UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Métodos de detecção automática e incremental de textos gerados por modelos de linguagem

Guilherme Rodrigues Barbazza

Monografia - MBA em Inteligência Artificial e Big Data

ICMC USP SÃO CARLOS

Guilherme Rodrigues Barbazza

Métodos de detecção automática e incremental de textos gerados por modelos de linguagem

Monografia apresentada ao Departamento de Ciências de Computação do Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo - ICMC/USP, como parte dos requisitos para obtenção do título de Especialista em Inteligência Artificial e Big Data.

Área de concentração: Inteligência Artificial

Orientador: Prof. Dr. Tiago Agostinho de Almeida

Versão original

São Carlos

2023

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi, ICMC/USP, com os dados fornecidos pelo(a) autor(a)

Barbazza, Guilherme Rodrigues
S856m Métodos de detecção automática e incremental de textos gerados por modelos de linguagem / Guilherme Rodrigues Barbazza ; orientador Tiago Agostinho de Almeida. – São Carlos, 2023.
54 p. : il. (algumas color.) ; 30 cm.

Monografia (MBA em Inteligência Artificial e Big Data) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2023.

1. Categorização de texto. 2. Modelos Generativos de Linguagem . 3. Large Language Models. 4. Aprendizado Incremental. I. Agostinho de Almeida, Tiago, orient. Almeida, Tiago Agostinho, orient. II. Título.

Guilherme Rodrigues Barbazza

Métodos de detecção automática e incremental de textos gerados por modelos de linguagem

Monograph presented to the Departamento de Ciências de Computação do Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo - ICMC/USP, as part of the requirements for obtaining the title of Specialist in Artificial Intelligence and Big Data.

Concentration area: Artificial Intelligence

Advisor: Prof. Dr. Tiago Agostinho de Almeida

Original version

São Carlos

2023

RESUMO

Barbazza,G,R **Métodos de detecção automática e incremental de textos gerados por modelos de linguagem**. 2023. 54p. Monografia (MBA em Inteligência Artificial e Big Data) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2023.

Modelos de linguagem natural são sistemas computacionais que podem gerar ou processar textos em linguagem natural. Eles são baseados em técnicas de aprendizado profundo, também conhecidas como *deep-learning*. Os modelos de linguagem generativa, especificamente, podem produzir textos a partir de um contexto ou uma entrada inicial. Estes modelos têm avançado nos últimos anos, graças ao desenvolvimento de arquiteturas mais sofisticadas e ao aumento da disponibilidade de dados e recursos computacionais, sendo a quantidade de parâmetros que possuem um dos fatores que influencia seus desempenhos. Se por um lado, os modelos de linguagem generativa podem ser usados para diversas aplicações, como resumir textos, tradução e responder perguntas, podem, também, serem utilizados para fins maliciosos. Diante deste cenário de lançamentos cada vez mais frequentes de modelos geradores de textos, torna-se imperativo a necessidade da criação de uma linha de defesa para identificação de textos sintéticos, que seja capaz de responder rapidamente à evolução que esses modelos trazem a cada geração lançada. Este trabalho investiga o desempenho dos classificadores com abordagens clássicas, que possuem capacidade de aprendizado incremental, na detecção de textos gerados por humanos e sintéticos, comparando-os com um modelo considerado o estado da arte. Para os métodos de classificação com abordagem clássica, são utilizadas diferentes técnicas de representação de palavras visando obter o melhor desempenho. Para a realização do experimento, é considerado o seguinte cenário: treinamento de classificadores de textos com amostras de dados produzidas por humanos e gerados por um pequeno modelo de linguagem generativa. Posteriormente, estes classificadores recebem dados de um modelo com mais parâmetros. Assim, é aferido o desempenho destes classificadores neste cenário de transição, simulando o lançamento de um modelo de linguagem generativa mais atual. Como os classificadores são criados a partir de algoritmos que suportam atualizações dos parâmetros sem a necessidade do retreinamento completo (*online-learning*), passam por um processo de atualização de seus parâmetros para que seja possível medir novamente os seus desempenhos neste cenário transitório. Neste contexto, este trabalho tem como objetivo analisar a deterioração da performance dos classificadores e investigar como a atualização de seus parâmetros afeta seus desempenhos.

