POLITECNICO DI TORINO

Master’s degree in Data Science and Engineering

Master’s Degree Thesis

Text Generation through a GPT-based GAN model

Academic Supervisors

Prof. Luca CAGLIERO

Prof. Moreno LA QUATRA

Company Supervisor

Ing. Fabio RAIMONDI

Candidate

Gianluca LA MALFA

April 2023

Abstract

Lack of data represents one of the main problems in the Machine Learning field. The greater part of the algorithms, used for different scopes and goals, needs a huge quantity of data to be trained on. This problem becomes even bigger in the Deep Learning field, because usually, neural networks require more data than classical machine learning algorithms. The latter will reach earlier a point in which new data does not improve the learning capabilities of the model.
Strictly connected to this, there is also the problem of class imbalance that can arise for different kinds of situations, above all in classification tasks. With class imbalance, machine learning models will typically over-predict the most frequent class label due to their increased prior probability. As a result, the instances belonging to the smaller class are typically misclassified more often than those belonging to the larger class. Even if there are already different approaches to solve the problem when working on tabular data, it is still difficult to solve the issue when working with natural language. Being able to generate completely new sentences, and more specifically to re-balance the distribution of data based on the minority class represents an important challenge to face. Generative models are one way of performing text generation. In particular with the introduction of Generative Adversarial Networks (GAN) a new way of approaching the generation problem is born. While in the computer vision field, GANs are already obtaining sensational results, the same does not apply for natural language processing. Although there are different models that are reaching significant results, there is still room for improvements for GANs to be competitive even in this field.
The goal of this thesis work is to try to understand if the GAN architecture can reach comparable performances to GPT,which represents the state of the art,in the text generation field. To do this the financial domain will be object of study, words that in natural language can assume one meaning can be completely different in finance. The goal is to show the robustness and versatility of the proposed models. The presented work is divided into several phases: a first phase of finding and creation of the datasets to work on, a second phase consisting of the implementation of a traditional GAN and comparison with the state of the art approach by analyzing a classification task built on top of the dataset re-balanced through text generation, a third and final phase of building a modified conditional GAN model by exploring the power of GPT as generator and comparison with the state of the art approach in the same settings.
All the results are presented at the end of this work, showing the most important difference in terms of performance, robustness and completeness.

Acknowledgements

Ringrazio i miei genitori, da sempre sono stati la mia guida, il mio faro.Gran parte di questo traguardo lo devo soprattutto a loro.Non smetterò mai di essere loro grato per quanto hanno fatto per me.

Table of Contents

Abstract ii

List of Tables ix

List of Figures xii

1 Introduction 2

1.1 Document outline 3

2 Machine Learning and Deep Learning background 4

2.1 Artificial intelligence 4

2.2 Types of Machine Learning 4

2.2.1 Reinforcement Learning 5

2.3 Datasets 6

2.4 Natural Language Processing 7

2.4.1 Main definitions 7

2.4.2 Text Generation 8

2.4.3 Text Classification 8

2.5 Discriminative vs Generative Models 8

2.6 Main Architectures 9

2.6.1 Recurrent Neural Networks 9

2.6.2 Convolutional Neural Networks 10

2.6.3 GAN 11

2.7 Transformers 13

2.7.1 Attention 13

2.7.2 BERT 14

2.7.3 RoBERTa 15

2.7.4 GPT-3 15

3 Material and methods 17

3.1 Pipeline Overview 17

3.2 Datasets 18

3.2.1 Exploratory Data Analysis 18
3.2.2 Preprocessing 20
3.2.3 Keyword extraction 20
3.3 Text Generation Architectures 22
3.3.1 SEQ-GAN 22
3.3.1.1 Monte Carlo Search 23
3.3.1.2 Architecture 23
3.3.2 GPT-3 24
3.3.3 GPT-2 25
3.3.3.1 Architecture 25
3.3.3.2 How GPT-2 works 26
3.3.3.3 Masked Self-Attention 27
3.3.3.4 Fully connected Neural Network 28
3.3.3.5 Fine Tuning 29
3.3.3.6 Conditional generation on tags 29
3.3.3.7 Unconditional generation 29
3.3.4 GPT-GAN 30
3.3.4.1 Architecture 30
3.3.4.2 Contrastive Discriminator 31
3.3.4.3 Proximally Optimized Generator 32
3.3.4.4 Training Strategy 33
3.4 Classification Architectures 34
3.4.1 Gated Recurrent Unit 34
3.4.1.1 Update Gate 34
3.4.1.2 Reset Gate 34
3.4.2 FinBERT 36
3.5 Deployment 36
3.5.1 Application functionalities 38

4 Results 39
4.1 Experimental settings 39
4.2 Evaluation Metrics 40
4.2.1 Text Generation metrics 40
4.2.2 Classification metrics 42
4.3 Text Generation Results 43
4.3.1 Hyperparameters tuning 43
4.3.2 Unconditional Generation 44
4.3.2.1 Seq-GAN 44
4.3.2.2 GPT-2 Unconditional Generation 46
4.3.3 Conditional Generation 48
4.3.3.1 GPT-2 Conditional Generation 48

4.3.3.2 GPT-GAN 51
4.3.3.3 GPT-GAN vs GPT-2 55
4.4 Text Classification Results 56
4.4.1 Hyperparameters tuning 56
4.4.2 Original dataset 56
4.4.2.1 GRU 57
4.4.2.2 FinBERT 58
4.4.3 Seq-GAN 60
4.4.3.1 GRU 60
4.4.3.2 FinBERT 61
4.4.4 GPT-2 Unconditional Generation 63
4.4.4.1 GRU 63
4.4.4.2 FinBERT 64
4.4.5 GPT-2 Conditional Generation 66
4.4.5.1 GRU 66
4.4.5.2 FinBERT 67
4.4.6 GPT-GAN 69
4.4.6.1 GRU 69
4.4.6.2 FinBERT 70
4.5 Parameter Analysis 72
4.5.1 Number of keywords 72

5 Conclusions and future works 74

Bibliography 77

List of Tables

3.1 Dataset's snippet 18
3.2 Snippet of the modified dataset containing keywords 21
3.3 Snippet of domain adaptation dataset 33

4.1 Generator and Discriminator shared parameters 44
4.2 Discriminator parameters 45
4.3 Generated sentences 45
4.4 GPT-2 best hyperparameters 46
4.5 GPT-2 Unconditional generated sentences 47
4.6 GPT-2 best hyperparameters 48
4.7 GPT-2 Training parameters 48
4.8 GPT-2 Conditional generated sentences 49
4.9 GPT-2 Generator Training parameters 51
4.10 Training parameters 52
4.11 GPT-GAN Conditional generated sentences 54
4.12 GRU training parameters 57
4.13 GRU Base Dataset Classification Report 58
4.14 FinBERT main parameters 59
4.15 FinBERT Base Dataset Classification Report 59
4.16 GRU training parameters 60
4.17 GRU SeqGAN Dataset Classification Report 61
4.18 FinBERT main parameters 61
4.19 FinBERT SeqGAN Dataset Classification Report 62
4.20 GRU training parameters 63
4.21 GRU GPT-2 Unconditional Dataset Classification Report 64
4.22 FinBERT main parameters 65
4.23 FinBERT GPT-2 Unconditional Dataset Classification Report . . . 65
4.24 GRU training parameters 66
4.25 GRU GPT-2 Conditional Dataset Classification Report 67
4.26 FinBERT main parameters 68
4.27 FinBERT GPT-2 Conditional Dataset Classification Report 68

4.28 GRU training parameters 69
4.29 GRU GPT-GAN Conditional Dataset Classification Report 70
4.30 FinBERT main parameters 70
4.31 FinBERT GPT-GAN Dataset Classification Report 71

List of Figures

2.1 Reinforcement Learning overview [1] 6
2.2 Simple RNN Architecture [2] 9
2.3 CNN in NLP enviroment [3] 10
2.4 Example of activation function, ReLU [4] 11
2.5 GAN Architecture [5] 12
2.6 Transformers architecture [6] 13
2.7 BERT architecture [7] 15
2.8 Number of parameters of the most used models [8] 16

3.1 Procedure pipeline 17
3.2 Target distribution 19
3.3 Sentences' statistics 19
3.4 Word clouds 20
3.5 SEQ-GAN Architecture [9] 23
3.6 Performances comparison for zero-shot,one-shot and few-shot learning [10] 24
3.7 Transformer-Decoder Architecture [11] 26
3.8 Starting sequence generation's overview [11] 27
3.9 GPT-2 computations to get first output [11] 27
3.10 Score computation overview [11] 28
3.11 Complete GPT-2 blocks overview [11] 28
3.12 Model's architecture [12] 30
3.13 Contrastive Discriminator [12] 32
3.14 PPO overview [13] 32
3.15 GRU overview [14] 35
3.16 FinBERT pre-training and fine-tuning [15] 36
3.17 Application preview 37

4.1 BERT score procedure overview [16] 41
4.2 Confusion matrix [17] 43
4.3 ReLU vs GeLU activation functions [18] 47

4.4 Perplexity trend for positive and negative models 49
4.5 Rouge-1 and Rouge-L of positive and negative models 50
4.6 BLEU and BERT score of positive and negative models 50
4.7 Perpelxity of positive and negative models 53
4.8 Loss trends of positive and negative models 53
4.9 Rouge-1 and Rouge-L of positive and negative models 54
4.10 BLEU and Bert score of positive and negative models 55
4.11 GPT-GAN vs GPT-2 quality measures 56
4.12 Loss and Accuracy trend of GRU on the original dataset 57
4.13 GRU Base Dataset Confusion Matrix 58
4.14 FinBERT Base Dataset Confusion Matrix 59
4.15 Loss and Accuracy trend of GRU on the SeqGAN dataset 60
4.16 GRU SeqGAN Dataset Confusion Matrix 61
4.17 FinBERT SeqGAN Dataset Confusion Matrix 62
4.18 Loss and Accuracy trend of GRU on the GPT-2 Unconditional dataset 63
4.19 GRU GPT-2 Unconditional Dataset Confusion Matrix 64
4.20 FinBERT GPT-2 Unconditional Dataset Confusion Matrix 65
4.21 Loss and Accuracy trend of GRU on the GPT-2 Conditional dataset 66
4.22 GRU GPT-2 Conditional Dataset Confusion Matrix 67
4.23 FinBERT GPT-2 Conditional Dataset Confusion Matrix 68
4.24 Loss and Accuracy trend of GRU on the GPT-GAN dataset 69
4.25 GRU GPT-GAN Unconditional Dataset Confusion Matrix 70
4.26 FinBERT GPT-GAN Dataset Confusion Matrix 71

Chapter 1

Introduction

One of the main problems arised in Machine Learning is the lack of data. Modern algortihms require a huge quantity of data to be trained on, they learn directly from the data and if there is more data there is a higher chance for a machine learning algorithm to understand it and give accurate predictions to the unseen data. The situation becomes even worse when dealing with Deep Learning algorithms. If we can train Support Vector Machines with some thousands of samples and still obtaining great results, there are neural network architectures that need a quantity of data in the order of hundred thousands. Strictly related to this, there is also the problem of dataset imbalance. Imbalanced data refers to those types of datasets where the target class has an uneven distribution of observations, one class label has a very high number of observations and the other has a very low number of observations. This will cause some problems in classification tasks because the instances belonging to the smaller class are typically misclassified more often than those belonging to the larger class. When dealing with continuous data, there is the possibility of applying traditional re-sampling techniques, but when working on discrete data the same approach can not be applied. In particular, in the natural language processing field, one of the branches that are constantly gaining attention is *Text Generation.* Thanks to Text Generation we can create completely new sentences, that satisfy certain constraints, in order to try to solve both aforementioned problems. The idea is to generate synthetic examples to solve both problems of lack and re-balance of data. In the family of generative models, GANs have received a lot of appreciation, above all for the incredible performances reached in the computer vision field. They are not receiving the same praise in natural language processing because of the different problems this approach suffers from when dealing with this type of data. The idea is to try to understand if the GAN approach can be beneficial also in NLP, and to understand if the performance reached are comparable to the ones obtained by GPT-3, which represents the state of the art in the Text Generation field. The goal of our thesis work is to first find the weak points of the traditional GAN model for text generation and then to study the architecture in order to explore a GPT-GAN using GPT-3 as generator.

1.1 Document outline

1. Chapter 1: *Introduction*: the goal of this chapter is to briefly introduce the scopes and the objectives of the thesis.
2. Chapter 2: *Machine Learning and Deep Learning background*: this chapter aims to remark some machine learning and deep learning concepts, which are fundamental to follow the reasoning of the proposed work
3. Chapter 3: *Materials and Methods*: all the external materials and model that are used are provided and deeply analyzed.
4. Chapter 4: *Results*: exhaustive presentation of all the results obtained, with explanation and comments by the author
5. Chapter 5: *Conclusions*: Final remarks, take-aways and possible future improvements, are reported in this section

Chapter 2

Machine Learning and Deep Learning background

This chapter introduces you to some of the most important and fundamental concepts of Machine Learning and more specifically Deep Learning. The goal is to have all the basic knowledge necessary to follow the path. Starting from the role of artificial intelligence, until more specific architectures which have a key role in the proposed models.

2.1 Artificial intelligence

Artificial intelligence refers to the theory and development of computer systems which aim to perform tasks that usually require human intelligence and cognitive skills [19]. Machine Learning is an application of Artificial Intelligence which goal is to create procedures able to to learn and improve from experience without explicit programming.Models explore the power of data and use it for the learning phase. [20]. Deep Learning is a division of Machine Learning that tries to replicate the way in which humans acquire knowledge. Differently from conventional algorithms, deep learning models are based on a hierarchical organization and work on input data by performing non-linear transformations. The key point of deep learning approaches is the unsupervised learning, which is fundamental to obtain precise and fast to build models. [21]

2.2 Types of Machine Learning

Machine Learning Algorithms can be classified according to different features, but one of the most critical one, is how and what type of data the algorithms manages to learn [22]:

- *Supervised Learning:* The algorithm is trained using input data that has been labeled for a specific output. The model keeps training until it can identify the patterns and connections between the input points and the target output.

- *Unsupervised Learning:* When provided with unlabeled data, the algorithm does not have access to any target attribute, and it has to discover similarities and possibly patterns within the data points.

- *Reinforcement Learning:* In this type of learning, the training phase is based on the determination of the best action by maximizing a defined reward.This approach is iterative, braking away from the previously mentioned ones.

The last type of machine learning approach is a hybrid solution which aims to explore the benefits of both supervised and unsupervised techniques

- *Semi-supervised Learning*: The idea is to use a limited set of labeled data augmented by a huge amount of unlabeled data. The model can learn and generate predictions for new examples

2.2.1 Reinforcement Learning

Reinforcement Learning [23] has a key role in the development of the proposed and analyzed models. It is an approach to machine learning that is about maximizing a reward through different actions. It is up to the reinforcement learning agent to decide the actions to be performed to complete the proposed task. It is a form of learning through trials and experience. From a mathematical point of view, the Reinforcement Learning problem can be formulated through Markov Decision Process Model:

$$max\ E\{\sum_{t=0}^{T} R(s_t, x^{\pi}(s_t))|s_0\}$$

To pursue this goal we can have two approaches called policies [24]:

- Policy iteration: fix a strategy, which is to be followed by the agent to maximize the reward

- Value iteration: maximize the value functions and update the policy rewards.

When we want to evaluate an agent , we generally refer to evaluate how well the agent is performing, following the given policy.

- Reinforce: it is variant of policy gradients based on Monte Carlo. The agent collects a trajectory τ using its current policy, and uses it to update τ. Given that a complete trajectory must be executed to create a sample space, updates are calculated by performing actions (that could be of different kinds) using an off-policy approach.
- PPO: it is a policy gradient method. The Actor-Critic Model FIG.2.1 is the most adopted implementation. It is based on the use of two different neural networks, for the actor(actions) and the critic(rewards). PPO follows a similar procedure to the other policy gradient methods, it compute the output probabilities in the forward pass and the gradients to manage the decisions.

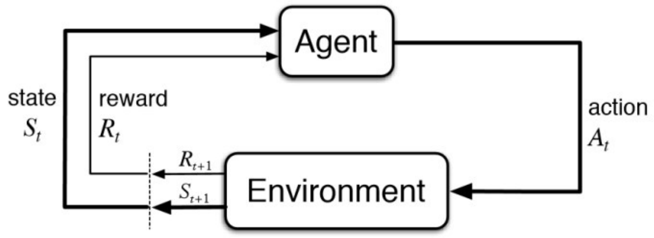


Figure 2.1: Reinforcement Learning overview [1]

In Natural Language Processing, the use of Reinforcement Learning can be crucial, because its application can be exploited for a lot of different aspects. For example, it is possible to use GANs, which are not designed for NLP task, through reinforcement learning. By fixing some elements, as we will see later in detail, there is the possibility of finding a workaround for the discrete and non-differentiable distribution of data used.

2.3 Datasets

As *Daniel Keys Moran* once said, "You can have data without information but you cannot have information without data". This simple but effective quote summarizes how important is to have data in Artificial Intelligence. Datasets are a collection of record that share a common attribute. Usually datasets are composed by more than one attribute, we refer to the this number as the dimensionality. We can have different datasets based on different domains. In machine learning, datasets are usually divided into three splits, which have a key role in the entire learning phase [25].

- *Training*: it represents the sample of data used to fit the model which observes and learns from this data.

- *Validation*: it provides an unbiased evaluation of a model fit on the training data while tuning the model hyperparameters.
- *Test*: it represents the portion of data used to have an unbiased evaluation of the final model. It is used only when the model is complete.

Once the key concepts of Machine Learning are clear, it is possible to introduce more complex notions of deep learning.
Deep Learning architectures can be employed in a lot of different scenarios.Their power is so famous that a lot of different specialized branches of learning are born, such as Computer Vision or Audio processing. In this work, the focus is on Natural Language and more specifically in Deep Natural Language Processing.

2.4 Natural Language Processing

With Natural Language Processing we refer to the branch of Deep Learning which aims to analyze and study natural language. There can be different application, such as sentiment analysis, machine translations, text summarization and more [26]. In this work we will first explore the *Text Generation* task and lately also *Text Classification* to deeply evaluate the effectiveness of the model in a built-on top task.

2.4.1 Main definitions

In this section we will provide some of the key definitions that will be found across the entire work.

- *Language model:* it is at the core of many NLP tasks and it is simply a probability distribution over a sequence of words
- *Corpus:* it is the core element of the analysis in a NLP task. It represents the input consisting of large and structured text.
- *Paragraph*: portion of text composed of different sentences
- *Section*: portion of text consisting of consecutive paragraphs
- *Token*: it is the entity composed by a sequence of characters used as input for the majority of the NLP models. A token can be a word, a symbol or a number.
- *N-gram*: An n-gram is a contiguous sequence of n words.
- *Lemma*: it represents the base form of a word or in general of a token.

- *Stem*: base form of a word to which can be added affixes or suffixes to generate different forms.

2.4.2 Text Generation

Text generation [27] explores artificial intelligence to automatically generate texts (sentences or even entire paragraphs), depending on the context. Since the goal of text generations is to create new sentence, sometimes there is the risk of obtaining out of context phrases. For this reasons it is also plausible to have a sort of guidance in the generation phase, thanks to it is possible to have more accurate results, in this case we talk about *conditional text generation*
Text generation can be performed by different models, such as:

- RNNs
- Transformers
- GANs

2.4.3 Text Classification

Text classification [28] is a machine learning task which goal is to correctly assign a group of labels to the analyzed text. In our thesis work we will perform sentiment analysis which is a task that aims to classify if the given data is positive, negative or neutral. It has proven to have crucial importance, above all in the financial field to control the trend of a specified market.

