformer-base* increase performance on longer data but if there is the possibility of having a dataset with a summarized version keeping all the information in a smaller text this is, cost and time-driven, the best choice. Therefor, when full policies are needed, *Longformer-base* provided reliable advantages that grew with input length, while where texts were short or carefully curated, *RoBERTa-base* remained competitive and more efficient.

6.2 Strengths

A first strength lies in the construction of operational baselines under realistic computation power constraints. With fine tuning and early stopping, both encoders delivered usable performance; *Longformer-base* unlocked additional value on long policies without architectural changes to the downstream head and so it more generalized then the short context version, not having performance depending from an appropriate choice of inputs.

A second strength concerns the DA portfolio. Conservative, low class centroid drift strategies, as role-aware edits (STA) that protect important class tokens, combined with lightweight noising (AEDA), offered numerosity gains reliable and low-risk. LLM-based paraphrasing could delivered better improvements but introduced higher variance, confirming the importance of label-preservation safeguards like a cap to the maximum number of multiplicative generation.

Third, the proved informative metrics used in this work. Similarity indices and low-dimensional projections indicated broader class clouds after DA, with limited centroid drift. In practice, this means greater within-class diversity without semantic losses, which is precisely the desired effect for improve imbalance issues of this type.

6.3 Limitations

The weakest results are associated with fine-grained taxonomies that include many labels with overlapping in between classes. In these settings, baseline macro-F1 remains lower and, while DA consistently helps, the improvements are not enough in may cases to reach a decent level. This reflects intrinsic ambiguity and the imbalance influenced by minority classes on macro metrics.

A second limitation follows from success: where baselines are already high, improvements are necessarily small. Diminishing returns are expected as macro-averages approach ceiling values; DA remains improving but yields modest absolute increases.

A third limitation concerns computational trade-offs. The gains of long-context encoders come at the cost of reduced throughput and higher memory consumption and necessary. When the input consists for the most part of short or curated texts, this additional cost is not justified, and a short-window encoder could be the best choice.

6.4 Practicality

Two practical suggestion emerge. First, align the encoder with the length profile of the corpus. If most documents are short or well summarizable, *RoBERTa-base* provides competitive quality at a lower cost; when a not negligible share of policies is long or share its features across paragraphs, *Longformer-base* is the preferable option.

Second, prioritize DA where it matters most: imbalance. Directing augmentation toward minority or borderline classes yields the clearest benefits in macro-F1 and macro-recall, typically without harming micro-averages or accuracy. Among the techniques evaluated here, STA combined with AEDA represents a strong default for policy text also can be run locally without calling any API; LLM paraphrases can be added selectively when the classes have necessity of enlarging its embedding cloud but being sure that this will not create overlapping.

6.5 Future Directions

The study intentionally constrained model size and compute to mirror realistic deployment scenarios, where also it has been tried on. The DA summary portfolio emphasized which ones are label-preserving and low-risk; future work could explore constrained paraphrasing that is policy-aware by design, aiming to balance diversity against drift more precisely. A two-step process could also be implemented, a first “reading” of the policy made by a specifically fine-tuned model that aim to summarize it by knowing which is the label that second model is training and classifying on; the second model could be one of our own, that receivign the right input could truly perform better, focusing just on understanding the feature and not also in finding them, having the discarding of useless pieces of text already done in the first step. Finally, taxonomy design itself is part of the solution: modest revisions to label definitions, or the creation of easily separable classes by diving or joining similar one, may better reflect how investment policies are written and, in turn, improve separability under limited data.

Overall, the study demonstrates a pragmatic recipe for KIID policy classification under realistic constraints: choose the encoder by input-length profile, and use conservative, label-preserving DA to systematically reduce imbalance generated error. Results were good where we expected them to be good, and *not* production ready when the taxonomy was both fine-grained and imbalanced, but even then, DA provided measurable gains.

Table 6.1: **Research questions (RQs) and answers.** RQs as defined in 4.2.