Palavras-chave: Modelos de linguagem natural. Modelos geradores de texto. Métodos de classificação. Deep-learning. Detecção de textos sintéticos. Online-learning.

ABSTRACT

Barbazza,G,R **Automatic and incremental methods for detecting text generated by language models**. 2023. 54p. Monograph (MBA in Artificial Intelligence and Big Data) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2023.

Natural language models are computational systems that can generate or process natural language texts. They are based on deep learning techniques, also known as *deep-learning*. Generative language models, specifically, can produce consistent and relevant text from context or initial input. These models have been advanced in recent years, thanks to the development of more sophisticated architectures and the increase in the availability of data and computational resources, with a number of functionalities being one of the factors that influence their performance. On the one hand, generative language models can be used for various applications, such as summarizing texts, translating and answering questions, they can also be used for malicious purposes. Faced with this scenario of increasingly frequent launches of text generating models, it is imperative to create a line of defense for the identification of synthetic texts, which is capable of responding quickly to the evolution that these models bring with each generation launched. This work investigates the performance of classifiers with classical approaches, which have incremental learning capacity, in detecting texts generated by humans and synthetics, comparing them with a model considered the state of the art. For classification methods with a classical approach, different word representation techniques are used to obtain the best performance. To carry out the experiment, the following scenario is considered: training text classifiers with data samples produced by humans and generated by a small generative language model. Subsequently, these classifiers receive data from a model with more parameters. Thus, the performance of these classifiers is measured in this transition scenario, simulating the launch of a more current generative language model. As classifiers are created from algorithms that support parameter updates without the need for complete retraining (*online-learning*), they go through a process of updating their interruptions so that it is possible to measure their performance again in this transient scenario . In this context, this work aims to analyze the variation in the performance of classifiers and investigate how updating their parameters affects their performance.

Keywords: Natural language models. Text generating models. Methods of classification. Deep-learning. Detection of synthetic texts. Online-learning.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

BERT	Bidirectional encoder representations from transformers
DNN	Deep neural networks
FN	False negatives
FP	False positives
GNB	Gaussian naive Bayes
GPT	Generative pretrained transformer
GPU	Graph processing unit
GRU	Gated recurrent units
LLM	Large language model
LSTM	Long short term memory
NLG	Natural language generation
PA	Passive aggressive
PLN	Processamento de línguas naturais
RNN	Recurrent neural networks
RoBERTa	Robustly optimized BERT pretraining approach
SGD	Stochastic gradient descent
TF-IDF	Term frequency - inverse document frequency
TN	True negatives
TP	True positives

SUMÁRIO

1 INTRODUÇÃO

O processamento de línguas naturais (*PLN*) é uma área em constante desenvolvimento com desafios intrínsecos do ramo do estudo de línguas naturais. Existem aplicações em uma variedade de campos, destacando-se principalmente em *chatbots*, tradutores automáticos, classificação de tópico e análise de sentimentos textuais.

Uma área que vem recebendo grande destaque recentemente é a geração de linguagem natural (do inglês, *natural language generation – NLG*), na qual são estudados e desenvolvidos métodos computacionais capazes de gerar textos semelhantes ao humano. Com a evolução deste tipo de modelo, espera-se que os algoritmos sejam capazes de criar textos cada vez mais complexos e sofisticados, abrindo novas possibilidades para a geração automática de conteúdo relevante em áreas mais técnicas e específicas. Em especial, uma classe de modelos geradores têm chamado a atenção: os modelos de linguagem massivos (do inglês, *Large Language Models* – LLMs). Os *LLMs* contemplam os algoritmos estado da arte em geração de texto como, por exemplo, o ChatGPT, disponibilizado publicamente pela OpenAI [1] .

A capacidade do *ChatGPT* e de outros *LLMs* de gerar textos convincentes tem sido pauta de diversas discussões. Embora seja uma ferramenta que pode liberar as pessoas de tarefas monótonas e repetitivas, há um potencial risco de uso mal-intencionado, como, por exemplo, sofisticação de escrita de notícias falsas. A medida que estes modelos evoluem, discernir entre um texto escrito por máquina e por humano torna-se mais difícil e por consequência, a criação de uma linha de defesa que permita a identificação de textos gerados por máquinas através de modelos classificadores torna-se ao mesmo tempo uma tarefa imperativa e também desafiadora, pois a complexidade dos *LLMs* geradores de texto tende a aumentar a cada geração.