2.5 Discriminative vs Generative Models

In this section we will explain the main differences between Discriminative and Generative models:

- *Discriminative models*: Given a set of features X and a target variable Y, the goal is to find the conditional probability P(Y|X). The estimation of the parameters of P(Y|X) can be done by exploring the training data. The aim of discriminative models is to separate classes.
- *Generative models*: In this case, to find the probability, it is crucial the estimation the prior probability P(Y) and likelihood probability P(X|Y) with the help of the training data and by using the Bayes Theorem, it is possible to calculate the posterior probability P(Y |X). The main idea of generative models is to explore the likelihood function and the probability estimation to

model data points in order to correctly distinguish different labels in a given dataset. The key difference with discriminative models lies in the possibility of generating new data points.

To summarize, a generative model is about understanding how data is generated while the discriminative counterpart is focused on the prediction of the correct category.

2.6 Main Architectures

In this section we will briefly introduce some of the architectures used during our thesis work.

2.6.1 Recurrent Neural Networks

Recurrent Neural Networks (RNN) are among the most used neural networks in the deep learning field.They are suitable to process time-series data or generally sequential data. Its principal attribute is that the previous step's outputs are used as inputs for the current step. The key point of the RNN is the *hidden state*, which is fundamental to recall all the information about a sequence. They have a sort of memory to remember all information about previous computations.

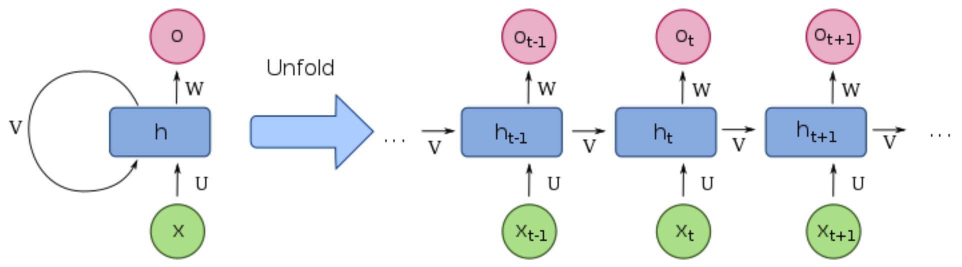


Figure 2.2: Simple RNN Architecture [2]

We can distinguish different types of RNNs:

- One to one: there is a single pair of nodes
- One to many: a single input can produce multiple outputs
- Many to one: many inputs produce a single output

- Many to many: there can be different possibilities, in general many inputs produce many outputs.

RNNs offer numerous benefits, first of all it is possible to handle sequential data of different length, or to recollect historical information as well.The weak points are related to the slow computation phase when dealing with huge amounts of data and above all to the problem of vanishing gradient.

With **vanishing and exploding gradient** we refer to one of the most common problem of deep learning applications. The gradients used to compute weights may get very close to zero or to infinity, this prevents the network from learning new weights. The deeper the network is the more probable it is for this problem to appear.

2.6.2 Convolutional Neural Networks

Convolutional Neural Networks (CNN) are really famous for their applications in the Computer vision field, but they showed to be really competitive in NLP as well.
A CNN is composed by *convolutional layers* and *downsampling* layers.

- The goal of the convolutional layers is to use their neurons to scan the provided input for patterns
- Downsampling layers, also known as *pooling layers* are used to reduce the feature map dimensionality in order to improve computational efficiency.

Typically the two aforementioned layers occur in alternated order.

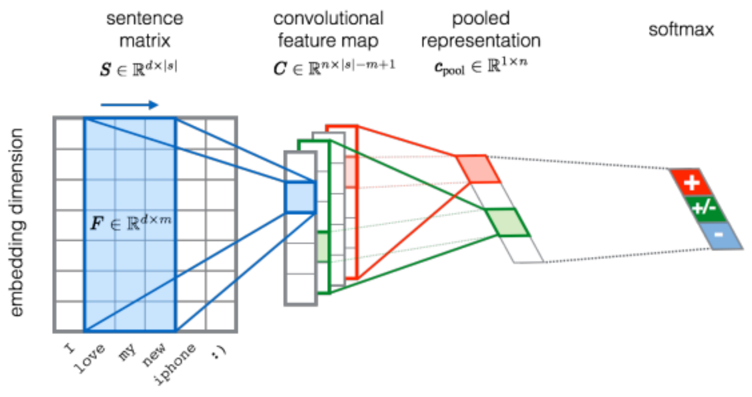

sentence matrix
$S \in \mathbb{R}^{d \times |s|}$
convolutional feature map
$C \in \mathbb{R}^{n \times |s|-m+1}$
pooled representation
$c_{\text{pool}} \in \mathbb{R}^{1 \times n}$
softmax
embedding dimension
$F \in \mathbb{R}^{d \times m}$
+
+/-
-
I
love
my
new
iphone
:)

Figure 2.3: CNN in NLP enviroment [3]

The processing phase of a CNN is as follows: the feature matrix is used as input, it can be from images or in our study case sentences, make it pass through

convolutional layers, activation layers (ReLU) and pooling layers and finally reach a Fully Connected layer before applying an activation function, for example the *softmax* on the logits to obtain output values between 0 and 1.
An **activation function** [4] is a particular function that is usually integrated in a neural network, which operates on the output of a neuron. It determines the outcome based on the input and it permits to add non-linearity as well.

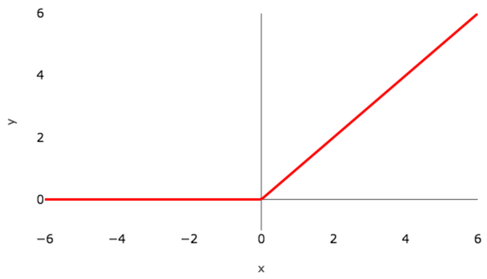


Figure 2.4: Example of activation function, ReLU [4]

Some **Pooling layers** are used to reduce the dimension of the feature map, by summarizing all the feature in a particular region. There are different kind of pooling operations:

- Max pooling: the maximum element from the region is selected, the output will be a feature map containing the most important features of the previous map
- Average pooling: the average of the elements of the region is returned, the output will be a feature map containing the average of the features.

A **Fully Connected Layer (FC)** is typically the last layer in a CNN,it is obtained by a combination of affine and non-linear functions. As the name suggests, the main idea is to connect every neuron from the previous layer to the ones in the following. This is useful to perform complex and non-linear transformations.

2.6.3 GAN

Before the introduction of GANs in 2014 with the paper "Generative Adversarial Nets" [29], generative models were not as used as their discriminative counterpart, because of various difficulties of approximation in the computation of the joint probabilities. In *Generative Adversarial Networks* the generative model is joined by an adversary, a discriminative model which aims to determine if an example belongs to the real distribution of data or to the "fake" one. The goal of the *Generator* is to create a fake sample which is as realistic as possible while the aim of the *Discriminator* is to distinguish the generated samples from the real ones.

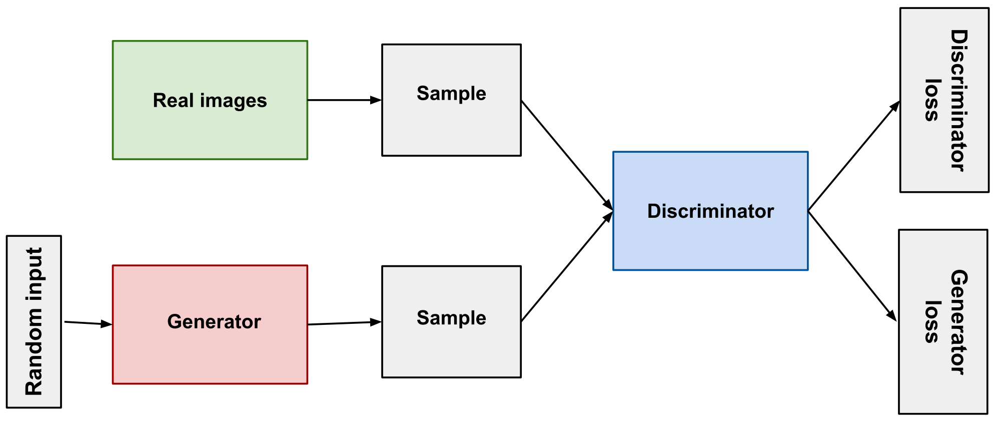


Figure 2.5: GAN Architecture [5]

The objective function of the two agents is opposite, when one wins the other one loses. The feedback shared between the two is fundamental, because on the base of the answers given by the discriminator, the generator improves his production of fake samples. At some point, the discriminator, that at each iteration will improve his capabilities of detection, will not be able to distinguish real samples from fake ones, ending the training.
To learn the distribution of the generator p_g , an input noise $p_z(z)$ and a mapping function on the data space $G(z;\theta_g)$, in which G is a differentiable function guided by parameter θ_g , are defined. The discriminator model $D(z;\theta_g)$ outputs a scalar, which represents the probability that the sample x comes from the real distribution of data rather than from the distribution of generated data p_g. D is trained to maximize the probability of assigning the correct label to the input sample, while G to minimize $log(1 - D(G(z))$. The competition between the two networks can be expressed as:

$$min_G max_D V(D,G) = Ex\ p_{data}(x)[logD(x)] + Ez\ p_z(z)[log(1 - D(G(z)))]$$

Over the years the application of GANs has reached incredible results, above all in Computer Vision. The same does not apply in Natural Language processing, because of some problems related to the discrete nature of the data used. Because of this it is not possible to use the classical GAN architecture, there must be applied some modifications in the learning phase.

2.7 Transformers

Recurrent Neural networks have achieved very good results but as already anticipated, they suffer from performance degradation when dealing with long sequences and huge amount of data. Transformers [6] aim to establish the dependence between inputs and outputs by replacing "recurrence" with "self-attention". The latter makes it able to compute the target word as a combination of vectors, on the contrary the former relies just on the decoder's hidden state. Self-attention creates a connection between the hidden states of each encoder and decoder. As remarked, the architecture is a typical encoder-decoder:

- Encoder: composed by a stack of six identical layers, each layer has two layers: the first is the multi-head self attention while the second is a positional fully connected feed-forward network. Residual connections are employed for each of the two sub-layers with a layer normalization.
- Decoder: The decoder has the same structure, except the addition of a third sub-layer to perform multi-head attention over the output of the encoder stack.

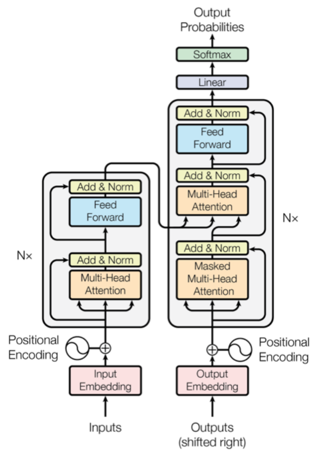


Figure 1: The Transformer - model architecture.

Figure 2.6: Transformers architecture [6]

2.7.1 Attention

The revolution in the transformer architecture is the attention mechanism [6], which is defined as a mapping from a query and a collection of key-value pairs to an output. The output is computed as a weighted sum of the values, in which the weights are modelled by a function of the query and its corresponding key. What it is done is simply computing dot products, but by adding the key,value and query parameters we also have the possibility of training to improve these weights. Given Q,K,V we have:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

Instead of performing a single attention function it is possible to parallelize the process with h-heads, so we have the *multi-head attention*. Multi-head attentions is used to allow the model to manage information from different feature spaces. On the contrary, the single-head mechanism does not allow this.

$$MultiHead(Q, K, V) = Concat(head_1, ..., heand_h)W^O$$

$$where\ head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

2.7.2 BERT

Traditional language models operate unidirectionally, this can be problematic when dealing with tasks that require context from both directions. Bidirectional Encoder Representation from Transformers (BERT) [7] tries to get solved the unidirectionality constraint by introducing a Masked Language Model pre-training objective. This approach randomly masks some of the tokens in the document, the goal is to predict the missing tokens. There are two fundamental steps:

- **Pre-training:** the model is trained on unlabeled data from a downstream task. The pre-training phase is done by performing two unsupervised tasks:
  - *Masked Language Model (MLM)* : The only way to perform a bidirectional training is to mask a fixed percentage of the tokens
  - *Next Sentence Prediction (NSP)*: Understanding the relationship between two sentences is fundamental. More specifically, given two sentences A and B, we aim to understand if B is actually following A
- **Fine-tuning:** each downstream task has separate fine-tuned models. The intuition is that final layers learn features which are strictly related to the context of use. While the initial layers are frozen, these ones are trained to adapt to the specific task.

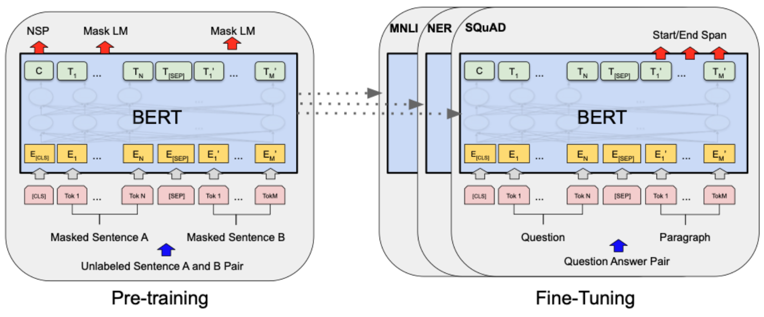


Figure 2.7: BERT architecture [7]

2.7.3 RoBERTa

RoBERTa represents an important study on the design of the famous BERT architecture, with the aim of understanding which concepts contribute the most to the power of the model and what decisions could be made for further improvements. In the original BERT implementation, the *masking* operation was done once, resulting in a *static* mask. This strategy was compared to a *dynamic* masking where the masking pattern is generated every time a sequence is fed to the algorithm. [30] Moreover, a study regarding the importance and influence of the NSP task has been carried forward, and surprisingly it was demonstated that removing the NSP loss leads to a slight improvement of the performances. This optimizations showed to outperform the traditional BERT architecture, bringing a lot of advantages both in terms of resources and results.

2.7.4 GPT-3

So far, we have learned that a model can adapt to a specific task by applying fine-tuning. This seem to be quite far from how the human brain acquires the ability of completing new tasks. Tipically humans do not require large amount of examples to start becoming proficient in different tasks. Moreover, humans are usually multi-tasking, they do not have to specialize on only one, they are capable of doing multiple activities, especially language related ones. GPT-3 is an autoregressive language model trained on 175 billion parameters as it is possible to see in FIG 2.8. It has shown impressive results on different tasks, and above all on few shot learning. The number of examples needed for the specialization on the task is very limited. In terms of its architecture, this model keeps the same structure as its predecessor GPT-2, which includes the modified initialization, pre-normalization, and reversible tokenization. However, there is a small difference in the transformer layers, dense and locally banded sparse attention patterns are alternated.

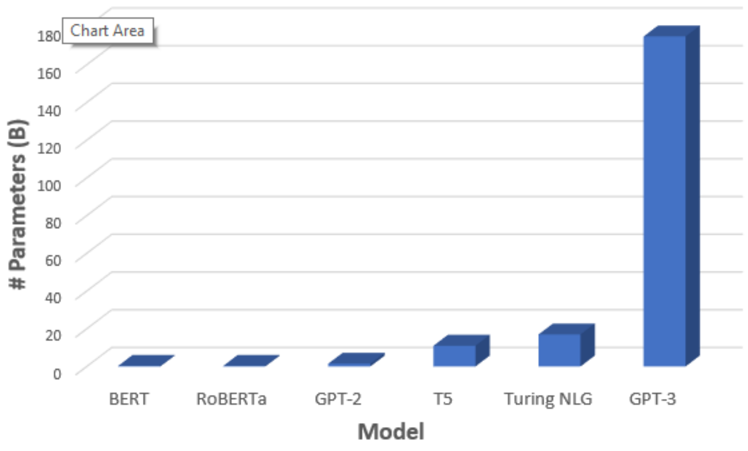


Figure 2.8: Number of parameters of the most used models [8]

Chapter 3

Material and methods

In this section we will report all the materials and methods applied through our entire thesis work.

3.1 Pipeline Overview

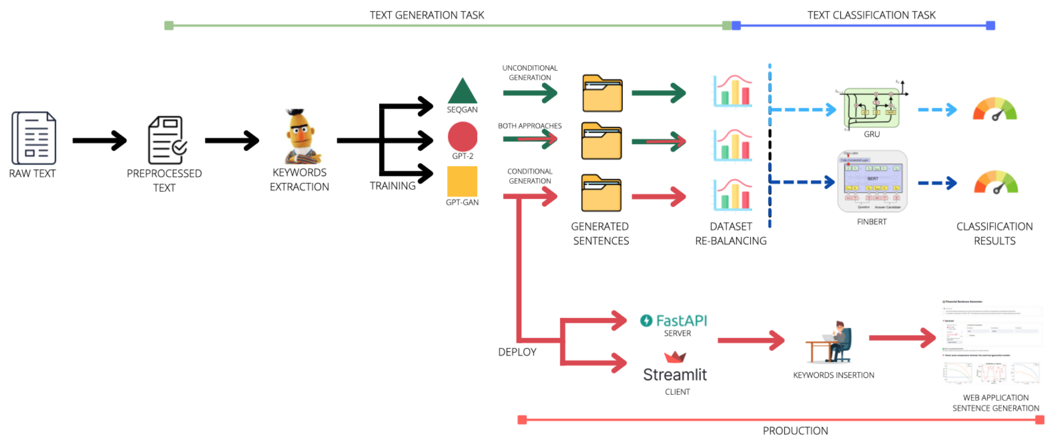


Figure 3.1: Procedure pipeline

As it is possible to see in FIG.3.1, starting from the raw text of the object of study dataset 3.2, after some preprocessing steps, the goal is to prepare the input for the Conditional Generation, or eventually use it as it is for the Unconditional generation.Three architectures will be presented SeqGan 3.3.1, GPT-2 3.3.3 and GPT based GAN 3.3.4. After the generation of their respective sentences, these last will be used to create some rebalanced dataset that will be used for the training of the two proposed classifiers GRU 3.4.1 and FinBERT 3.4.2. In addiction, as it can be noticed from the bottom of FIG.3.1 there is also the deployment phase 3.5, which aims to obtain an application to easily visualize the functioning of the proposed model in different scenarios.

3.2 Datasets

One of the key points of our thesis work was to create and process a dataset. In this case, we decided to work on the financial domain, considering its possible future applications and its complexity. In fact, dealing with words related to finance is slightly different from dealing with words of every other context. Terms that in our everyday language have a particular meaning could have a completely different one in financial settings. We decided to work on this field, being aware of its pitfalls but conscious of the possibility of adapting the pipeline to different context by simply apply some modifications in the preprocessing phase. Throughout our entire work we used data coming from various sources for different scopes and goals. For the sake of simplicity we will report as principal dataset the one on which we performed the task of text generation and classification, we will cite all the other sources used in the respective section in which they are used.

3.2.1 Exploratory Data Analysis

The dataset FinancialPhraseBank [31] contains the sentiments for financial news headlines from the perspective of a retail investor. As it is possible to see in Table 3.1, it contains two columns:

- Sentiment: it indicates the feeling that the news headline expresses
- News Headline: it contains the corpus of the entire news headline

News Headline	**Sentiment**
Technopolis plans to develop in stages an area of no less...	Neutral
The international electronic industry company Elcoteq has laid off...	Negative
With the new production plant the company would increase...	Positive

Table 3.1: Dataset's snippet

The dataset has been chosen because it fits perfectly our intentions of performing both text generation and text classification tasks. As it is possible to notice in FIG.3.2 the distribution of the target value "Sentiment" is strongly imbalanced towards the *neutral* class. Our aim is to perform the text generation task to re-balance the dataset and the text classification one to assess the performances.