RQ	Answer
RQ1: Model adequacy under local constraints	*Yes.* Compact PLMs (RoBERTa-base; Longformer-base) achieved operationally useful performance within a two-GPU budget. Longformer added value when inputs were long; RoBERTa sufficed on short curated excerpts. (§5.2, §5.4).
RQ2: Effect of imbalance	*Material.* Strong class-support effects and a negative correlation between label granularity (more classes) and macro-F1 were observed; class imbalance is one of the main driver of baseline quality. (§5.2, encoder baselines).
RQ3: DA behavior on our domain	*Good diversity.* DA enlarged class clusters without large centroid shifts in our diagnostics (similarity indices, PCA), indicating added variability with preserved semantics, what we where looking for. (§5.1).
RQ4: DA effect on imbalance	*Positive and consistent.* Conservative DA (STA/AEDA, GPT paraphrases) increased macro-F1 and minority recall across taxonomies without harming micro/accuracy; with larger gains where starting baseline were worse. (§5.3).
RQ5: Which DA works best in finance?	*STA + AEDA and GPT are strong defaults.* Role-aware edits that protect class-indicative tokens plus low-drift noising delivered the most reliable improvements. LLM paraphrases added gains with higher variance and also with label-drift risk (consistent with §3) and with effect shown in §5.3), it need to be used in controlled environment but could perform well.
RQ6: Long context value	*High on long inputs.* A 4096-token window reduced truncation errors and improved performance where evidence was dispersed; the benefit grew with input length, at the cost of higher training/inference time. (§5.4).

Bibliography

[1] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer. In *arXiv preprint arXiv:2004.05150*, 2020.

[2] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. In *IEEE Transactions on Neural Networks*, volume 5, pages 157–166, 1994.

[3] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, et al. Language models are few-shot learners. In *NeurIPS*, 2020.

[4] Tadeusz Calinski and Jerzy Harabasz. A dendrite method for cluster analysis. *Communications in Statistics*, 3(1):1–27, 1974.

[5] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *EMNLP*, pages 1724–1734, 2014.

[6] David L. Davies and Donald W. Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2):224–227, 1979.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 4171–4186, 2019.

[8] Biyang Guo, Songqiao Han, and Hailiang Huang. Selective text augmentation with word roles for low-resource text classification. *arXiv preprint arXiv:2209.01560*, 2022.

[9] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[10] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[11] Akbar Karimi, Leonardo Rossi, and Andrea Prati. Aeda: An easier data augmentation technique for text classification. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2748–2754, Punta Cana, Dominican Republic, 2021. Association for Computational Linguistics.

[12] Varun Kumar, Ankit Gupta, and Jonathan May. Data augmentation using pre-trained transformer models. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pages 18–27. Association for Computational Linguistics, 2020.

[13] Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents. *Proceedings of the 31st International Conference on Machine Learning (ICML)*, pages 1188–1196, 2014.

[14] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohiuddin, Luke Miller, et al. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *ACL*, pages 7871–7880, 2020.

[15] Bohan Li, Yutai Hou, and Wanxiang Che. Data augmentation approaches in natural language processing: A survey. *AI Open*, 2022.

[16] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. In *arXiv preprint arXiv:1907.11692*, 2019.

[17] Andrew McCallum and Kamal Nigam. A comparison of event models for naive bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization*, pages 41–48. AAAI, 1998.

[18] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *Proceedings of the ICLR 2013 Workshop*, 2013.

[19] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *NAACL-HLT*, pages 746–751, 2013.

[20] Thomas P. Minka. Estimating a dirichlet distribution. Technical Report MSR-TR-2000-184, Microsoft Research, 2003.

[21] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[22] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. In *OpenAI Technical Report*, 2018.

[23] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. Technical report, OpenAI, 2019. OpenAI Technical Report.

[24] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. In *Journal of Machine Learning Research*, volume 21, pages 1–67, 2020.

[25] Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987.

[26] Gerard Salton and Chris Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523, 1988.

[27] Rico Sennrich, Barry Haddow, and Alexandra Birch. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany, 2016. Association for Computational Linguistics.

[28] Connor Shorten, Taghi M. Khoshgoftaar, and Borko Furht. Text data augmentation for deep learning. *Journal of Big Data*, 8(1):101, 2021.

[29] Amane Sugiyama and Naoki Yoshinaga. Data augmentation using back-translation for context-aware neural machine translation. In *Proceedings of the Fourth Workshop on Discourse in Machine Translation (DiscoMT 2019)*, pages 35–44, Hong Kong, China, 2019. Association for Computational Linguistics.

[30] Lichao Sun, Congying Xia, Wenpeng Yin, Tingting Liang, Philip S. Yu, and Lifang He. Mixup-transformer: Dynamic data augmentation for NLP tasks. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3436–3440, Barcelona, Spain (Online), 2020. International Committee on Computational Linguistics.

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5998–6008, 2017.

[32] Jason Wei and Kai Zou. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6382–6388, Hong Kong, China, 2019. Association for Computational Linguistics.

[33] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations (ICLR)*, 2018.