Os modelos detectores de texto estado da arte são baseados em redes neurais, em especial o modelo *RoBERTa* (LIU *et al.*, 2019) (do inglês, *Robustly Optimized BERT Pretraining Approach*). Via de regra, estes classificadores são treinados sob a ótica do paradigma de aprendizagem *off-line*. Neste formato, um conjunto de dados é apresentado uma única vez ao classificador e, conforme novos modelos geradores de texto são lançados, estes classificadores tendem a ficar obsoletos, pois a cada nova geração os *LLMs* tornam-se mais otimizados em gerar textos convincentes ao leitor humano. Isto implica em uma mudança na distribuição probabilísticas dos textos gerados, impactando negativamente a performance dos classificadores. Portanto, é imperativo que os detectores tenham a capacidade de se adaptar rapidamente a mudança de cenário, que nos últimos meses têm se mostrado bastante dinâmica, com a inundação de novos *LLMs* prontos para serem

[1] Disponível em: www.chat.openai.com. Acessado em: 30/11/2023

usados pelos usuários finais.

Uma forma de responder rapidamente a esta mudança de cenário é a utilização de classificadores mais tradicionais que possibilitam o aprendizado incremental (em inglês, *incremental-learning*). Este paradigma permite que os modelos sejam readaptados com novos conjuntos de dados sem a necessidade do retreinamento completo dos classificadores, permitindo que o seu desempenho não deteriore ao longo do tempo (SILVA; ALMEIDA, 2021).

1.1 Hipótese e objetivos

Atualmente, os classificadores de texto baseados no modelo *RoBERTa* apresentam um bom desempenho em discriminar textos gerados pelos *LLMs* atuais. Porém, nada garante que estes mesmos classificadores mantenham a performance com relação a *LLMs* lançados futuramente. Assumir esta premissa é uma forma ingênua de tratar este problema, pois estudos apontam que a simples mudança para *LLMs* com diferentes quantidades de parâmetros já compromete significativamente o desempenho desses discriminadores (SOLAIMAN *et al.*, 2019; IPPOLITO *et al.*, 2019). Além disso, as redes neurais exigem grandes volumes de dados e poder computacional a disposição para que sejam readaptados a novos dados de treinamento (KOWSARI *et al.*, 2019), tornando esta tarefa trabalhosa e sem garantias que a performance se mantenha a mesma para os dados do passado (em inglês, *back-testing*). Neste contexto, este trabalho assume a hipótese de que é possível treinar classificadores de detecção automática de textos gerados por IA e humanos usando algoritmos que permitam o aprendizado incremental com desempenho razoável, com o objetivo de obter modelos classificadores que se adaptem rapidamente aos lançamentos mais recentes de *LLMs* geradores de texto.

2 FUNDAMENTAÇÃO TEÓRICA

Como um computador não consegue processar a linguagem natural, é necessário que um texto seja computacionalmente representável e que essa representação seja significativa o suficiente para que um classificador de texto consiga distinguir os registros (KOWSARI *et al.*, 2019). Diante disto, neste capítulo serão apresentadas as técnicas mais tradicionais para que um texto humano passe a ser computacionalmente representado. Também, serão mencionados brevemente os principais métodos utilizados para categorização de textos e formas de análise de seus resultados.

2.1 Vocabulários

Vocabulário é um conjunto de palavras ou, de uma forma mais abrangente, *tokens* que são utilizados para representar o texto escrito em linguagem natural pelos computadores. Em termos mais específicos, um vocabulário pode ser definido como um conjunto finito de elementos, cada um dos quais é associado a um índice. No contexto de modelos de linguagem natural, os elementos do vocabulário são os *tokens*, e os valores são os índices desses elementos. A criação de vocabulário é um passo fundamental, pois delimita a quantidade de palavras passíveis de serem processadas. Nas próximas seções, são apresentadas as principais técnicas utilizadas.

2.1.1 *Tokenization*

A *tokenization* é uma ténica que separa dados não estruturados como textos em partes elementares, denominados *tokens*.

2.1.2 *n-Grams*

Um *n-gram* é uma sequência de palavras contendo n elementos. No geral, os *n-grams* são criados por caracteres, palavras e símbolos. Eles são uma forma de melhorar o nível de importância das palavras em um *corpus* de texto, separando-as em pares pelo menos.