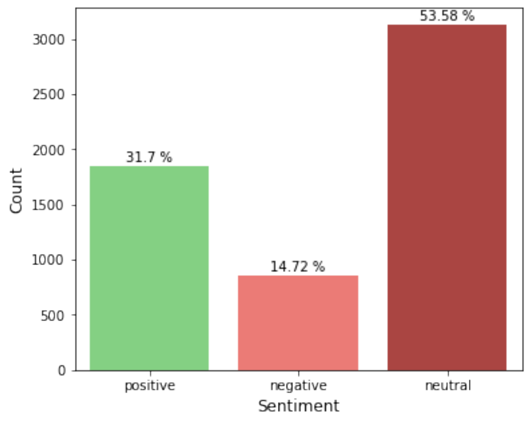
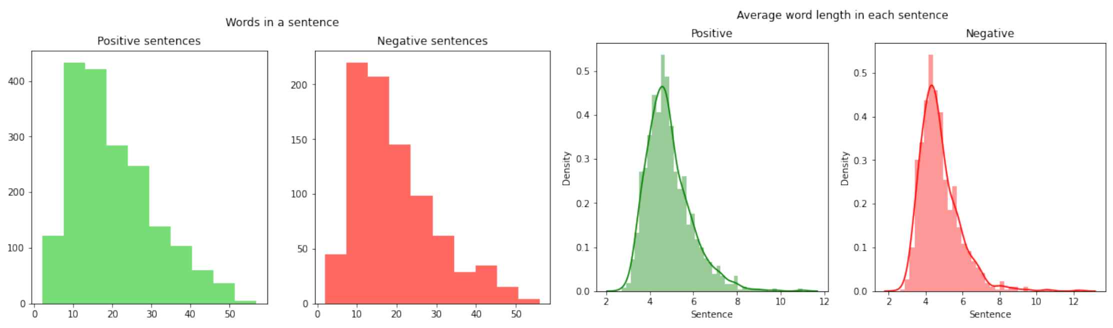


Figure 3.2: Target distribution

To understand the composition of the dataset we decided to deeply analyze it and plot some statistics.

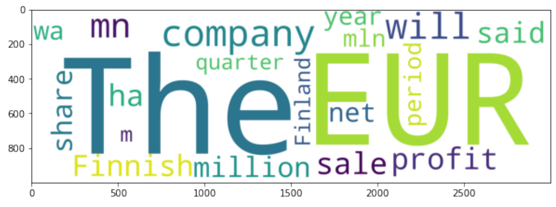


(a) Number of words in a sentence **(b)** Average word length in each sentence

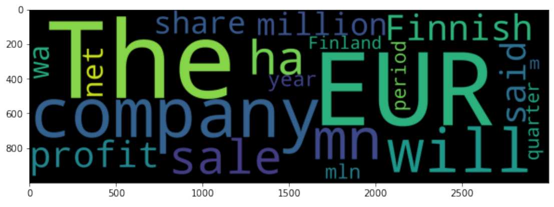
Figure 3.3: Sentences' statistics

Positive sentences tend to contain more words than negative ones, while as concerns the average word length there are no noticeable differences as it can be seen in FIG.3.3a and FIG.3.3b. Another factor we decided to investigate was the presence of repeated words, and in general which were the most commons.

As it is notable, we have a lot of common language words, known as stopwords, but considering the nature of the task we will perform we are not authorized to remove them. FIG.3.4a and FIG.3.4b show how there is no huge difference in the most used words in both negative and positive cases.



(a) Most common words in Positive sentences



(b) Most common words in Negative sentences

Figure 3.4: Word clouds

3.2.2 Preprocessing

Data is coming from financial reports, so it is necessary to clean the sentences to make them closer to natural language. Considering that our aim is to perform a text generation task, it is important to apply only the necessary preprocessing techniques. For example, we will not apply any stemming or stopwords removal because it will prevent our model to generate substantial tokens fundamental for the composition of the final sentence. Proceeding by order, we first checked the language of all the sentence contained in the dataset. We will work only with *English* sentences, but it is plausible to make some modifications to work also with other languages. After language selection, our objective was to clean all the sentences from non-alphanumerical and external resources such as links or tags. By doing this we had to deal with simple and clean financial reports. Finally, we tried to simplify the financial context by converting the most known symbols and abbreviations into natural language (€ in Euro). After sentence cleaning we had to deal with the categorical values. A categorical variable is one that has two or more categories. [32].
Machine learning models can only work with numerical values, so it was necessary to encode the *sentiment* attribute. Considering that it can assume only three values, we decided to perform a simple mapping.

3.2.3 Keyword extraction

As we will see more precisely in the following sections, our main goal is to perform a conditional generation task through a GPT based GAN model. Conditional generation showed to have better performance compared to the unconditional counterpart. There are different ways to condition the generation, but for this particular case we decided to use keywords. By definition, a keyword is a word of great importance in a sentence. In our case we decided to extract the top 3 keywords in each sentence to create the training set which will guide our model during learning phase. To extrapolate the keywords we used **KeyBERT**.

KeyBERT [33] is among the most famous and easy-to-use algorithms to extrapolate keywords or keyphrases that are most similar to a document. As it can be imagined, it explores the BERT embeddings and operates as:

1. Extraction of document embeddings
2. Extraction of word embeddings for n-grams, words or phrases
3. Computation of the cosine similarity to find n-grams that are most similar to the document, they are the most representative words of the entire document.

News Headline	Keywords	Sentiment
Technopolis plans to develop...	['technopolis', 'telecommunication', 'develop']	Neutral
The international electronic industry...	['international ', 'industry', 'employees']	Negative
With the new production plant the company...	['production', 'increase', 'capacity']	Positive

Table 3.2: Snippet of the modified dataset containing keywords

3.3 Text Generation Architectures

In this section we will show which architectures have been used during the thesis work. Even if the proposed and central model is the GPT based GAN, we decided to first present the base models on which the latter is based on. For this reasons, we will first present SEQ-GAN, which is the first model of this type applied on a NLP setting, then GPT-2 and finally the GPT based GAN model. Even if Large Language Models have already gained incredible reputation, they suffer from the *exposure bias* [34] in the inference stage: the difference between training and test performances is due to the fact that the the distribution of data of the former can be really far from the one of the latter. In the text generation task this can be a huge problem because the model will have to work with tokens that have not been seen during the learning phase.

3.3.1 SEQ-GAN

Sequential-GAN was born in a historical moment in which GAN based approaches applied in the NLP field were gaining more and more criticism. This is due to the fact that GANs were achieving incredible results in Computer Vision while they were still not applicable in NLP task due to the discrete nature of data treated. In fact, traditional GANs could be used only on continuous and differentiable data. The small changes made during every iteration on images were not possible on text corpus because there was no corresponding token to them.
With the work presented by Lantao Yu et Al. [9] there is a complete change. By exploring the power of Reinforcement Learning, it is now possible to use GANs on textual data. With RL we are able to bypass the generator differentiation problem by directly performing the gradient policy update.
The **Reward** comes from the GAN discriminator (complete sequence) and is passed back to the intermediate state-action steps using Monte Carlo search. The *state* is the generated tokens until that moment and the *action* is represented by the next token to be generated.

To address the problem of the back-propagation of the gradient when,like in our study case, the output is discrete, the generative model is considered a stochastic policy.Specifically,Monte Carlo (MC) search is used to estimate the state-action value. We train the policy directly using the **policy gradient method** [35], which is useful to overcome the differentiation constraint for discrete data encountered in traditonal GANs.

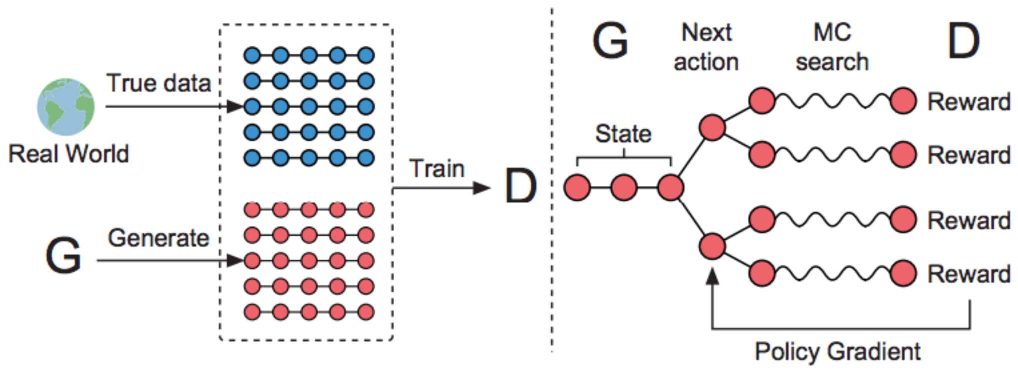


Figure 3.5: SEQ-GAN Architecture [9]

Given a dataset of real-world sequences:

- Train a θ parameterized generative model G_θ to produce a sequence $y \in Y$ where Y is the vocabulary
- In timestep r the state s is the current produced tokens and the action a is the next token to select.
- We also train a ϕ parameterized discriminative model D_ϕ to provide a guidance for improving generator (discriminator) $G_\theta.D_\phi$ indicates the probability of how likely a sequence is from real sequence data or not.

3.3.1.1 Monte Carlo Search

Monte Carlo Search [36] is an heuristic based approach which explores the power of reinforcement learning.

The main idea is to generate random samples from the current position and use them to evaluate different moves. The core process is represented by the simulation, starting from the current state it is possible to complete the path and register the eventual success of failure. The goal is to predict the probability of an event or the value of a function by using a simple distribution of random generated samples. As already anticipated, this approach is very powerful to solve very complex problems.

3.3.1.2 Architecture

- Generator: RNN, the softmax function is used to map the hidden states into the final token distribution. Moreover the LSTM cells are forced to implement the update function g in order to address the vanishing gradient issue.

- Discriminator: CNN, because it has shown to be fast and accurate even on text classification tasks.It is also considered the case in which the discriminator predicts the probability that a finished sequence is real.

3.3.2 GPT-3

GPT-3 [10], is a neural network trained using an immense quantity of data to be able to handle and understand almost any type of input text. Created by OpenAI, it only requires a small amount of input text to produce large and well written, both from the syntactical and semantic point of view, text. The key idea is that larger models are better at managing contextual information, which is critical for the positive outcome of the learning process. Even if fine-tuning could improve the already impressive performances of the standard model, the core idea is to provide one model for different use cases trying to stem one of the main problems of its predecessor GPT-2, the difficulties of working on niche domains. When GPT-3 was initially introduced, it went through different tests, performed for various task under several domains.They can all be summarized under three categories:

- Few-shot : the model predicts the answer given only a natural description of the task and some examples.
- One-shot : the model predicts the answer given only a natural description of the task and a single example.
- Zero-shot : the model predicts the answer given only a natural description of the task.

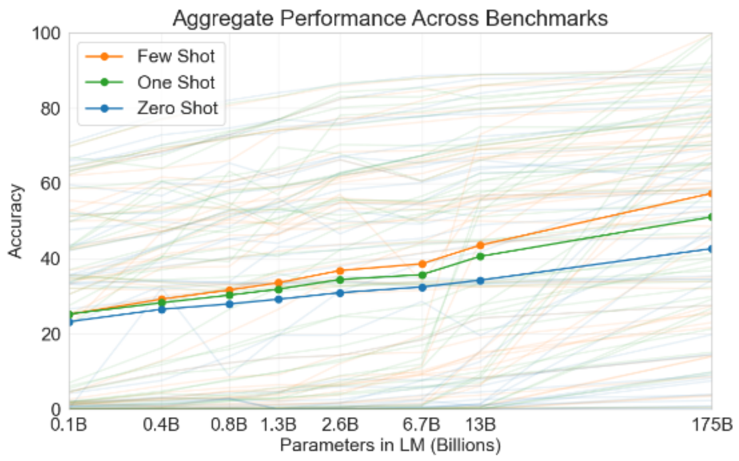


Figure 3.6: Performances comparison for zero-shot,one-shot and few-shot learning [10]

As it can be seen from FIG.3.6 as the number of parameters increases the capability of few-shot learning rises as well.This demonstrates that larger models are more proficient at in-context learning. Even if GPT-3 has shown incredible results, its computational heaviness makes it impossible to work without adequate infrastructures. Also considering that the source code of the model can not be accessed and only an API provided by OpenAI is available, we decided to not go further in this direction and to only concentrate on its predecessor GPT-2 which almost shares the same architecture but with a significant inferior number of parameters. All the results that we will be obtained should be improved using the largest version GPT-3.

3.3.3 GPT-2

GPT-2 [37] is a large language model with 1.5 billion parameters, trained on a dataset composed by more than eight million web pages. It's architecture is transformer based. There is also the possibility to access to smaller models with a significant minor quantity of training parameters.In our case, we used GPT2-small composed by 117M parameters. Following a traditional path, NLP tasks are faced through supervised learning, using a huge quantity of data. As it can be imagined, trying to adapt the model to multi-task learning can become even more difficult and computationally heavier, with the need of a greater quantity of training data. This is the cause of the necessity of finding alternative models to perform multitask learning. As language naturally follows a sequential order, it is conventional to split joint probabilities of symbols into the product of conditional probabilities. To complete a single task, it is suitable to use a probabilistic framework based on the estimation of the conditional distribution.

$$p(output|input)$$

Considering that we have to perform many tasks, even for the same input, it should be conditioned not only on the input but also on the task to be performed

$$p(output|input, task)$$

3.3.3.1 Architecture

As already showed in 2.7 the **initial** transformer paper [6] introduced two blocks:

- Encoder: It is composed by two layers which are the *Feed Forward Neural Network* and the *Self-Attention* blocks. It can take inputs up to 512 tokens.
- Decoder: It presents a small variation from the encoder block with the introduction of the *Masked Self Attention*

Later, with the paper "Generating Wikipedia by Summarizing Long Sequences" [38] there was the introduction of a modified transformer architecture in which the Encoder was completely removed. This new architecture is called **Transformer Decoder** and it's reported in FIG.3.7. As it is possible to see, it maintains a structure which is similar to its predecessor, but removes the self-attention block. As suggested, GPT-2 is implemented by a stack of decoder blocks, on the contrary, BERT is composed by encoders only.

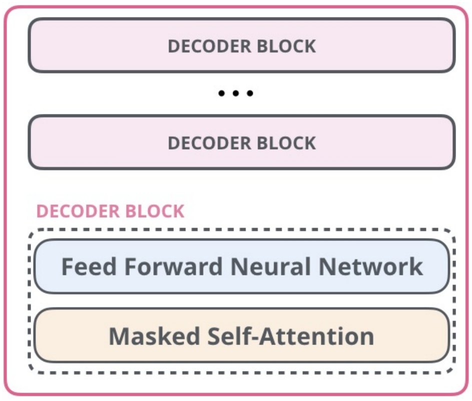
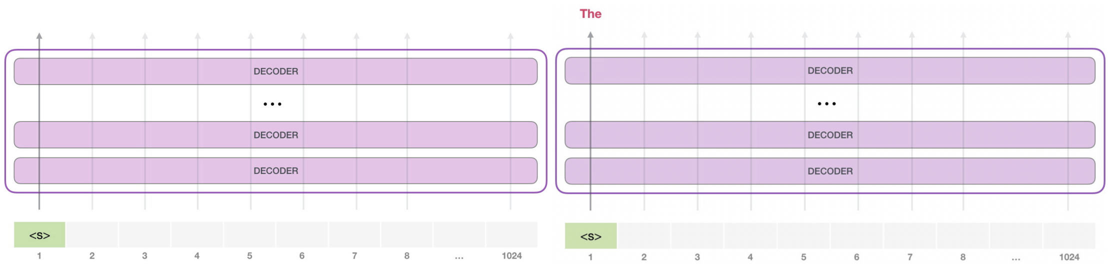


Figure 3.7: Transformer-Decoder Architecture [11]

3.3.3.2 How GPT-2 works

Given an input token, it is processed through all layers, a vector is produced and it is scored with respect to the vocabulary, which for this particular model has maximum size set to 50.000 words. After the scoring function, the selection of the most important token is done following different rules, normally the one with highest probability is chosen but different criteria can be applied. Once the first iteration is completed, the output is simply added to the input for the next iteration and the entire process is repeated.

More specifically, the model retrieves the embedding of the preceding token from the embedding matrix. Before sending it to the initial block, a positional embedding is attached to underline the word order in the sequence.
After the top block of the model generates its output vector (which is the result of the neural network processing plus the self-attention),the model multiplies this generated vector with the embedding matrix. Each row in the embedding matrix represents the embedding of a word in the model's vocabulary, thus as a consequence, this multiplication is considered as a score for every word in the model's vocabulary. For the token selection, it is also possible to consider multiple words, this choice could lead to even more precise results. In fact, it is suggested to randomly picking a word from the list, with the score indicating the likelihood



(a) Start of Transformer decoder generation **(b)** First iteration of transformer decoded generation

Figure 3.8: Starting sequence generation's overview [11]

of the word being chosen. [11]
All the steps are summarized in FIG.3.9

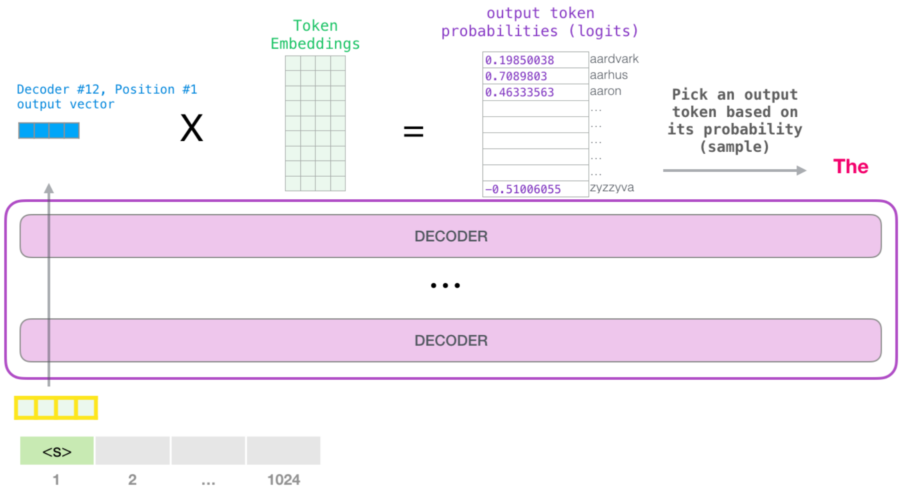

Decoder #12, Position #1 output vector
X
Token Embeddings
=
output token probabilities (logits)
0.19850038 aardvark
0.7089803 aarhus
0.46333563 aaron
-0.51006055 zyzzyva
Pick an output token based on its probability (sample)
The
DECODER
...
DECODER
<s>
1
2
...
1024

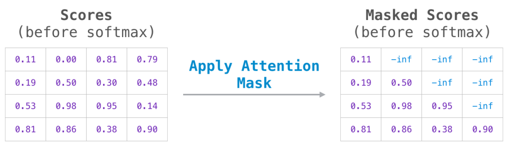
Figure 3.9: GPT-2 computations to get first output [11]

3.3.3.3 Masked Self-Attention

Masked Self-Attention is a variation of the traditional Self-Attention concept. Originally, the Self-Attention block was articulated on three main action:

- *Creation* of Query, Key, and Value vectors
- *Scoring* for each input token
- *Sum* of the value vectors obtained from the product of their corresponding scores.

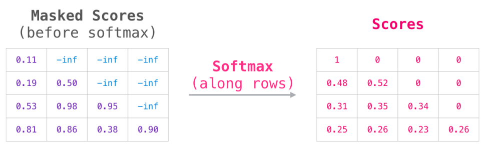
In **Masked Self-Attention** we follow the same path except for the second step. A fixed percentage of the tokens is masked. The masking is implemented as a matrix called *attention-mask*. The scores are calculated by multiplying the queries matrix by the keys matrix. After this multiplication, as it can be noticed from FIG.3.10 the cells we want to mask are set to -inf. Finally the final results are projected by using a softmax activation function. The entire process is shown in FIG.3.10a and FIG.3.10b



0.11	0.00	0.81	0.79
0.19	0.50	0.30	0.48
0.53	0.98	0.95	0.14
0.81	0.86	0.38	0.90

0.11	-inf	-inf	-inf
0.19	0.50	-inf	-inf
0.53	0.98	0.95	-inf
0.81	0.86	0.38	0.90

(a) Attention mask application



0.11	-inf	-inf	-inf
0.19	0.50	-inf	-inf
0.53	0.98	0.95	-inf
0.81	0.86	0.38	0.90

1	0	0	0
0.48	0.52	0	0
0.31	0.35	0.34	0
0.25	0.26	0.23	0.26

(b) Softmax to obtain actual scores

Figure 3.10: Score computation overview [11]

3.3.3.4 Fully connected Neural Network

After the self-attention step, the input token is processed by a fully-connected neural network that consists of two sub-layers [11]:

1. Initial layer: It is of the size of the model multiplied four times(for example for GPT2-Small $768 * 4 = 3072$). This gives transformers enough capacity to handle the majority of the tasks

2. Projection layer: the outputs of the initial layer is projected back into original model's dimension.

FIG. 3.11 reports the detailed structure of the entire GPT-2 model block.

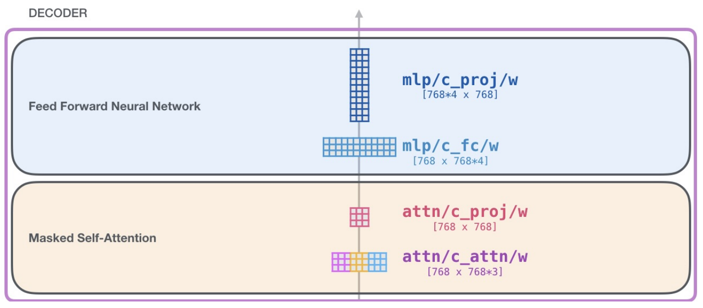


Figure 3.11: Complete GPT-2 blocks overview [11]

3.3.3.5 Fine Tuning

The original GPT-2 model was trained on 40 gigabytes of text ranging across many subjects. Considering this, its performances are very good for text generation but it is possible to have a further improvement by applying fine-tuning. With the term Fine-tuning we refer to the operation of unfreezing certain layers (usually top layers) of a pre-trained model to perform feature extraction and consequently training the model with the added unfrozen top layers. Fine-tuning is based on the idea of making small changes to a pre-exisiting model to better focus on a assigned specific task. If the new task is affine to the original one, by utilizing pre-trained networks, it is possible to explore the feature extraction of the top layers without the need of building it from scratch. [39] With fine-tuning we aim to solve one of the main pitfalls of the GPT-2 model which is the poor performances obtained on specific domains far from common language.

3.3.3.6 Conditional generation on tags

Conditional text generation is a fundamental task in natural language generation. The goal is to generate text given some pre-conditioning with the scope of guiding to a more precise and semantic aware text generation. The original GPT-2 model generates sentences by completing given input sentences. In our case we slightly modified the training phase in order to generate text from an input consisting of up to three keywords:

$$< BOS > + < KEYWORDS > + < EOS >$$

By combining fine-tuning and conditional generation we think we are able to generate semantic appropriate sentences without so strict input conditions.

3.3.3.7 Unconditional generation

GPT-2 also gives the possibility of generating unconditional text by providing only the start of string token $< |endoftext| >$ as input. As a result the generated sentences will contain the semantic information learned through the fine-tuning phase, but they will be more inaccurate because of the absence of a strong guiding input signal.

3.3.4 GPT-GAN

Inspired by the work of Qingyang Wu *TextGAIL* [12], a gpt-based gan model is presented.It aims to incorporate a so important large language model such as GPT-2 with the benefits of the GAN architecture. The objective is to improve the performance obtained in inference phase by classical language models trying to overcome the exposure bias problem.
The most used approaches for text generation are based on the maximization of the probability of the target text sequence (Maximum Likelihood Estimation MLE), but as already stated, it suffers from exposure bias which will cause a huge downgrade of the performances at inference stage. Solving this problem could be crucial for a further improvement of the already good performance that large language models already have.
One of the possible path to follow to solve this issue is by using GANs. The idea is to have a conditional GAN which can explore the power of large language models as generator and discriminator as well. With conditional generation, we aim to use the guidelines provided by the input keywords to have a stronger and more robust generation which could suit the context for which the model has been trained for.

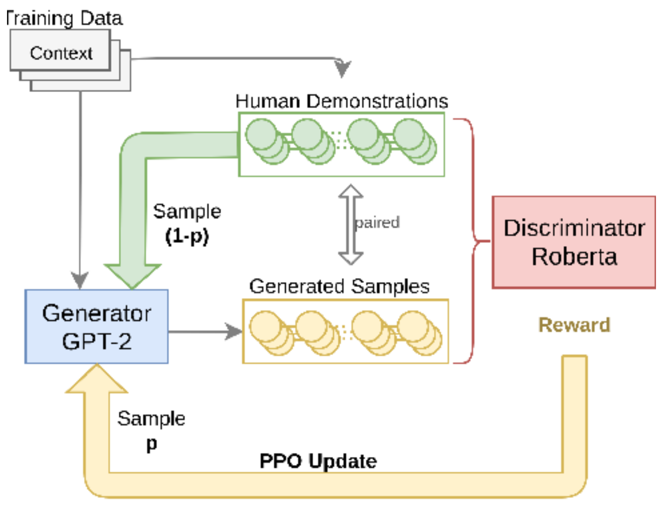


Figure 3.12: Model's architecture [12]

3.3.4.1 Architecture

The model is an extension of "Generative Adversarial Imitation Learning"(GAIL) [40]. The GAN model consists of a generator G_θ and a discriminator D_ϕ. As it happens in the traditional GAN, the aim of the generator is to output sequences as similar as possible to human generated ones, while the goal of the discriminator

is to be able to discern fake sequences from real ones, providing a reward for each generated sequence. With respect to classical GAIL the state s is substituted by the text generation prompt x while the corresponding action a by the target sequence y.
Generator and discriminator should satisfy the optimization problem:

$$min_{G_\theta} max_{D_\phi} \; E_{preal}[D_\phi(x,y)] + E_{G\theta}[1 - D_\phi(x, G_\theta(x))]$$

To stabilize the training an **imitation replay** is introduced (Paine et Al. 2019). The replay buffer is used to store past information regarding the interaction between an agent and the environment. Generally the data stored on it is used to update the policy during the training phase. In this specific implementation, the main idea is to fill the replay buffer by using ground truth sequences during the training of the generator model. The reward for this sequences, which will be treated as a set of normally generated sentences, is fixed to be constant.
To summarize, the main models employed for this architecture are:

- Generator : GPT-2 small (117M parameters) [3.3.3]

- Discriminator : RoBERTa base (125M parameters) [2.7.3]

3.3.4.2 Contrastive Discriminator

As already stated, the goal of the discriminator is to distinguish real sequences from fake generated ones. In the classical architecture, the discriminator uses the *logistic loss (sigmoid)*, but it suffers from early saturation. To overcome this problem, the idea is to slightly modify the discriminator capacity of finding fake samples by estimating the relative realness of the sequences. There is no more a simple output, but a value to indicate how much a real sequence is more realistic than a generated one.
Softmax cross-entropy loss is used to perform the prediction. Now the discriminator analyzes the pair of sentence composed by the real sequence and its generated counterpart and returns a result value indicating the relative goodness of the generated with respect to the real sentence.

$$D_\phi(< x, y_r >, < x, y_g >)$$

The model can be trained with cross entropy loss to maximize the probability for the real sequence p_r while the probability of the generated series p_g is used as reward to guide the training of the generator.

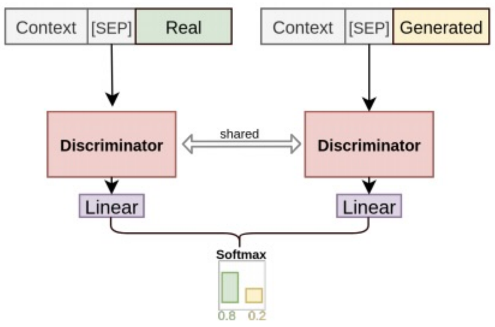


Figure 3.13: Contrastive Discriminator [12]

3.3.4.3 Proximally Optimized Generator

The probability of a text sequence can be expressed as the joint probability of all its words:

$$G_\theta(y_{1:T}|x = \prod_{t=0}^{T} G_\theta(y_t|y_{<t}, x)$$

$y_{1:T}$ is the text sequence, T is the sequence length and y_t is the word at the time t. The reward is maximized with policy gradient:

$$E_{y \sim G_\theta}[\nabla_\theta log G_\theta(x) \hat{R}_y]$$

The straightforward optimization of this objective function causes high variance for the gradient, so PPO policy is used to reduce the effect of it. The likelihood ratio between the old and current policy for $y \sim G_{\theta old}(\cdot|x)$ is used by PPO to perform importance sampling:

$$r(\theta) = \frac{G_\theta(y_{1:T}|x)}{G_{\theta old}(y_{1:T}|x)}$$

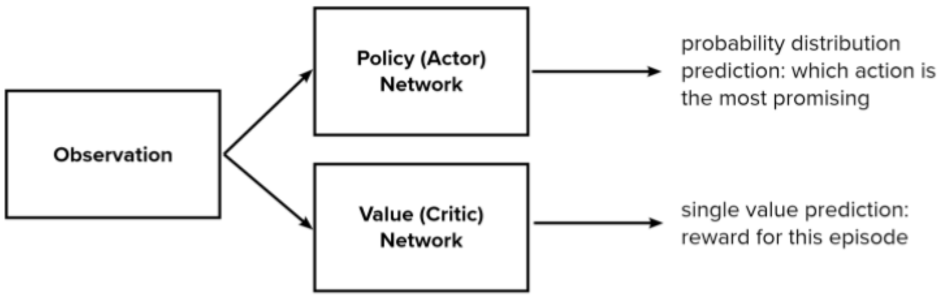

Observation
Policy (Actor) Network
Value (Critic) Network
probability distribution prediction: which action is the most promising
single value prediction: reward for this episode

Figure 3.14: PPO overview [13]

3.3.4.4 Training Strategy

Considering the specificity of the domain, we decided to perform Domain Adaptation on the Generator Architecture. This is due to the fact that, even if the generator is robust itself, it needs a huge quantity of data to deeply learn the specific financial vocabulary.
For this reasons we built a dataset composed by 500.000 records, containing financial reports from the most important companies around the world, from 2017 to 2021 as it can be seen in Table 3.3

Sentence	**Tags**
Carrying value as of the balance sheet date of...	['liabilities', 'invoices', 'balance']
Amount for accounts payable to related parties..	['liabilities', 'accounts', 'portion']
According the finnish russian chamber commerce...	['finland', 'finnish', 'russian']

Table 3.3: Snippet of domain adaptation dataset

Once again we used KeyBERT [33] to extract the main keywords from each report. As we did for the main dataset, we selected the top-3 most influential keywords.

As concerns the tokenization phase, as we already said, we have to deal with two different architectures, GPT-2 as the Generator and RoBERTa as the discriminator. Both of the two share a tokenizer which has the same structure, but with some substantial differences. In fact, the RoBERTa tokenzer is derived from the GPT-2 one, resulting in a **Byte-level Byte-Pair-Encoding**.
Byte-Pair Encoding (BPE) was introduced in Neural Machine Translation of Rare Words with Sub-word Units [41].
The functioning of BPE can be summarized in the following steps:

- Split of the training data into tokens through a pre-tokenizer
- Creation of a list of unique words and their corresponding frequency
- Vocabulary generation through merging rules. Merging rules represent some techniques used to add new terms to the vocabulary until a predefined stopping condition is reached.

The objective of Byte-Level Byte-Pair-Encoding is to create a robust initial vocabulary. Even if the two tokenizers share the same approach they have **different tagging tokens**, which makes it impossible to make the exact matching.
To overcome this problem, we decided to make a mapping of the two vocabularies during the data loading phase, the idea is to use the RoBERTa tokenizer but we force it to map the GPT-2 stranger tokens into its vocabulary. As a result there will no longer be unknown tokens, making the learning phase smooth.

3.4 Classification Architectures

To have a deeper evaluation of the proposed methods for text generation we decided to build a task of text classification on top of it. By doing this we will be able to understand if the synthetic generated data effectively bring in some benefits to the entire process. In this section we will present two different architectures, a modified recurrent neural network GRU and FinBERT: the former represents a basic approach to the text classification problem while the latter a more specialized one for the financial domain we decided to focus on.

3.4.1 Gated Recurrent Unit

RNN is one of the most used deep learning approach for sentiment analysis. Due to its small training time and easy coding, this method serves as a baseline for the evaluation. It is based on sequential data to generate results based on previous computations. [42]. In our case we used a **Gated Recurrent Unit (GRU)** which represents an improved version of standard recurrent neural networks. To solve the vanishing and exploding gradient problem, it introduces *update and reset gate*. This two vectors decide what should be passed to the output. They can be trained to keep information for a long time. [43]

3.4.1.1 Update Gate

The update gate has the fundamental role of deciding the portion of past information to be stored. From a mathematical perspective it computes the update gate z_t for time step t:

$$z_t = \sigma(W^{(z)}x, +U^{(z)}h_{(t-1)})$$

As it can be noticed from FIG.3.15 when the input signal x_t starts its processing it is first multiplied by its corresponding weight $W(z)$ and then summed with the hidden state $h_{(t-1)}$. $h_{(t-1)}$ follows the same preparation as x_t. The final result is obtained by applying a sigmoid function. [43]

3.4.1.2 Reset Gate

The object of this gate is to determine the quantity of information to be forgotten [43].The same procedure saw in 3.4.1.1 is followed, except for the different activation function.
From a mathematical perspective we have:

$$r_t = \sigma(W^{(r)}x_t + U^{(r)}h_{t-1})$$

GRU's complete architecture is reported in FIG.3.15

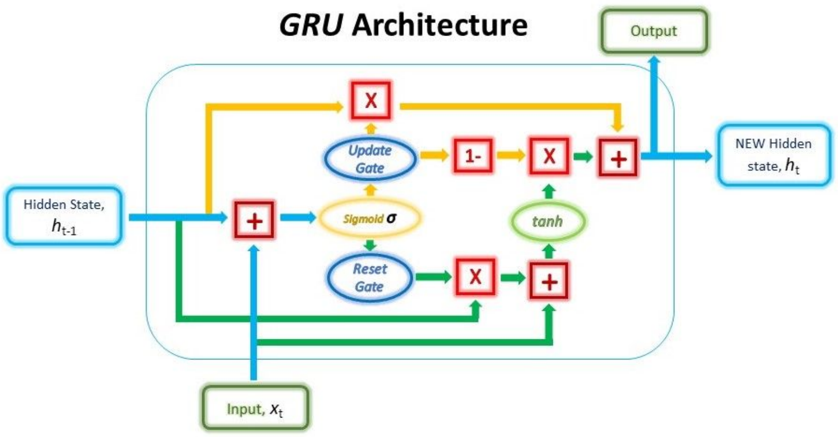


Figure 3.15: GRU overview [14]

3.4.2 FinBERT

FinBERT [15] is an NLP model that has been pre-trained to capture the sentiment of documents belonging especially to the financial domain. The model is based on the BERT language model and simply re-trains it with a large financial corpus, which specializes it for the purpose of financial sentiment classification. The model has been developed to assess the problems related to the extreme specificity of the financial language.
To perform sentiment classification, a dense layer is appended to the final hidden state of the [CLS] token. This approach is considered the best for exploring BERT in any classification task. As a result, the classifier network is trained on the dataset that has been previously labeled following the supervised approach.
FIG.3.16 reports a schema to summarize the main learning steps used to train the model.

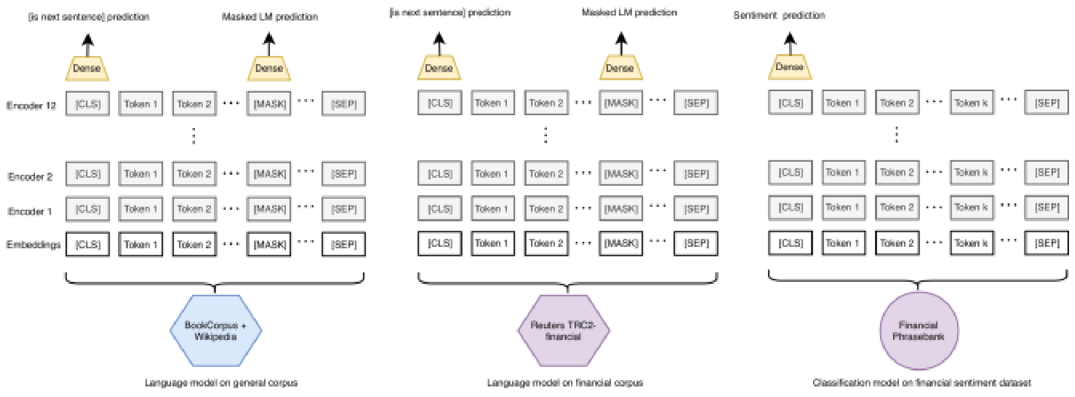


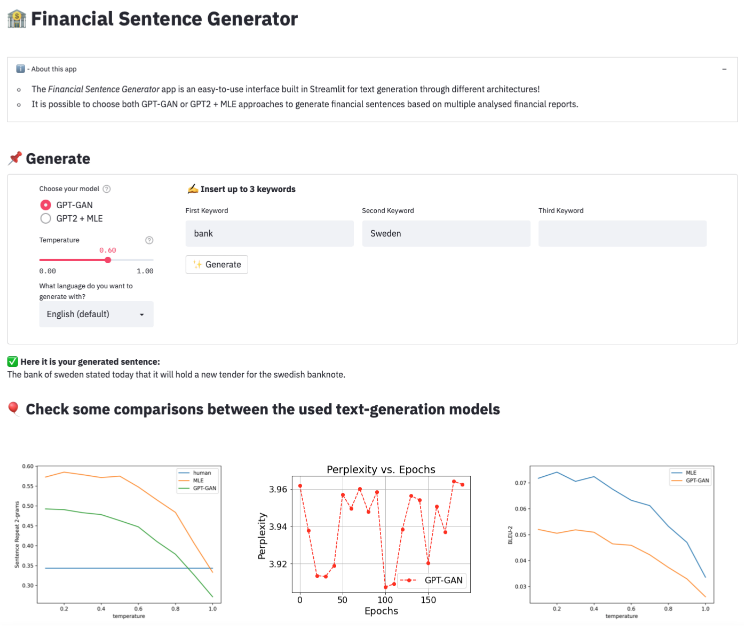
Figure 3.16: FinBERT pre-training and fine-tuning [15]

3.5 Deployment

After working with our algorithm, we decided to actually deploy it. It is a good way to visually see what the proposed model is capable of. For this study case we decided to include only the conditional generation pipeline because of the possibility of the user to interact by inserting guiding keywords. It could be modified to use also the other unconditioned architecture, but we thought it was not as useful as its conditioned counterpart. Going into details, we had to deal with:

- Server: it is a responsive process. It makes computation after a request from a client, it is always running.

- App: it is typically dedicated to user interaction. It makes request to the server and its responsible of all the service and activities.



🏦 **Financial Sentence Generator**

ℹ️ - About this app

- The *Financial Sentence Generator* app is an easy-to-use interface built in Streamlit for text generation through different architectures!
- It is possible to choose both GPT-GAN or GPT2 + MLE approaches to generate financial sentences based on multiple analysed financial reports.

📌 **Generate**

Choose your model

[✓ GPT-GAN] GPT2 + MLE

Temperature 0.60 (0.00 – 1.00)

What language do you want to generate with? English (default)

✍️ **Insert up to 3 keywords**

First Keyword	Second Keyword	Third Keyword
bank	Sweden	

✨ Generate

✅ **Here it is your generated sentence:**
The bank of sweden stated today that it will hold a new tender for the swedish banknote.