Utilizando o exemplo:

"O rato roeu a roupa do rei de Roma"

No processo de unigrama:

"o", "rato", "roeu", "a", "roupa","do","rei","de","roma"

Na representação em bigramas:

"o rato", "rato roeu", "roeu a", "a roupa", "roupa do"

2.2 Normalização

Nas próximas seções, são apresentadas técnicas que visam padronizar os atributos textuais. Esta etapa possibilita que os modelos sejam mais eficientes do ponto de vista computacional, consumindo menos memória e exigindo menos poder computacional de processamento (KOWSARI *et al.*, 2019).

2.2.1 *Stemming*

Uma técnica comum de normalização de vocabulário é eliminar as sutis diferenças de pluralização, terminações e variações verbais das palavras (LANE; HOWARD; HAPKE, 2019). Esta normalização, identifica um radical comum entre as palavras. O resultado final do *stemming* pode gerar palavras gramaticalmente erradas, porém semelhantes.

Com base no exemplo anterior, o vocabulário resultante após a técnica de *stemming* seria:

"o", "rat", "roeu", "a", "roup", "do", "rei", "de", "rom"

2.2.2 *Lemmatization*

As palavras são reduzidas às suas raízes semânticas, denominado lema.

O vocabulário resultante após a técnica de *lemmatization* seria:

"o", "rato", "roeu", "o", "roupa", "de o", "rei", "de", "roma"

2.2.3 Remoção de *stop-words*

Stop-words são palavras que ocorrem com alta frequência, mas normalmente não são importantes para o desempenho do modelo. Portanto, são palavras que geralmente são removidas para suavizar ligeiramente o tamanho do vocabulário e preservar palavras com mais importância.

O vocabulário resultante após a técnica de remoção de *stop-words* seria:

"rato", "roeu", "roupa", "rei", "roma"

2.3 Representação de textos

De acordo com Kowsari *et al.* (2019), algumas formas de representação alteram de forma significativa a qualidade de classificação dos modelos. Nas seções seguintes, são apresentadas as principais formas de representar textos computacionalmente e uma

breve análise sobre seu desempenho. Estas, em síntese, têm o objetivo de transformar as palavras em uma representação para que os textos possam ser processados pelos modelos computacionais.

2.3.1 *Bag of words*

O termo *bag of words* designa-se à maneira de representar as sentenças por meio de um conjunto das palavras, com o objetivo de criar uma representação vetorial.

Considere as duas sentenças abaixo.

Sentença 1:

"Preparando o TTC do MBA IA e Big Data USP"

Sentença 2:

"Preparando um modelo de IA em uma base de dados de Big Data"

Aplicando o processo de *tokenização* nestas duas sentenças e utilizando o processo de eliminação de *stop-words* descrito na Seção 2.2.3 , a representação vetorial resultante para as respectivas sentenças será:

Sentença	"base"	"big"	"dados"	"data"	"ia"	"mba"	"modelo"	"preparando"
1	0	1	0	1	1	1	0	1
2	1	1	1	1	1	0	1	1

Porém, a representação binária pode não ser uma boa representação das palavras pois acaba não fazendo distinção entre palavras que tem um potencial de importância maior.

2.3.2 *Term frequency - inverse document frequency*

O *Term frequency - inverse document frequency* (*TF-IDF*) é uma forma de ponderar palavras mais importantes em um conjunto de dados textuais. As representações de contagem de termos costumam apresentar um melhor desempenho quando utilizadas como representações de palavras em tarefas que naturalmente as sentenças são pequenas (por exemplo, classificação de mensagens SMS) se comparados a técnicas mais complexas (KOWSARI *et al.*, 2019), que costuma apresentar um melhor desempenho ao representar sentenças mais longas.