🎈 **Check some comparisons between the used text-generation models**

Figure 3.17: Application preview

To implement the **server** we used FastAPI [44], which is currently gaining so much reputation. FastAPI is a collection of tools and more precisely a Python framework that allows developers to employ a REST (Representational State Transfer) interface to execute known functions in order to create applications. In particular, we trained the two conditional model GPT-2 alone and GPT-GAN and we stored their best checkpoints.After that we modified the generation script to be able to handle real time inference by receiving up to three keywords as input. The server receives a json file through a GET request and returns the generated sentence by simply invoking the generation function.
It must be noted that the model initialization is done once the server is started, so there is no delay in the generation function.

As concerns the **client** we decided to use one of the most recent library, which is Streamlit [45]. The objective of this open-source Python library, is to simplify the creation of the front-end by exploring the power of python-friendly scripts to create various and interactive web applications.

3.5.1 Application functionalities

With the aim of fully cover all the aspects related to text generation and class imbalance problem, we decided to implement the following functionalities as services:

- **Text Generation**: as previously introduced, we decided to focus only on the conditional models, due to the key role of the user which has the possibility to guide the sentence creation phase. It is possible to insert up to three different keywords to obtain a financial sentence with the further possibility of choosing between the gpt-2 architecture and the gpt-gan one. To make the application as spendable as possible, we also joined our algorithm with some pre-trained machine translation models provided by HuggingFace [46]. The goal is to make the user able to generate and insert terms coming from different languages.
The principal model works only with the English vocabulary, but the machine translation pipeline allows us to translate the keywords and the final sentence, after and before the the generation phase.
As shown in the preview FIG.3.17, it is possible to choose between English, Italian, Spanish and French.

- **Dataset Re-balancing**: one of the key part of our entire thesis work is to generate synthetic sentences to solve the so famous problem of missing and imbalanced data. We decided to provide a service which gives the possibility of simply uploading a `.csv` file that will be elaborated by the server to provide a final `dataset-rebalanced.csv` containing all the required additional sentences for each class. At the moment this functionality is working only for sentiment analysis applied on financial data, but we will reserve to implement further additions to make it more generic.

- **Custom fine-tuning**: as previously mentioned, even if our models have been tested on the financial domain, considering the great power of GPT-2, it is possible to apply the fine-tuning phase to data coming from different domain. With this functionality we aim to give to the user the possibility of working with both of the implemented models without dealing with the difficulties of setting-up the training phase. In fact, by simply using a GUI, it is possible to create a new model, select the desired working dataset and wait for the training to be completed to have the optimized checkpoints.

Chapter 4

Results

In this section we will report all the results obtained for both text generation and text classification task. First we will go through text generation to understand how the different models generate the text that will be used to obtain a balanced dataset for the second task of text classification. The main experiments that have been performed are:

- Text Generation
  - Unconditional Generation: SeqGAN vs GPT-2 (4.3.2)
  - Conditional Generation: GPT-GAN vs GPT-2 (4.3.3)
- Text Classification
  - Base dataset classification (4.4.2)
  - SeqGAN re-balanced dataset classification (4.4.3)
  - GPT-2 Unconditioned re-balanced dataset classification (4.4.4)
  - GPT-2 Conditioned re-balanced dataset classification (4.4.5)
  - GPT-GAN re-balanced dataset classification (4.4.6)

4.1 Experimental settings

All the experiments have been performed using Google Colab, which is a service provided by Google that gives the possibility of training even large and deep neural networks by offering GPUs. In this case all the tests were performed using the Nvidia P-100. The use of this graphics permitted us to run some of the lighter algorithm in less than a hour, as it happened for the training of the SeqGAN and the GPT-2 Unconditional models. As concerns the conditional pipeline, the run of the GPT-GAN took up to 20 hours, considering the fine-tuning on the modified datasets, while for the Conditional GPT-2 up to 12 hours. As regards the execution times of the classification task, all the experiments performed with GRU were completed in less than a hour while up to 4 hours were needed to evaluate the results of FinBERT.

4.2 Evaluation Metrics

During our thesis work we will perform and test both text generation and text classification task, so it is important to define the metrics on which we will deduct the performance of the proposed algorithms.

4.2.1 Text Generation metrics

Evaluating a text generation task can be crucial and difficult at the same time. Considering the nature of task, it is impossible to use classical metrics in the unconditional scenario. In fact the generated sentences are completely produced from scratch by the model, so it is not suitable trusting on syntactic aware indices, because there will not be any reference to compare with. In this case we will use human evaluation. In the conditional generation, as it happens in GPT-2 and GPT-GAN, we do not only have the input tokens that will guide the conditional generation, but we will also have the human generated sentence to use as ground truth to make comparison with the generated ones. Saying this it is possible to use classical NLP metrics for guidance, such as ROUGE and BLEU score, but we also have the possibility of using semantic aware metrics such as the BERT score.

- **Bilingual Evaluation Understudy (BLEU) score** [47]: the rationale behind BLEU is that the closer a generated text is to its human reference, the better it is. This score is computed by finding the number of identical n-grams in the candidate text and in the reference one. There is the possibility to refer to single token (1-gram) or pair of tokens (bi-gram). The word order does not influence the final computed score. Modified n-gram precision is used to prevent candidate sentences from receiving high score for huge amount of overlapped words. [47].
  Theoretically it is possible to achieve a perfect score, but it is unrealistic because the generated sentence would have to completely match the reference one.

- **Recall-Oriented Understudy for Gisting Evaluation (ROUGE) score**: the main idea of this scoring function is to automatically compare a machine-generated sentence to a human generated reference text. The *recall* tries to understand how much the proposed sentence captures the content of the reference.It can be computed as:

$$\frac{number\ of\ overlapping:\ \ words}{total\ words\ in\ reference\ sentence}$$

Even if this two metrics seem really powerful, they suffer from the inability of understanding semantics. This means that if even long sentences have a lot of token overlapping they could have a completely different meaning. To solve this problem we can use BERT score, which uses the transformer structure to capture the semantic meaning of sentences.

- **BERT score** [16]:it calculates a similarity score for every token in the candidate sentence in comparison to each token in the reference sentence. However, it does not rely on exact matches but explores contextual embeddings to evaluate the similarity between tokens. BERT score has a higher correlation with human judgments leading to a better model selection compared to traditional metrics. Moreover it addresses two prevalent issues in n-gram based scoring functions. First, they are not able to recognize paraphrases and then they fail to capture distant dependencies causing an underestimation of the performances.

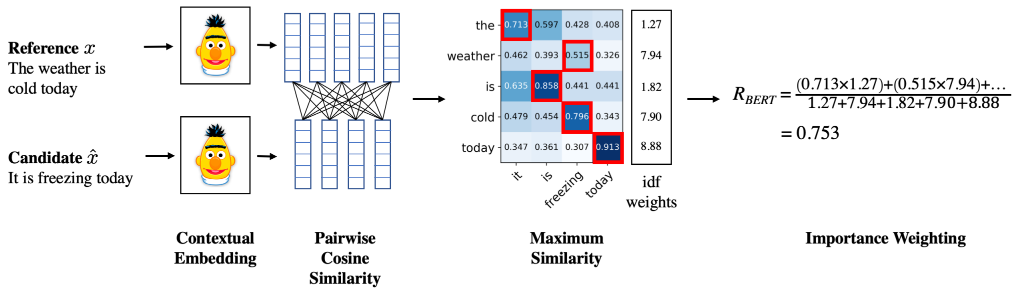


Figure 4.1: BERT score procedure overview [16]

The fundamental steps are:

- **Token representation:** the input tokens are represented exploring the contextual embedding technique. This leads to a generation of different representation of even the same word, depending on the captured context.
- **Similarity measure:** this approach explores a more flexible similarity measure. To do so vector representation is used. In fact, cosine similarity joined by contextual embedding, make the extraction of contextual information easier from not only the analyzed pair of tokens but from the entire sentence as well.

– **Scoring:** The complete score is obtained by matching each token in the reference sentence to a token in the generated one to compute *recall* while it does the opposite to compute the *precision*.

$$R_{BERT} = \frac{1}{|x|} \sum_{x_i \in x} max_{\hat{x}_j \in \hat{x}} x_i^T \hat{x}_j \quad P_{BERT} = \frac{1}{|\hat{x}|} \sum_{\hat{x}_i \in \hat{x}} max_{x_i \in x} x_i^T \hat{x}_j$$

$$F_{BERT} = 2 \frac{P_{BERT} \, R_{BERT}}{P_{BERT} + R_{BERT}}$$

– **Importance weighting:** Recent studies have shown that rare words incorporate more meaning than common ones. With *inverse document frequency (idf)* it is possible to give rare words more importance.

4.2.2 Classification metrics

An evaluation metric quantifies the performance of a predictive model. In general, the process starts with the training of a model on a particular dataset, then generate the predictions on a separate holdout dataset (test set), not used during training, and finally compare the obtained results with the ground truth. The most used metrics to evaluate a classification model are:

- $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$
- $Precision = \frac{TP}{TP+FP}$
- $Recall = \frac{TP}{TP+FN}$
- $F1 = \frac{2 \times Precision \times Recall}{Precision+Recall} = \frac{2 \times TP}{2 \times TP+FP+FN}$

The challenge of correctly evaluating a model becomes even harder when dealing with an imbalanced dataset like in our case. The reason for this is that many of the standard metrics become unreliable or even misleading. What we are trying to achieve with the **F1-score** metric is to find an equal balance between precision and recall, which is extremely useful in most scenarios when we are working with imbalanced datasets. The range of F1 is in [0, 1], where 1 is perfect classification and 0 is total failure.

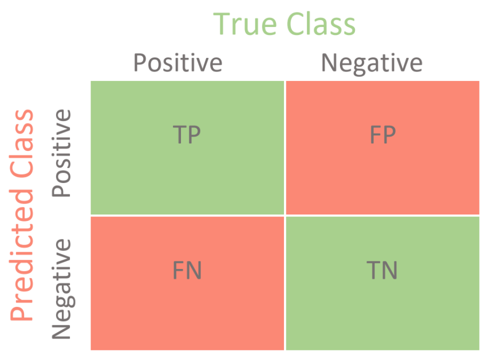


Figure 4.2: Confusion matrix [17]

FIG.4.2 reports an example of confusion matrix. It represents an important tool to first understand how the most used classification metrics are computed but it is also a fundamental way to visualize how the analyzed model is approaching to the classification problem on the desired dataset. As it is possible to notice a simple task with two possible outcomes is presented but the same reasoning applies for multi-class problems. On the main diagonal are reported the True Positive and True Negative results. This two scores must be maximized to obtain a good classification. While on the secondary diagonal are reported the False Positive and False Negative.
With the use of the confusion matrix we are able to visually understand where our algorithm is lacking on and obtain a general idea of the classification performances.

4.3 Text Generation Results

The goal of theses experiments is to generate synthetic text which is as close as possible to the financial domain used for this study case. In this section we will show how the different architectures have behaved. We will present first the base models which will perform unconditional generation, then we will go through the comparison between the two conditional approaches GPT-GAN and GPT-2 by itself.

4.3.1 Hyperparameters tuning

For the following experiments we will report the best combination of hyperparameters found for each model test. Considering the heaviness of each training, we performed a **random search**. The concept is about encircle a search space as a limited number of hyperparameters and randomly selecting point in that range.

The choice is due to the fact that some of the presented models took a really long time to complete the training, and trying to test all the possible combinations of parameters was not suitable.

4.3.2 Unconditional Generation

With Unconditional generation we aim to generate sentences without any guideline, just by fine-tuning the model on the studied dataset. For this reason, it is quite difficult to have a complete and deep evaluation of the results. It is not possible to use any n-gram based evaluation metric, due to the absence of a ground truth reference, the same applies for BERT score. For this reason, we will provide a human evaluation trying to underline different aspects, such as the variety of the words used, the repetition of the sentence and generally the meaning provided by the entire sentence.

4.3.2.1 Seq-GAN

Considering that our aim is to use text generation to re-balance the dataset, as it can be seen in 3.2.1, it is strongly imbalanced towards the neutral class, while there is still a good number of positive sentences, the negative ones are in minority. In this case we will focus only on the generation of positive and negative phrases to make the dataset balanced. To do this, we need to train two different models:

- The Positive Seq-GAN: it is trained only on the positive sentence of the previously mentioned dataset (3.2). The goal is to learn how to generate positive phrases.
- The Negative Seq-GAN: it is trained only on the negative examples.

Our aim is to generate the number of samples needed to reach the perfect balance between the three aforementioned classes.
In Table 4.1 and Table 4.2 are provided all the hyperparameters used and tested to reach the best results for both generator and discriminator.

SHARED HYPERPARAMETERS	**VALUE**
EMB_DIM	32
HIDDEN_DIM	32
SEQ_LENGTH	20
MIN_SEQ_LENGTH	10
BATCH_SIZE	64

Table 4.1: Generator and Discriminator shared parameters

- *Emb_dim*: it indicates the dimension of the embedding
- *Hiddem_dim*: hidden state dimension of the LSTM cell
- *Seq_length*: number of words in the generated sequence
- *Batch_size*: number of samples processed before the model is updated

DISCRIMINATOR	**VALUE**
EMB_DIM	64
FILTER_SIZES	[1,2,3,4,5,6,7,8,9,10,15,20]
NUM_FILTERS	[100, 200,200,200,200,100,100,100,100,100,160,160]
DROPOUT_KEEP_PROB	0.75
L2_REG_LAMBDA	0.2

Table 4.2: Discriminator parameters

Example Sentence	Sentiment
If you need malware removal tool type the URL of your vendor of choice directly into the browser bar and use link on their website wrote Trend Micro Rik Ferguson on Monday	Neutral
Previously Grimaldi held a pct stake in the Finnish company following the takeover bid launched in November	Neutral
Barclays sell benchmark index unit to Bloomberg	Positive
To our member and partner the use of IT will mostly be apparent in the increased efficiency of the result service observes Perttu Puro from Tradeka	Positive
The result solution group finland excluding for into may million to margin excluding million the first watch	Negative
Finnish electronics contract manufacturer Scanfil report net sale of EUR mn in the second quarter of down from EUR mn a year earlier	Negative

Table 4.3: Generated sentences

As it is possible to see in Table 4.3 the overall quality of the sentence is mediocre. We reported some of the most meaningful phrases, but there is an alternation of proper written sentences and bad written ones.
From a sentence length point of view, we can see that there is a satisfactory number of words. Even variety seems to be fully accomplished, as we can see there are no repetition. As concerns the meaning expressed by each sentence, even if some of the phrases are good overall, there a some which are completely out of context. As expected, the model seems to be very sensitive and unstable, this results in a good number of sentences which are not acceptable for the task. Another problem which appeared during the inference phase, is that the negative generated sentences tend to be more confused and less adaptive to the domain. This could be certainly

caused by the significant inferior number of examples on which the model has been fine-tuned on, so it may be necessary to use more data to better accomplish the task. The overall results are not satisfactory, the niche domain of application seems to be too difficult to be understood by the model to generate proper sentences.

4.3.2.2 GPT-2 Unconditional Generation

As mentioned in 3.3.3.7, even if the original model performs the text generation task by completing an input sentence, there is also the possibility of performing unconditional generation by giving only the start of string token $< |endoftext| >$ as input. Table 4.4 summarizes all the best hyperparameters found. Also for this model we decided to train two different models for both positive and negative sentence generation, following the same procedure of 4.3.2.1.

Parameter	Value	Parameter	Value
vocab_size	50257	*resid_pdrop*	0.1
n_positions	1024	*embd_prop*	0.1
n_embd	768	*atti_pdrop*	0.1
n_layer	12	*layer_norm_epsilon*	1e-5
n_head	12	*initializer_range*	0.02
activation_function	GeLu	*bos_token_id*	50256
scale_attn_weights	True	*eos_token_id*	50256

Table 4.4: GPT-2 best hyperparameters

- *vocab_size*: Vocabulary size of the model
- *n_positions*: Maximum sequence length that can be used
- *n_embd*: Dimension of the embedding and the hidden states
- *n_layer*: Number of hidden layers in the encoder
- *n_head*: Number of attention heads for each attention layer in the encoder
- *resid_pdrop*: Dropout probability for all fully connected layers
- *embd_pdrop*: Dropout ratio
- *att_pdrop*: Dropout ratio for the attention
- *layer_norm_epsilon*: The epsilon to be used in the normalization layers

- *initializer_range*: Standard deviation used for initializing all weight matrices
- *activation_function*: Activation function to be used. In this case we used the Gaussian Error Linear Unit (GeLU) [48], which brings some advantages with respect to its predecessor ReLU. In fact it combines the functionalities of Dropout Regularization and ReLU, leading to better results across different tasks.

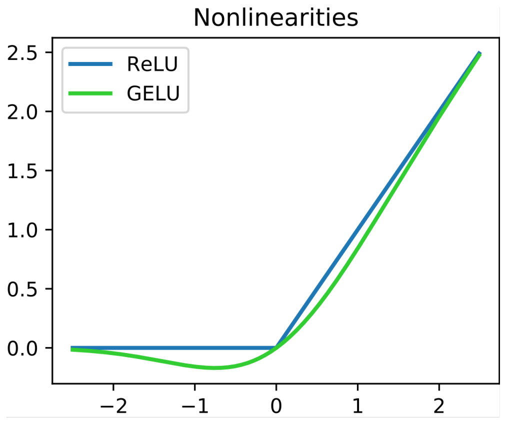


Figure 4.3: ReLU vs GeLU activation functions [18]

As we can see from Table 4.5 the overall quality of the sentences is good, the phrases seems to have a good variety of words, they look well formulated and semantically rich of meaning. Here GPT-2 fine-tuned shows its strong and weak points at the same time. In fact, even if the generated sentences are good, they don't seem to follow the context. As already introduced before, this model finds some important difficulties in specializing in niche domains. As predicted, the negative model suffers from this problem even more, because of the minor quantity of data used for training. Considering this, even if the sentences are well formulated, they can not be considered acceptable.

Example Sentence	Sentiment
The man accused of killing four people in a Swedish ski resort was the biggest perpetrator of political murder in Sweden in years, an investigation has revealed.	Negative
I hope you found this an informative read. I hope you've found it enlightening to you. Why are you so obsessed with casting?	Negative
I just returned from Moscow, so was not sure if I'd be able to go in the evening and get it. My husband arrived to see.	Positive
Right now, my operating system is not available on the newly launched platform. I am using a patching tool that will improve the performance and stability.	Positive

Table 4.5: GPT-2 Unconditional generated sentences

4.3.3 Conditional Generation

With conditional generation we aim to use some guiding input signal, in this case, some keywords to generate specialized sentences. Considering the nature of the task, as opposite to Unconditional generation, we are able to evaluate the generated phrases having some human generated ones. It is possible to use both syntactic and semantic metrics, as well as human evaluation to deeply understand the correctness of the output sentences.

4.3.3.1 GPT-2 Conditional Generation

As already presented, this model is able to generate sentences by completing input tokens. In our case, we aim to slightly modify the training procedure in order to generate phrases by adding up to three keywords as input signal.
To do this we applied a fine-tuning phase by modifying the data loading stage. In fact, after extrapolating keyword following 3.2.3, we changed the tokenizer to work with keywords. As a result, after completing the fine-tuning, it is possible to use the model to make inference based on input words. The best model parameters used are summarized in Table 4.6

Parameter	Value
vocab_size	50257
n_positions	1024
n_embd	768
n_layer	12
n_head	12
activation_function	GeLu
scale_attn_weights	True

Parameter	Value
resid_pdrop	0.1
embd_prop	0.1
atti_pdrop	0.1
layer_norm_epsilon	1e-5
initializer_range	0.02
bos_token_id	50256
eos_token_id	50256

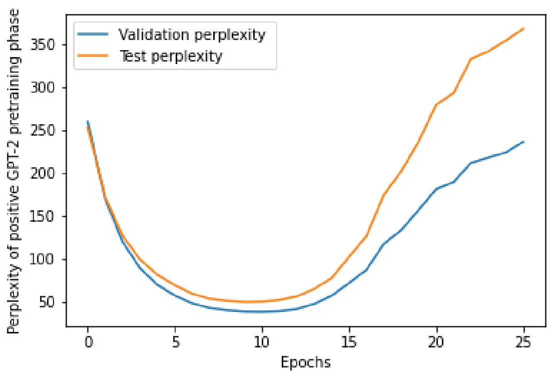
Table 4.6: GPT-2 best hyperparameters

As concerns the training parameters, they are reported in Table 4.7

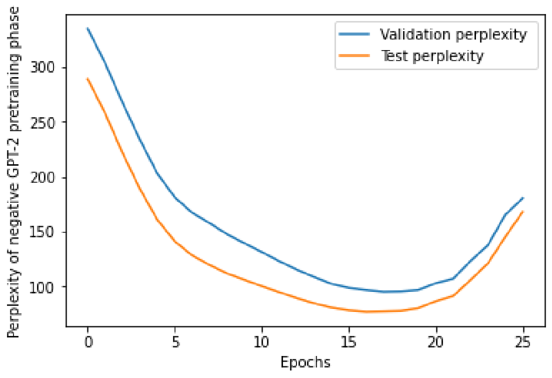
Parameter	**Value**
Epochs	10
Learning rate	5e-4
Epsilon	1e-8
Warmup steps	1e2

Table 4.7: GPT-2 Training parameters

For conditional models we decided to report the trend of the perplexity as the epochs increase. This can be considered an important factor because it will lead us to the choice of the best model.A lower perplexity will result in higher quality of the generated sentences. As it can be noticed from FIG.4.4a and FIG.4.4b, the validation perplexity starts from a high value, then it linearly decrease until it reaches a minimum point. After the minimum peak there is a phase of increase, this is due to the fact that the model is overfitting, the specialization on the training set has gone too far and the capacity of generalization starts to get lost. In both model, the perplexity reaches really high values at the end of the training. As concerns the best results, while the positive model achieves a perplexity of 40, the negative one is about 80. So there will be some differences in the quality due to this factor.



(a) Perplexity trend for positive model



(b) Perplexity trend for negative model

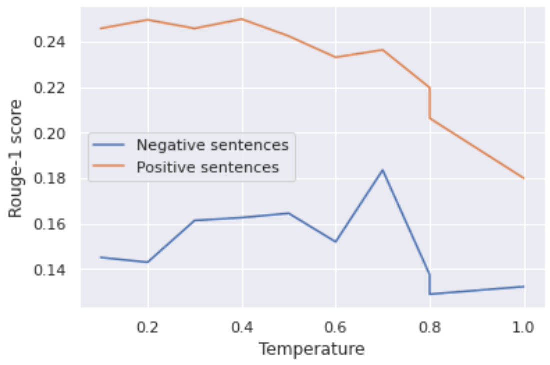
Figure 4.4: Perplexity trend for positive and negative models

Table 4.8 reports some example of generated sentences. In this case, the human generated reference are drawn as well. The first two examples are from *positive* generation while the last two from the *negative* one

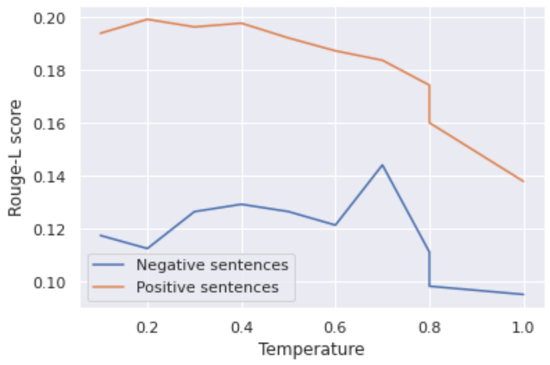
Generated	Reference
"Both sales and net sales of the group grew by 3m and 4m respectively in 2006"	"The ai million is to increase sales by at least one fifth in 2006"
"Operating profit increased by eur 200m compared to a loss of 0 for the corresponding period in 2005"	"Operating profit increased to eur 140 mn fro million eur 49 mn in the corresponding period in 2005"
"The patent on the Flsmidth is not in the Danish company"	"Danish company Flsmidth has acknowledged that it has violated a patent held by finnish Metso"
"The ban on the sale of the Mario WVX stock and the sale of the WVX stock to the company in the country have been temporarily halted and the company will resume trading in the country"	"At the moment Valio is not worried but if the ban continues for long it may become quite a problem"

Table 4.8: GPT-2 Conditional generated sentences

As it can be seen from Table 4.8 the overall quality of the generated sentences is good, they seem to be close to the human generated ones. The length and variety is satisfying and the context is respected too. From a mathematical point of view, we decided to plot some statistics with the aim of understanding and underlining the small details that can not be seen through simple human analysis. We will plot both positive and negative models, trying to extrapolate the differences.



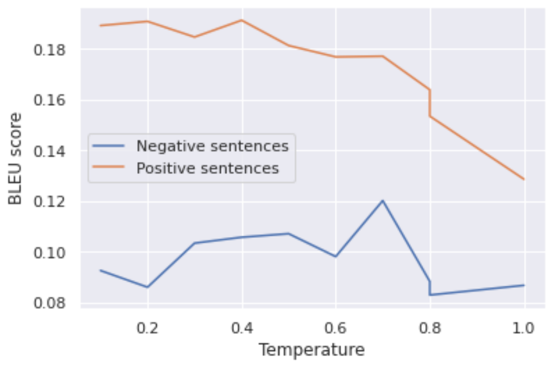
(a) Rouge-1



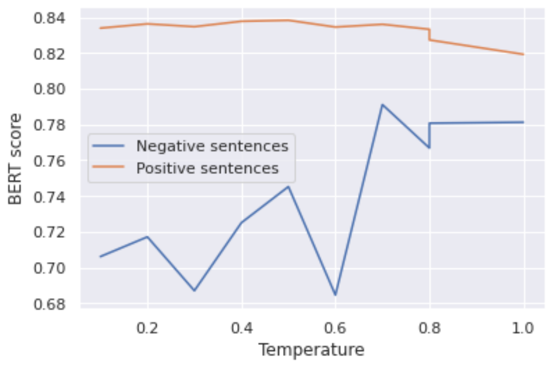
(b) Rouge-L

Figure 4.5: Rouge-1 and Rouge-L of positive and negative models

In FIG.4.5a and FIG.4.5b we plotted both *Rouge-1* and *Rouge-L* scores as the *temperature* value increases. Temperature is a parameter used in natural language processing models to increase or decrease the "confidence" a model has in its most likely response. In our setting, it is extremely indicative, and the study of its variation is crucial to understand which model will perform the best. As expected, the rouge score in both its computations, starts to decrease as the temperature increases, but while in the negative model it reaches its highest peak in 0.7, in the positive model the degrowth is linear.



(a) BLEU



(b) BERT score

Figure 4.6: BLEU and BERT score of positive and negative models

The same trend can be observed in FIG.4.6a for the BLEU score. Its behavior is similar to Rouge-L. Also in this case the negative model reaches slightly worse results. As already anticipated, the significant minor quantity of negative data available has a key role.
From the semantic point of view, the results are impressive. As reported in FIG.4.6b, the BERT score computed between the human reference sentence and the gpt-2 generated one is quite high, reaching a peak of 0.84 for positive phrases and and 0.80 for negative ones. This suggests us, that with the conditional generation the context is more easily integrated into sentences leading to a more accurate generation.

4.3.3.2 GPT-GAN

As already mentioned in 3.3.4 we tried to explore the power of a large language model as GPT-2 in a GAN environment against a powerful adversary as RoBERTa. Even in this case the generation is conditioned by the input keywords, we believe that their help could be even more influential. As concerns the training procedure, we first did a warm-up training to obtain the weights of an already specialized generator. We used the same model presented in 4.3.3.1 while the training parameters are reported in Table 4.9

Parameter	**Value**
Epochs	30
Learning rate	2e-5
Optimizer name:	AdamW
Weight decay	0.01

Table 4.9: GPT-2 Generator Training parameters

As concerns the discriminator, we pre-loaded the Roberta-base model.

In particular all the detailed parameters of the learning phase are reported in Table 4.10

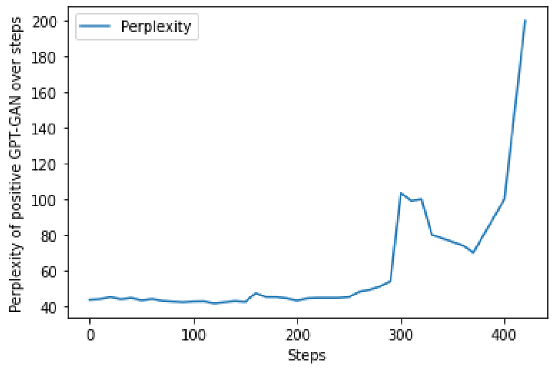
Parameter	Value
PPO buffer size	128
PPO espilon:	0.2
Sample batch size	32
Discriminator pretrain steps	200
Recompute log probs	True

Table 4.10: Training parameters

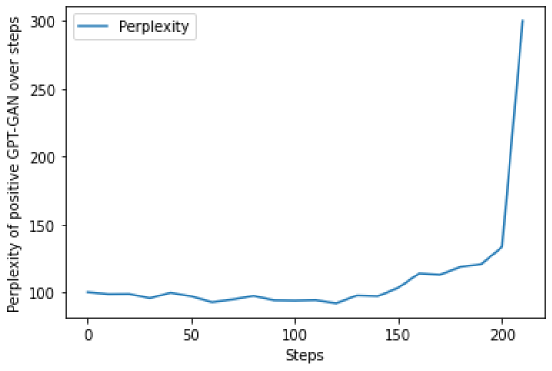
- *PPO buffer size*: This refers to the amount of experiences required to be collected before updating the policy model.
- *PPO epsilon*: This refers to the acceptable level of difference between the previous and new policies during gradient descent updating. If this value is small,more consistent updates will be made, but it will also cause the training process to be slower.
- *Recompute log probs*: It may be tuned to to address precision issues in autoregressive generation.

As we did in 4.3.3.1 we reported the perplexity trend to try to understand better the behavior of the model during the training phase. In this case we reported only the validation perplexity in FIG 4.7a and FIG.4.7b. Considering that the generator is initialized with the weights of the GPT-2 model, this analysis will focus only on the behavior of the model during steps, as there are no fixed epochs. As we can see there is a initilization phase in which the perplexity tends to be stable, only the loss is optimized, but as it happened for the MLE model, at around step 300 the model starts overfitting reaching very high peaks.

As it can be seen in FIG.4.8a and FIG.4.8b for both positive and negative models, the generator and discriminator losses cross in a point and then they will diverge. This is because, as already mentioned, even if both generator and discriminator have the goal of optimization of their loss function they are at the same time adversary. This means that when one starts to prevail the other will fail, it is not possible for the two to "win" at the same time

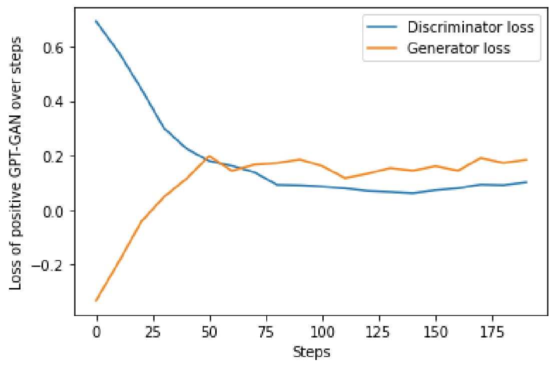


(a) Perplexity trend for positive model

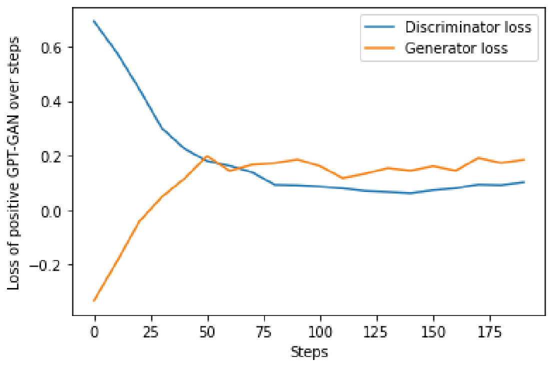


(b) Perplexity trend for negative model

Figure 4.7: Perpelxity of positive and negative models



(a) Generator and discriminator loss trend for positive model



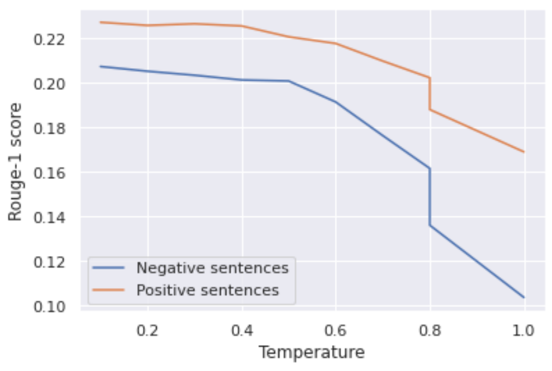
(b) Generator and discriminator loss trend for negative model

Figure 4.8: Loss trends of positive and negative models

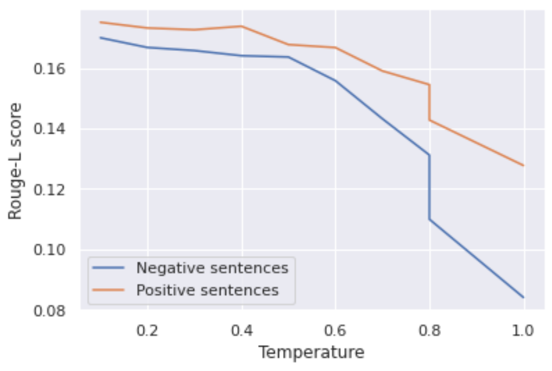
Table 4.11 reports some of the generated sentences. The first two are from the positive model while the last two from the negative one. As it can be noticed both length and variety are really good. The context seems to be focused and the generated sentence are really close to the human reference. As we did for GPT-2 alone, we reported some statistics to deeply understand how the generation behaved. The collected metrics will be very useful also for the comparison phase. In fact, thanks to these we will be able to determine which models fits the task better leading to more complete synthetic data. Once again we plotted both syntactical and semantic metrics as the temperature values increase, following the same procedure of 4.3.3.1

Generated	Reference
"In finland the average sales were eur 9 million up from eur 48 million in 2006"	"However sales volumes in the food industry are expected to remain at relatively good levels in finland and in scandinavia atria said"
"Nordea estimates its long term growth outlook for baltic and american to be positive and that it expects its long term growth to be medium term"	"Nordea sees a return to positive growth for the baltic countries in 2011"
"The company said that fewer people had been hired in the previous two years than in the previous year"	"As a result some 20 persons will no longer be needed"
"The company condition is that the corporator in the country will reduce the profit of its office in the coming year"	"Finnish technology company Raute corporation Omx Helsinki Rutav issued on tuesday 23 September a profit warning for the financial year 2008"

Table 4.11: GPT-GAN Conditional generated sentences



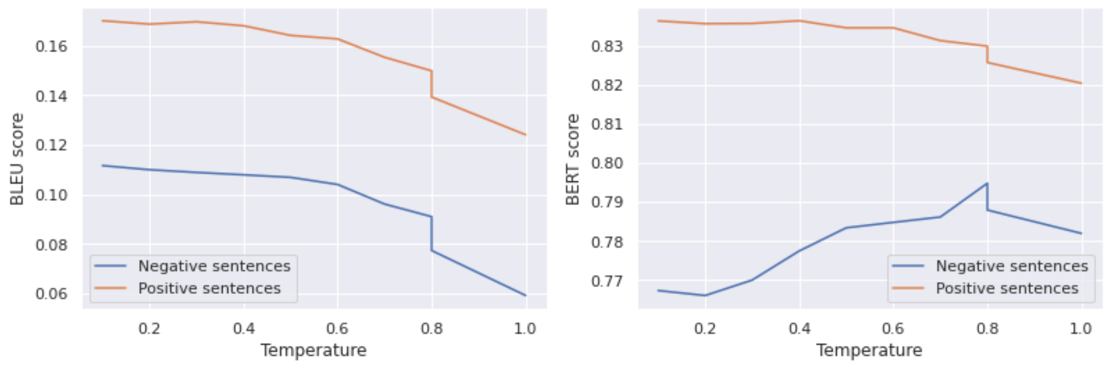
(a) Rouge-1



(b) Rouge-L

Figure 4.9: Rouge-1 and Rouge-L of positive and negative models

As it can be noticed from FIG.4.9a and FIG.4.9b, in this case both positive and negative models follow a linear decrease of the Rouge-1 and Rouge-L values. This is completely understandable as the possibility of the model of varying increases as well. Less and less words are overlapping leading to a drastic decrease of about 0.1. The drop is even harder for the negative model.



(a) BLEU **(b)** BERT score

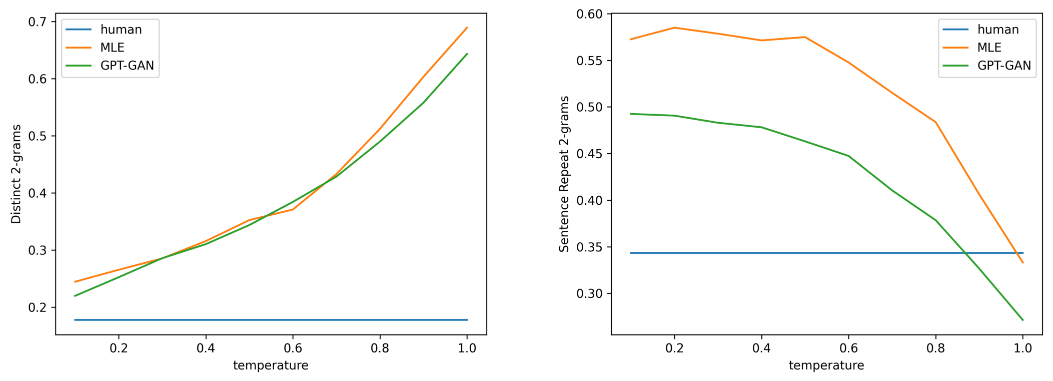
Figure 4.10: BLEU and Bert score of positive and negative models

As regards the BLEU score in FIG.4.10a, for the positive model the peak is reached for the initial temperature value of 0.1 but acceptable results are obtained until 0.7. The same does not apply for the negative model, dealing with a significant minor score in the range 0.10 - 0.12.
As concerns the semantic, the BERT score in FIG.4.10b shows incredible performances, keeping a 0.81 score for positive sentences while a slightly less value of 0.79 for negative ones. The particular fact is that while the negative phrases reach the highest peak for a temperature value of 0.8, for the same point the positive reach the minimum. Once again the conditional generation has showed to obtain really good results.

4.3.3.3 GPT-GAN vs GPT-2

In this section we will compare the two main conditional approaches, trying to underline objective factor such as variety and repetition.

As we can see we reported some statistics of the three approaches, we also considered the human reference as guiding benchmark. For both GPT-2 and GPT based GAN we reported the results obtained by the positive model. As it is noticeable from FIG.4.11a the number of distinct 2-grams is very high for both the two approaches. They seem to use a huge variety of the words in the vocabulary trying to avoid redundancy. As concerns the number of sentence repetition in FIG.4.11b, GPT-GAN shows better performance, more and more different phrases are generated and as expected, as the temperature value increase, the number of repetition linearly decreases, becoming really close to the human benchmark.