A representação matemática do *TF* é dada por:

$$\mathrm{TF}(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \tag{2.1}$$

Onde:

$f_{t,d}$ - Contagem da palavra t no documento d

O denominador representa a quantidade de palavras no documento d

Tomando como exemplo as sentenças 1 e 2 citadas anteriormente, o cálculo do *term-frequency* será:

TF - Sentença 1

preparando	tcc	mba	ai	bia	data
0.2	0.2	0.2	0.2	0	0

TF - Sentença 2

preparando	modelo	ia	base	dados
0.25	0.25	0.25	0.25	.25

A representação matemática do *IDF* é dada por:

$$\mathrm{IDF}(t,D) = \log \frac{N}{|\{d \in D : t \in d\}|} \tag{2.2}$$

Onde:

N - Número de documentos no *corpus* $N = |D|$

$|\{d \in D : t \in d\}|$ - Número de documentos em que a palvra t aparece

Desta forma, obtêm-se uma representação vetorial de cada palavra do vocabulário, levando em consideração suas frequências com relação ao documento em que ela aparece e também comparando-se com os documentos restantes. Apesar do *TF-IDF* contabilizar termos mais comuns diferentemente de termos mais raros, este método é incapaz de contabilizar a similaridade entre palavras, pois elas são representadas individualmente como um *index* e também levam em conta apenas a estrutura sintática das sentenças (KOWSARI *et al.*, 2019). Modelos mais recentes de representação de palavras, denominados *word embbedings*, são capazes de incorporar a similaridade das palavras, além das estruturas sintática e semântica das frases.

2.3.3 *Word embbedings*

A representação sintática de palavras não garante que o modelo capture o significado semântico. Por exemplo, palavras como “avião”, “aeroplano” e “aeronave” são

frequentemente usadas no mesmo contexto, de acordo com a hipótese distribucional de palavras. No entanto, os vetores correspondentes a essas palavras são ortogonais no modelo de *BoW*.

As *word embeddings*, também conhecidas como representações vetoriais de palavras, são técnicas de processamento de linguagem natural que mapeiam palavras em vetores densos. Existem diversos algoritmos para gerar *word embeddings*, sendo que os mias conhecidos são:

- *Word2Vec*: modelo que utiliza redes neurais rasas para gerar as *word embeddings* (MIKOLOV *et al.*, 2013; KOWSARI *et al.*, 2019). O *Word2Vec* tem duas abordagens: *Skip-gram* e *Continuous Bag of Words*.

- *GloVe*: técnica de aprendizado de máquina que utiliza uma matriz de co-ocorrência de palavras para gerar *word embeddings* (PENNINGTON; SOCHER; MANNING, 2014).

2.4 Classificadores de textos

Classificadores de texto são algoritmos de aprendizado de máquina que categorizam automaticamente o texto em diferentes classes ou categorias com base em seu conteúdo (*features*). De uma forma geral, a construção de algoritmos classificadores segue alguns passos básicos. O objetivo final de um classificador é ter seus parâmetros ajustados de forma que o erro entre as saídas do classificador e os dados de treino seja o menor possível (SEBASTIANI, 2002)

2.4.1 Algoritmos incrementais mais tradicionais

Nesta seção, será brevemente descrito os algoritmos classificadores que são habilitados ao aprendizado incremental.

- *Classificador Naive Bayes*: baseia-se no teorema de *Bayes* que utiliza a probabilidade condicional para classificar objetos em diferentes categorias. Ele assume independência entre as variáveis e estima as probabilidades usando os dados de treinamento. (WEBB; BOUGHTON; WANG, 2005).

- *Classificador SGD*: utiliza o algoritmo de otimização conhecido como Gradiente Descendente Estocástico (do inglês, *Stochastic Gradient Descent – SGD*) para encontrar os pesos que minimizam a função de perda (em inglês, *loss function*). Esse algoritmo de otimização é eficiente em relação ao uso de memória e a velocidade de treinamento, especialmente em grandes conjuntos de dados, pois fornece a possibilidade de treinamento em formato *mini-batch* (ZHANG, 2004). Isto significa que os dados

de treinamento são quebrados em pequenos “pacotes“ e passados gradativamente ao modelo. Desta forma é possível acompanhar a aprendizagem do modelo nas instâncias de treinamento (SKLEARN-SGDCLASSIFIER, 2023). Este tipo de treinamento é característico do treinamento de *NN* pois os *datasets* para estes modelos costumam envolver grandes quantidade de dados.