(a) Number of distinct 2-grams **(b)** Number of sentence repetition

Figure 4.11: GPT-GAN vs GPT-2 quality measures

4.4 Text Classification Results

The goal of the following experiments is to evaluate how much the sentences generated in 4.3 can influence the performances of the sentiment analysis. We will first present the task performed on the base dataset "FinancialPhraseBank" 3.2.1 and then we will compare the results obtained using balanced data coming from the different text generation models we presented in the previous section.

4.4.1 Hyperparameters tuning

For the following experiments we will report the best combination of hyperaparameters found for each model test. As concerns the GRU architecture, considering the restricted depth of the model, we performed a Grid Search that refers to a process that performs the testing of a set of manually defined hyperparameters for the desired algorithm.All the possible combinations of parameters will be evaluated.On the contrary, due to its architecture, FinBERT showed to take a really long time to complete its training phase, this forced us to perform Random Search. The main difference is that this approach only selects and tests a random combination of hyperparameters. In our case the heaviness of the training played a key role in the choice of the two different approaches.

4.4.2 Original dataset

As already mentioned, the base dataset is strongly imbalanced towards the "neutral" class. This will for sure influence the outcome of the classification task.

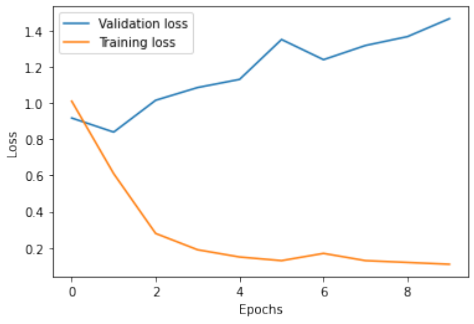
4.4.2.1 GRU

We first performed model training using a simple but effective GRU. It is known for reaching good performances, but also for being general purpose, performing well on different contexts. The most important parameters are drawn in Table 4.12:

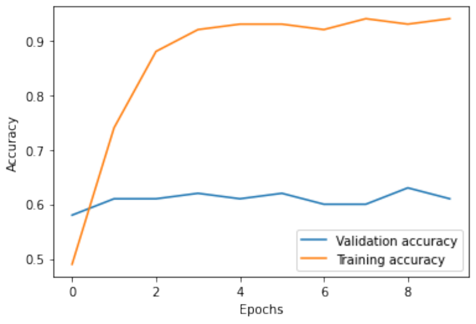
Parameter	Value
Epochs	10
Input dimension	7277
Output dimension	128
Optimizer	Adam
Loss	Sparse Categorical Cross Entropy

Table 4.12: GRU training parameters

In FIG.4.12a and FIG.4.12b we can see the behavior of the model during training phase.



(a) Loss' trend



(b) Accuracy's trend

Figure 4.12: Loss and Accuracy trend of GRU on the original dataset

Table 4.13 presents the classification report, underlying F1-score,Accuracy, Precision and Recall. Even if we presented the full report we will trust the F1-score, trying to find a good balance between precision and recall. As already mentioned in 4.4.2, F1-score is the best choice in an imbalanced environment.

	Precision	**Recall**	**F1-score**
0 (Negative)	0.49	0.61	0.55
1 (Positive)	0.71	0.71	0.66
2 (Neutral)	0.67	0.63	0.65

Table 4.13: GRU Base Dataset Classification Report

The first thing that stands out is that the F1-score for the negative class is significantly less than the positive and neutral class. It is clear that the model finds some difficulties in analysing the examples from the minority class. As concerns the positive label, even if its examples are less than the neutral ones, the classification seems to be already good.

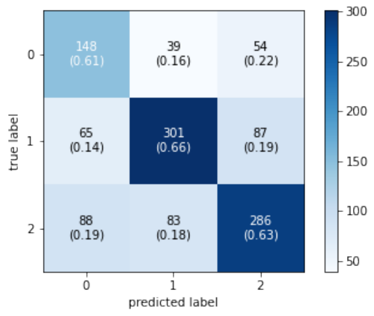


Figure 4.13: GRU Base Dataset Confusion Matrix

The classification is clearer in FIG.4.13. The number of negative examples classified correctly is strictly less than the other two classes, consequently there is a great part of misclassified examples belonging to the positive and above all negative class. The average F1-score is **0.64**.

4.4.2.2 FinBERT

Considering the niche domain of application, we decided to try a specialized classification model with the aim of improving the performances of the general RNN. The main parameters tuned are in Table 4.14.

Parameter	**Value**
Epochs	5
Optimizer	Adam
Learning rate	1e-5
Maximum sequence length	74

Table 4.14: FinBERT main parameters

The classification report is presented in Table 4.15.

	Precision	**Recall**	**F1-score**
0 (Negative)	0.47	0.39	0.43
1 (Positive)	0.83	0.87	0.85
2 (Neutral)	0.83	0.85	0.84

Table 4.15: FinBERT Base Dataset Classification Report

As it can be noticed, even if the classifier performs better than its counterpart, the negative class suffers even more due to the minor quantity of data available. The F1-score for the positive and neutral class increased by 0.20 while for the negative label there is a decrease of 0.10. The average F1-score is **0.70** which shows a increase of performance with respect to the GRU classifier.

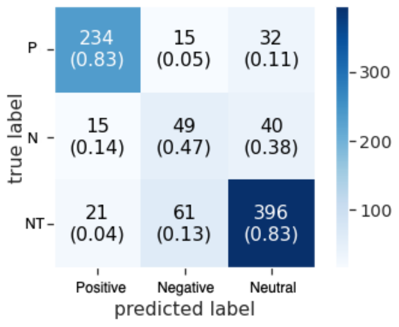


Figure 4.14: FinBERT Base Dataset Confusion Matrix

The confusion matrix in FIG.4.14 shows this effect even more, it is clear that not even the best classifier possible can achieve acceptable results, there is the need of introducing synthetic data to fully explore the potential of the model.

4.4.3 Seq-GAN

The dataset has been perfectly re-balanced by using the SeqGAN. In this section we will analyze how the presented models handle synthetic data.

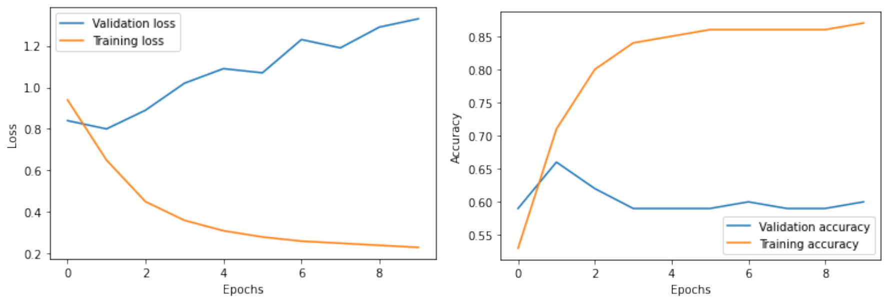
4.4.3.1 GRU

The training parameters are reported in Table 4.16. The same configuration from 4.4.2.1 is maintained except for the input dimension which was changed to 8524 due to the larger vocabulary.

Parameter	**Value**
Epochs	10
Input dimension	8524
Output dimension	128
Optimizer	Adam
Loss	Sparse Categorical Cross Entropy

Table 4.16: GRU training parameters

The training behavior is reported in FIG.4.15a and FIG.4.15b. The trends showed are similar to the one already presented for the base dataset, there are no significant differences



(a) Loss' trend **(b)** Accuracy's trend

Figure 4.15: Loss and Accuracy trend of GRU on the SeqGAN dataset

The classification report is shown in Table 4.17. As it is noticeable, the generation and re-balancing through GAN has brought some benefits, but as suspected the generation is not sufficient. In fact, even if a raise of the F1-score of the negative class is observed, the positive one has decreased. This could be explained by the fact that positive generated sentence are really far from the context resulting in a performance degradation.

	Precision	Recall	F1-score
0 (Negative)	0.61	0.70	0.65
1 (Positive)	0.65	0.47	0.55
2 (Neutral)	0.67	0.75	0.71

Table 4.17: GRU SeqGAN Dataset Classification Report

Our suspects are confirmed by the confusion matrix in FIG. 4.16. It is clearly showed that the model has difficulties in classifying positive examples, they are mainly misclassified with neutral ones. The consequences of what we found in 4.3.2.1 are confirmed, the poor context included in the generated phrases led to the same performances obtained on the base dataset, the average F1-score is **0.64**.

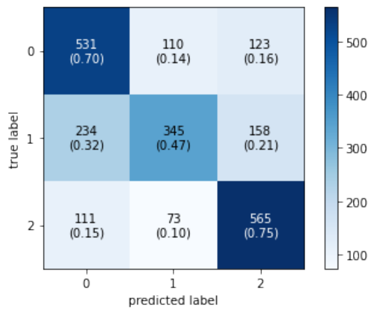


Figure 4.16: GRU SeqGAN Dataset Confusion Matrix

4.4.3.2 FinBERT

The main parameters studied for training are reported in Table 4.18. We kept the same parameters studied for the base dataset considering that in both cases the longest sequence was 74.

Parameter	Value
Epochs	5
Optimizer	Adam
Learning rate	1e-5
Maximum sequence length	74

Table 4.18: FinBERT main parameters

As the classification report in Table 4.19 shows, with this powerful and specialized classifier, the negative class suffers less for the worse quality of synthetic data introduced, and even more surprisingly it is that it performs better than the positive one. It could be due to the fact that this type model is better in finding patterns for more radical sentences.

	Precision	**Recall**	**F1-score**
0 (Negative)	0.65	0.87	0.75
1 (Positive)	0.81	0.64	0.71
2 (Neutral)	0.85	0.76	0.80

Table 4.19: FinBERT SeqGAN Dataset Classification Report

What we anticipated is better shown in the confusion matrix in FIG. 4.17. The classification of the correct negative classes is the best performing, but it should be also considered that the misclassified items are in substantial number. On the other hand, the positive points are well classified and the number of false positive and negative is minimal. As it is possible to notice, the presented results are a lot better than the base counterpart, achieving an average F1-score of **0.75**, which brings a lot of benefits into the task.

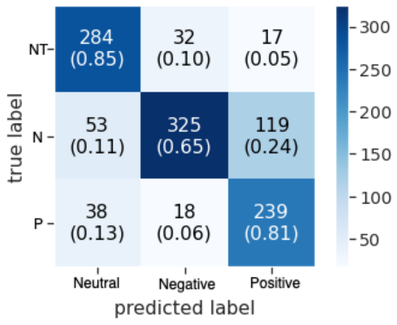


Figure 4.17: FinBERT SeqGAN Dataset Confusion Matrix

4.4.4 GPT-2 Unconditional Generation

Thanks to text generation through GPT-2 fine-tuned for unconditional generation, the dataset has reached a perfect balance among its classes. In the following sections we will analyze the behavior of the model when dealing with this new dataset.

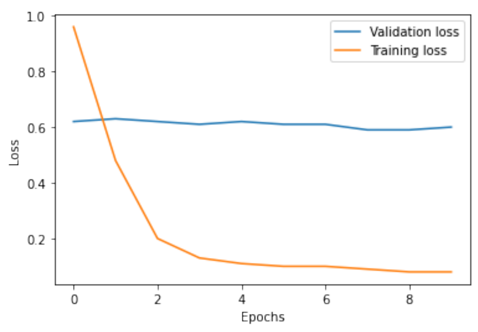
4.4.4.1 GRU

The training parameters are reported in Table 4.20. As usual we used the same parameters selected for the base tests and we simply modified the input dimension due to the different vocabulary length.

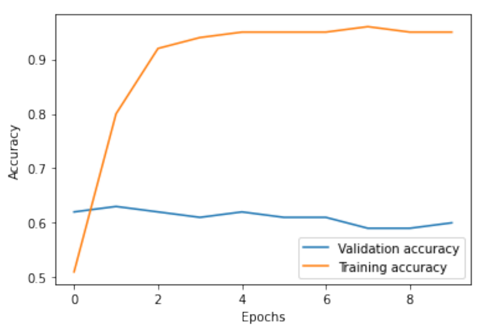
Parameter	Value
Epochs	10
Input dimension	8524
Output dimension	128
Optimizer	Adam
Loss	Sparse Categorical Cross Entropy

Table 4.20: GRU training parameters

As it is possible to see in FIG.4.18a and FIG.4.18b the trend followed by the loss and the accuracy is similar to the base model, there are no major differences. The validation loss tends to be stable at 0.6, while as concerns the accuracy, after reaching a peak on the first epoch, a slight decrease can be observed.



(a) Loss' trend



(b) Accuracy's trend

Figure 4.18: Loss and Accuracy trend of GRU on the GPT-2 Unconditional dataset

As the classification report in Table 4.21 underlines, there are no significant differences with the results obtained with the base dataset. There is an increase of performances for the negative label at the cost of a decrease of the positive one. The neutral class seems to be classified in more or less the same manner.

	Precision	**Recall**	**F1-score**
0 (Negative)	0.63	0.72	0.68
1 (Positive)	0.67	0.50	0.57
2 (Neutral)	0.66	0.74	0.70

Table 4.21: GRU GPT-2 Unconditional Dataset Classification Report

As reported in the confusion matrix, we can see that the precision of the positive class is quite good but the recall is not sufficient at all. The F1-score is fully influenced by this, resulting in an average score of **0.64**. There is an imperceptible improvement compared to the base and seqgan datasets, but the introduction of this kind of synthetic data does not seem to bring any advantage.

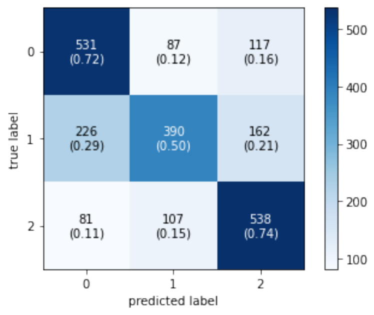


Figure 4.19: GRU GPT-2 Unconditional Dataset Confusion Matrix

4.4.4.2 FinBERT

The main parameters studied for training are presented in Table 4.22. Even in this case we kept the same parameters studied for the base dataset considering that in both cases the longest sequence was 74.

Parameter	Value
Epochs	5
Optimizer	Adam
Learning rate	1e-5
Maximum sequence length	74

Table 4.22: FinBERT main parameters

As the classification report in Table 4.23 shows, even if there is an improvement, the average F1-score is **0.74** compared to the baseline 0.70, the result of the classification task does not seem to be sufficient. Moreover, it can be observed an inverse trend with respect to the GRU classifier, the performance obtained on the GPT-2 Unconditional generated dataset are lower than the SeqGAN counterpart.

	Precision	Recall	F1-score
0 (Negative)	0.65	0.79	0.71
1 (Positive)	0.77	0.67	0.72
2 (Neutral)	0.82	0.76	0.79

Table 4.23: FinBERT GPT-2 Unconditional Dataset Classification Report

As shown in the confusion matrix in FIG. 4.20. the negative class suffers from lower precision resulting in many false positives and negatives.The opposite happens for the positive label. As reported for the SeqGAN, the introduction of sythetic data did not bring any improvements to the task

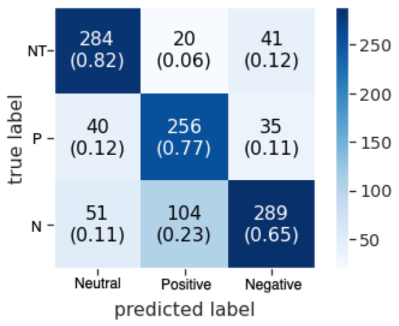


Figure 4.20: FinBERT GPT-2 Unconditional Dataset Confusion Matrix

4.4.5 GPT-2 Conditional Generation

The dataset used for the test has been balanced completely by using GPT-2 fine-tuned to perform conditional generation. Considering the nature of the text generation evaluation, we expect to increase and improve the baseline provided by the base dataset with both models. In this section we will see how the two architectures behave and we will quantify the benefits if any.

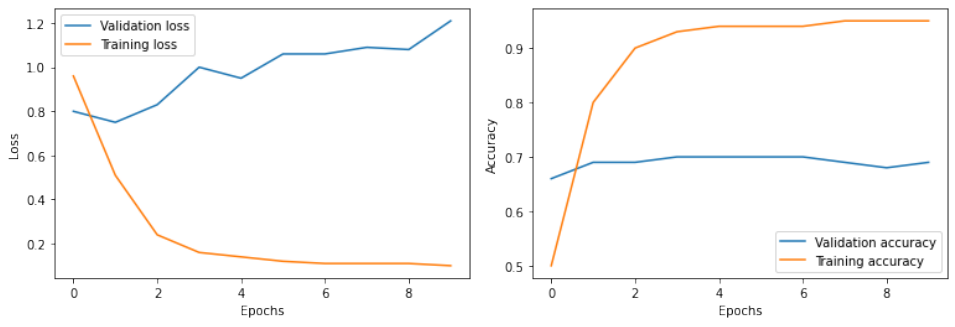
4.4.5.1 GRU

The training parameters are reported in Table 4.24. Once again, the only parameters we changed compared to the previous test is the input dimension to 7324. No other changes are registered.

Parameter	Value
Epochs	10
Input dimension	7324
Output dimension	128
Optimizer	Adam
Loss	Sparse Categorical Cross Entropy

Table 4.24: GRU training parameters

As it is possible to see from FIG. 4.21b, the accuracy of the model is really good, reaching a peak of 0.7. Also the loss in FIG. 4.21a follows a regular path, once the model starts overfitting, the validation loss increases, starting to assume large values.



(a) Loss' trend **(b)** Accuracy's trend

Figure 4.21: Loss and Accuracy trend of GRU on the GPT-2 Conditional dataset

As reported by the classification report in Table.4.25, the introduction of more accurate synthetic data brings more equity among the different classes. They seem to be treated in a very similar way by the model. The average F1-score is **0.70**, there is an improvement of 0.06 compared to the baseline.

	Precision	**Recall**	**F1-score**
0 (Negative)	0.72	0.68	0.70
1 (Positive)	0.78	0.63	0.70
2 (Neutral)	0.61	0.76	0.68

Table 4.25: GRU GPT-2 Conditional Dataset Classification Report

Our hypothesis are strengthen by the confusion matrix in 4.22. As it is possible to notice, all the classes behave in a similar manner. There is still a small gap between neutral and the other two mixes classes, but it seems to be reduced. The results are better, and the introduction of synthetic examples resulted in a increase of performance.

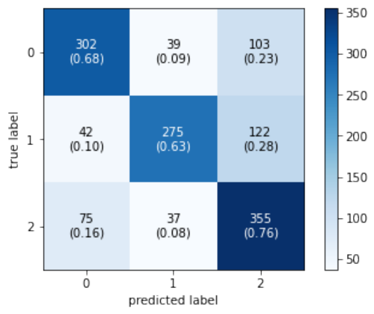


Figure 4.22: GRU GPT-2 Conditional Dataset Confusion Matrix

4.4.5.2 FinBERT

All the parameters tuned during training phase are reported in Table 4.26. In this case we had to deal with overall longer sequences, this made us change the Maximum sequence length to 164, which is the highest value registered

Parameter	**Value**
Epochs	5
Optimizer	Adam
Learning rate	1e-5
Maximum sequence length	164

Table 4.26: FinBERT main parameters

As can be seen from the classification report in Table 4.27 the model shows incredible performances reaching an average F1-score of **0.84**. The interesting thing to notice is that it seems to be the neutral class to have a lot of misclassifications, even if the precision is good there is a relatively low recall. The negative and positive classes, on the other hand, reaches the same results. This means that the quality and the context incorporated in the generated sentence is really good.

	Precision	**Recall**	**F1-score**
0 (Negative)	0.84	0.91	0.87
1 (Positive)	0.84	0.89	0.87
2 (Neutral)	0.88	0.75	0.81

Table 4.27: FinBERT GPT-2 Conditional Dataset Classification Report

The confirmation is showed in the confusion matrix in FIG. 4.23. A significant but not excessive number of examples from the neutral class are classified as positive or negative. This slightly influence the outcome of the model, but it must be underlined that the performances reached are impressive.

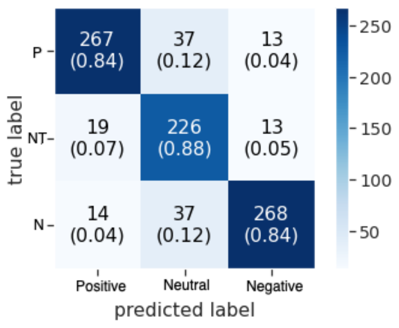


Figure 4.23: FinBERT GPT-2 Conditional Dataset Confusion Matrix

4.4.6 GPT-GAN

In this section we will analyze the classification results obtained on the GPT-GAN balanced dataset.As already seen, the peculiarity of the model is the jointly collaboration between two large language models in GAN environment. The goal is to try to understand if this approach can bring some more advantages with respect to GPT-2 alone.

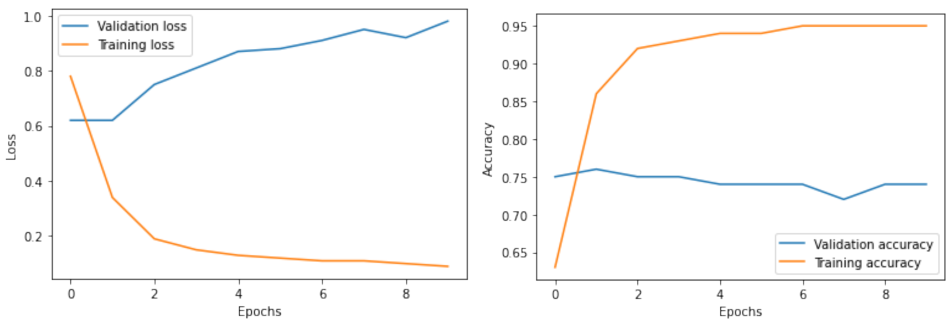
4.4.6.1 GRU

The training parameters are reported in Table 4.28. In this case the vocabulary length was extended to 11360 leading to a change in the input dimension.

Parameter	**Value**
Epochs	10
Input dimension	11360
Output dimension	128
Optimizer	Adam
Loss	Sparse Categorical Cross Entropy

Table 4.28: GRU training parameters

As it is possible to see in FIG. 4.24a and FIG.4.24b both accuracy and loss reach the best results we have seen so far. In particular the validation accuracy peaks at 0.76 while the validation loss minimum is registered at 0.6.



(a) Loss' trend **(b)** Accuracy's trend

Figure 4.24: Loss and Accuracy trend of GRU on the GPT-GAN dataset

The average F1-score is **0.76** which is by far the highest and better result achieved by GRU classifier in this application domain. Previously the best result was reached by the GPT-2 Conditional model with a score of 0.70. An interesting increase of 0.06 can be noticed

	Precision	Recall	F1-score
0 (Negative)	0.79	0.75	0.77
1 (Positive)	0.82	0.78	0.80
2 (Neutral)	0.70	0.76	0.73

Table 4.29: GRU GPT-GAN Conditional Dataset Classification Report

In this case the positive label is the one which is classified better, but even the negative one seems to be well threatened. Generally speaking a balance among the classes can be observed from FIG. 4.25. This is exactly what we wanted to achieve and what we expected from the text generation results.

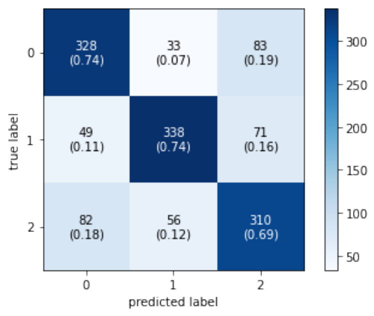


Figure 4.25: GRU GPT-GAN Unconditional Dataset Confusion Matrix

4.4.6.2 FinBERT

The main parameters are reported in Table 4.30. As it happened in 4.4.5.2 even in this case we are dealing to the longest sequences so far.

Parameter	**Value**
Epochs	5
Optimizer	Adam
Learning rate	1e-5
Maximum sequence length	220

Table 4.30: FinBERT main parameters

Table 4.31 show the classification report and surprisingly we have a different scenario from the one observed with GRU. In fact, the average F1-score is **0.80** which is inferior to 0.84 obtained on the GPT-2 conditioned dataset. While the positive and negative labels seem to be well classified, in this case the neutral class suffers from low recall. It is likely that the model is not able to find patterns to completely distinguish this sentiment from the others, resulting in a performance degradation.

	Precision	**Recall**	**F1-score**
0 (Negative)	0.73	0.86	0.79
1 (Positive)	0.87	0.85	0.86
2 (Neutral)	0.81	0.68	0.74

Table 4.31: FinBERT GPT-GAN Dataset Classification Report

What already anticipated can be seen in FIG. 4.26, a lot of negative example have been classified as neutral ones, so it is probable that the patterns found in classification are very similar resulting in misclassification mistakes.

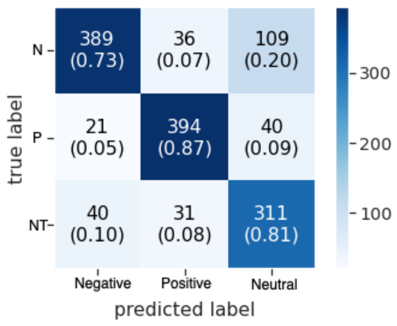
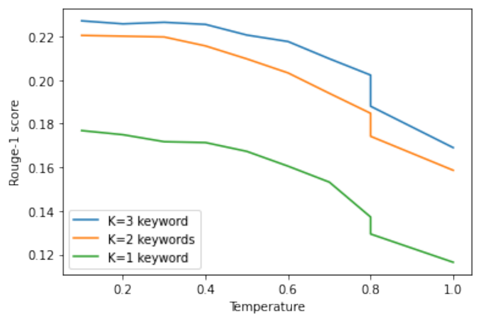


Figure 4.26: FinBERT GPT-GAN Dataset Confusion Matrix

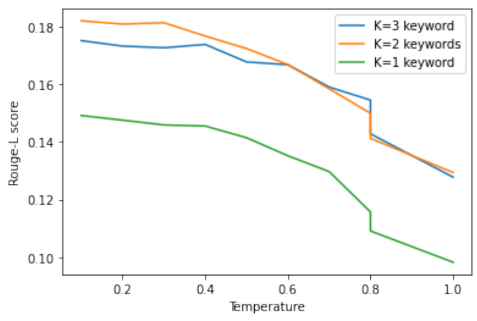
4.5 Parameter Analysis

4.5.1 Number of keywords

One of the key aspect of conditional text generation is to give a strong guiding signal that can keep the model on the path without risking of going out of context. The number of keywords plays a huge role in this phase. It is important to find a great balance between performances and resource needed. More keywords will lead to more specific generated sentence but at the same time to a longer training time. In this parameter analysis we decided to limit the generation to three keywords maximum and to plot the text generation statistics we already saw in 4.3.3.3. We will analyze BLEU and ROUGE scores to understand the syntactic correspondence while the BERT score for the semantics.

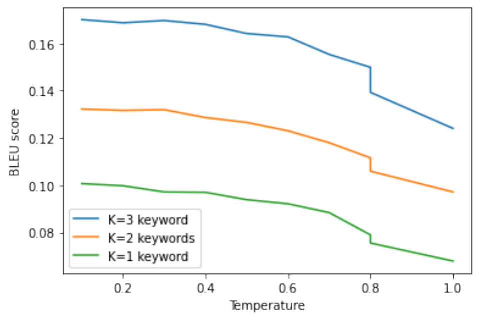


(a) Rouge1 score

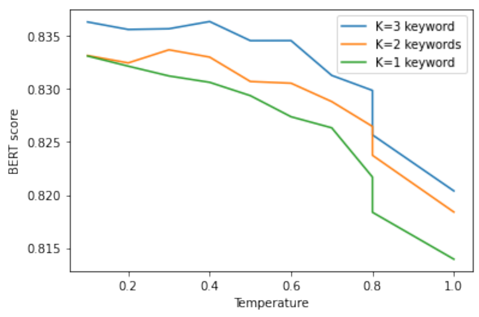


(b) RougeL score

As we can see from FIG.4.27a and FIG.4.27b the ROUGE-1 score reaches its highest points with `K=3 keywords` while a different trend can be seen for the ROUGE-L which shows its best behaviour for `K=2 keywords`. ROUGE-L its based on the *longest common subsequence*, even if there is a smaller number of input words it seems that the `K=2` performs better in this aspect.



(a) BLEU score



(b) BERT score

As concerns BLEU and BERT score, the results are reported in FIG.4.27a and FIG.4.27b. The charts cleary shows how a stronger input signal can give better results. It must be noted that from a semantic point of view the performances are very close, for a value of temperature of 0.8, which can be considered plausible for the final text generation, `K=2 and K=3` achieve the same results.

Chapter 5

Conclusions and future works

Generative Adversarial Networks gained a lot of success lately. They showed incredible performance in different tasks and domains, becoming one of the best way of generating synthetic data when real data is missing. In fact, thanks to this approach, it is possible to correct various underperforming models which suffer from the lack of data. During our thesis work we also explored the potentiality of this functionality by trying to solve a sub-problem of the lack of data related to class imbalance. By training different models on different data, we were able to generate a completely balanced working dataset to fully appreciate the skills of general and specific classifiers for sentiment analysis.

Even if GANs have demonstrated great performances in different tasks and domains, the problems and difficulties related to Natural Language Processing and above all the discrete nature of data treated, are still evident. To analyze the goodness of this approach we decided to perform different tasks by using different architectures. By comparing all of the experiments made we were able to understand if the GAN approach can represent a valid alternative to classical language model.

We started from the basic GAN approach to text data, by developing the SeqGAN [9]. Even if this approach represent a huge milestone, it is the first GAN based model working with textual data, its performances are quite delusional. Above all in such a niche field as it is the Financial one, the lack of some guiding signals results in a very generic generation which is quite far from the real distribution.

By simply comparing this approach to its language model counterpart GPT-2 small [37], it is possible to notice some differences and advantages in favour of the latter.
Once understood that simple architectures can not be as competitive as large language models, we decided to explore the power of these last and try to create a GAN based model. Thanks to the work of Professor Qingyang Wu [12] we were able to develop a big pipeline which can use both GPT-2 small [37] and RoBERTa [30] as Generator and Discriminator.

Our idea is now to understand if the GAN approach can bring some more creativity and specificity at the same time. To even enforce the generation, we decided to give as input some guiding signals. In fact, by using KeyBERT [33] we were able to extrapolate the most salient keywords for each training sentence. To create the parallelism, we slightly modified the GPT-2 fine-tuning.

Once the two different generations are complete, the results produced are very interesting. The GPT-GAN approach showed to be stronger in the majority of the cases, providing more diversity and better quality. It must be noted that GPT-2 by itself reached performances which are very close, becoming a great rival using less training resources.

After the text generation, a further comparison have been implemented by exploring the classification task. Two models have been used:

- GRU [42]: very simple and lightweight. Its complete training could be done in some minutes.
- FinBERT [15]: very financial dependent and specialized model, high number of parameters used for training resulting in very long running time, even days for very large datasets.

The results of this phase were really surprising. GPT-GAN showed once again to be the best model for GRU classification task. But surprisingly GPT-2 alone performed better with FinBERT. This could be due to the fact that GPT-2 generated sentence were more conventional and in general belonging to a more traditional schema more familiar with the human reference distribution of data, while the GPT-GAN ones were composed by a greater variety of tokens bringing more creativity but at the same time breaking away from the standard schema.

At the end the two models showed equal performances, with GPT-GAN slightly prevailing in most of the tasks. Only the financial domain has been fully analyzed, but it could be interesting to understand if the GAN approach could unlock more performances with more generic domains. It is important to understand that due to resource limitation, we were not able to use GPT-3 as our initial intention was. GPT-3 showed to be very effective also in niche domains, solving some problems of its predecessor GPT-2. We firmly believe that by switching to the newest GPT model, we will be able to reach even greater performances.

As concerns the model deployment, very different functionalities have been provided to the user to fully explore the potentiality of our work. From text generation with different languages, to real time dataset re-balancing and custom fine-tuning, we believe that all the key steps of our entire thesis work have been analyzed. In the future it could be interesting to try to develop a complete platform in which different users can work together and share their discoveries.

Bibliography

[1] "Actor-critic model." [Online]. Available: https://www.kdnuggets.com/2018/03/5-things-reinforcement-learning.html

[2] Y. Tang, "Three Types of Recurrent Neural Networks - Towards AI," 2 2022. [Online]. Available: https://pub.towardsai.net/three-types-of-recurrent-neural-networks-567b4e9c4261?gi=ba5d6013a7a7

[3] A. Severyn and A. Moschitti, "Unitn: Training deep convolutional neural network for twitter sentiment classification," in *International Workshop on Semantic Evaluation*, 2015.

[4] V. Jain, "Everything you need to know about "Activation Functions" in Deep learning models," 12 2021. [Online]. Available: https://towardsdatascience.com/everything-you-need-to-know-about-activation-functions-in-deep-learning-models

[5] "Overview of GAN Structure." [Online]. Available: https://developers.google.com/machine-learning/gan/gan_structure

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017. [Online]. Available: https://arxiv.org/abs/1706.03762

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2018. [Online]. Available: https://arxiv.org/abs/1810.04805

[8] M. Saifee, "GPT-3: The New Mighty Language Model from OpenAI - Towards Data Science," 2 2022. [Online]. Available: https://towardsdatascience.com/gpt-3-the-new-mighty-language-model-from-openai-a74ff35346fc

[9] L. Yu, W. Zhang, J. Wang, and Y. Yu, "Seqgan: Sequence generative adversarial nets with policy gradient," 2016. [Online]. Available: https://arxiv.org/abs/1609.05473

[10] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and

D. Amodei, “Language models are few-shot learners,” 2020. [Online]. Available: https://arxiv.org/abs/2005.14165

[11] J. Alammar, “The Illustrated GPT-2 (Visualizing Transformer Language Models).” [Online]. Available: https://jalammar.github.io/illustrated-gpt2/

[12] Q. Wu, L. Li, and Z. Yu, “Textgail: Generative adversarial imitation learning for text generation,” 2020. [Online]. Available: https://arxiv.org/abs/2004.13796

[13] O. Community, “Reinforcement Learning with PPO,” 9 2021. [Online]. Available: https://opendatascience.com/reinforcement-learning-with-ppo/

[14] G. Loye, “Gated Recurrent Unit (GRU) With PyTorch,” 1 2023. [Online]. Available: https://blog.floydhub.com/gru-with-pytorch/

[15] D. Araci, “Finbert: Financial sentiment analysis with pre-trained language models,” 2019. [Online]. Available: https://arxiv.org/abs/1908.10063

[16] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, “Bertscore: Evaluating text generation with bert,” 2019. [Online]. Available: https://arxiv.org/abs/1904.09675

[17] J. Mohajon, “Confusion Matrix for Your Multi-Class Machine Learning Model,” 12 2021. [Online]. Available: https://towardsdatascience.com/confusion-matrix-for-your-multi-class-machine-learning-model-ff9aa3bf7826

[18] “File:ReLU and GELU.svg - Wikimedia Commons,” 11 2020. [Online]. Available: https://commons.wikimedia.org/wiki/File:ReLU_and_GELU.svg

[19] Z. Saleh, “Artificial intelligence definition, ethics and standards,” 04 2019.

[20] B. Mahesh, “Machine learning algorithms -a review,” 01 2019.

[21] W. Zhang, G. Yang, Y. Lin, C. Ji, and M. Gupta, “On definition of deep learning,” 06 2018, pp. 1–5.

[22] S. Sah, “Machine learning: A review of learning types,” 07 2020.

[23] A. Hammoudeh, “A concise introduction to reinforcement learning,” 02 2018.

[24] W. van Heeswijk, PhD, “The Four Policy Classes of Reinforcement Learning - Towards Data Science,” 12 2022. [Online]. Available: https://towardsdatascience.com/the-four-policy-classes-of-reinforcement-learning-38185daa6c8a

[25] C. Cappi, C. Chapdelaine, L. Gardes, E. Jenn, B. Lefèvre, S. Picard, and T. Soumarmon, “Dataset definition standard (DDS),” *CoRR*, vol. abs/2101.03020, 2021. [Online]. Available: https://arxiv.org/abs/2101.03020

[26] S. Joseph, K. Sedimo, F. Kaniwa, H. Hlomani, and K. Letsholo, “Natural language processing: A review,” *Natural Language Processing: A Review*, vol. 6, pp. 207–210, 03 2016.

[27] M. Karakaya, “Fundamentals of Text Generation,” 12 2022. [Online]. Available: https://www.muratkarakaya.net/2022/11/fundamentals-of-text-generation.html

[28] "Text Classification: What it is And Why it Matters." [Online]. Available: https://monkeylearn.com/text-classification/

[29] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014. [Online]. Available: https://arxiv.org/abs/1406.2661

[30] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," 2019. [Online]. Available: https://arxiv.org/abs/1907.11692

[31] Malo et Al., "Financialphrasebank," 2014.

[32] "What is the difference between categorical, ordinal and interval variables?" [Online]. Available: https://stats.oarc.ucla.edu/other/mult-pkg/whatstat/what-is-the-difference-between-categorical-ordinal-and-interval-variables/

[33] M. Grootendorst, "Keybert: Minimal keyword extraction with bert." 2020. [Online]. Available: https://doi.org/10.5281/zenodo.4461265

[34] Anonymous (Under review), "Quantifying exposure bias for neural language generation," 2022. [Online]. Available: https://openreview.net/forum?id=rJg2fTNtwr

[35] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in Neural Information Processing Systems*, S. Solla, T. Leen, and K. Müller, Eds., vol. 12. MIT Press, 1999. [Online]. Available: https://proceedings.neurips.cc/paper/1999/file/464d828b85b0bed98e80ade0a5c43b0f-Paper.pdf

[36] GeeksforGeeks, "ML Monte Carlo Tree Search MCTS," 7 2022. [Online]. Available: https://www.geeksforgeeks.org/ml-monte-carlo-tree-search-mcts/

[37] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, "Language models are unsupervised multitask learners," 2018. [Online]. Available: https://life-extension.github.io/2020/05/27/GPTæŁĂæœŕåĹİæŐć/language-models.pdf

[38] P. J. Liu, M. Saleh, E. Pot, B. Goodrich, R. Sepassi, L. Kaiser, and N. Shazeer, "Generating wikipedia by summarizing long sequences," 2018. [Online]. Available: https://arxiv.org/abs/1801.10198

[39] Z. Elhamraoui, "Fine-tuning in Deep Learning - Artificial Intelligence in Plain English," 12 2021. [Online]. Available: https://ai.plainenglish.io/fine-tuning-in-deep-learning-909666d4c151

[40] J. Ho and S. Ermon, "Generative adversarial imitation learning," 2016. [Online]. Available: https://arxiv.org/abs/1606.03476

[41] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," 2015. [Online]. Available: https://arxiv.org/abs/1508.07909

[42] L. Kurniasari and A. Setyanto, "Sentiment analysis using recurrent neural

network," *Journal of Physics: Conference Series*, vol. 1471, no. 1, p. 012018, feb 2020. [Online]. Available: https://dx.doi.org/10.1088/1742-6596/1471/1/012018

[43] S. Kostadinov, "Understanding GRU Networks - Towards Data Science," 11 2019. [Online]. Available: https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be

[44] FastAPI. [Online]. Available: https://fastapi.tiangolo.com

[45] Streamlit. [Online]. Available: https://streamlit.io

[46] Huggingface, "https://huggingface.com."

[47] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu, "Bleu: a method for automatic evaluation of machine translation," 2002. [Online]. Available: https://aclanthology.org/P02-1040.pdf

[48] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," 2016. [Online]. Available: https://arxiv.org/abs/1606.08415