- *Passivo agressivo*: é um algoritmo de aprendizado de máquina habilitado para o paradigma de aprendizado online. Ele é baseado na ideia de ser “passivo” quando o modelo atual classifica corretamente um exemplo de treinamento, mas “agressivo” quando comete um erro (CRAMMER *et al.*, 2006). O método passivo-agressivo é útil em situações onde os dados são recebidos sequencialmente e o modelo precisa ser atualizado continuamente. Ele também pode ser usado em cenários de grande escala, onde é computacionalmente inviável treinar o modelo em todo o conjunto de dados de uma só vez.

- *Perceptron*: utiliza uma função de ativação com o objetivo de criar um modelo matemático que gere um hiperplano que separe as duas classes de dados (GALLANT *et al.*, 1990)

2.5 Modelos baseados em Transformers

RoBERTa (LIU *et al.*, 2019), é uma variante do modelo *BERT* introduzida Devlin *et al.* (2018). A principal diferença entre *RoBERTa* e *BERT* é que *RoBERTa* usa mascaramento dinâmico em vez de mascaramento estático. No *BERT*, a tarefa de modelo de linguagem mascarada envolve mascarar uma porcentagem dos *tokens* de entrada aleatoriamente e, em seguida, prever esses *tokens* mascarados. No modelo *RoBERTa*, o padrão de mascaramento é alterado dinamicamente durante o treinamento. Isso permite que o modelo se adapte aos dados de maneira mais eficaz.

2.6 Métricas de analises de resultados

Nesta seção, serão discutidas algumas formas de análise dos resultados dos classificadores.

2.6.1 Matriz de confusão

A matriz de confusão é uma matriz de valores inteiros não negativos (SUSMAGA, 2004) que mostra o desempenho de um modelo de classificação em um conjunto de dados de teste. Compara-se as previsões do modelo com as classes reais disponibilizadas durante o processo de separação mencionado na Seção 2.4. É uma ferramenta útil para avaliar o comportamento de um modelo e verificar quais os casos em que o modelo obtém o melhor desempenho.

A matriz de confusão é composta por quatro células:

- *True Positives* (TP): casos em que o modelo previu corretamente uma classe positiva.
- *False Positives* (FP): casos em que o modelo previu incorretamente uma classe positiva.
- *True Negatives* (TN): casos em que o modelo previu corretamente uma classe negativa.
- *False Negatives* (FN): casos em que o modelo previu incorretamente uma classe negativa.

Accuracy: Proporção de predições corretas sobre todas as predições. Pode ser calculada como:

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN} \tag{2.3}$$

Specificity: É a proporção de predições negativas corretas. Pode ser calculado como:

$$\text{Specificity} = \frac{TN}{FP + TN} \tag{2.4}$$

Recall (ou sensitivity): a proporção de verdadeiros positivos em relação ao total de positivos reais. É uma medida de quão bem o modelo identifica os casos positivos e é calculada como:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.5}$$

Precision: a proporção de verdadeiros positivos em relação ao total de casos positivos previstos. É uma medida de quão bem o modelo evita prever casos falsos positivos e é calculada como:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.6}$$

F1 Score: medida de desempenho do modelo que combina *recall* e *precision* em uma única métrica. É a média harmônica entre *recall* e *precision* e é calculada como:

$$F = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.7}$$

Geralmente, as métricas de *F1 Score* podem ser calculadas considerando as classes:

- *Micro Average*: Calcula a métrica global contabilizando o total de TP, FN e FP.
- *Macro Average*: Calcula métricas para cada classe e computa a média não ponderada. Portanto, não incorpora o desbalanceamento das classes.

As limitação mais conhecidas das métricas supracitadas são: *Accuracy* é uma medida que não fornece informações sobre Falso Negativo (FN) e Falso Positivo (FP); O *recall*, por sua vez, não avalia TN e FP. Qualquer classificador que preveja pontos de dados como positivos é considerado de alto *recall*; A *specificity* é semelhante ao *recall* e também não leva em conta FN e TP. Já a *precision* não avalia TN e FN e é considerada uma medida muito conservadora, sendo mais utilizado em cenários que existe uma grande probabilidade de ocorrem casos positivos (KOWSARI *et al.*, 2019).

A avaliação experimental de classificadores de texto mede a eficácia (ou seja, a capacidade de fazer o classificação correta ou decisão de categorização). A *precision* e o *recall* são amplamente usados para medir a eficácia dos classificadores de texto. Portanto, uma medida agregadora que envolva as duas métricas, como *F1 Score* macro e micro são mais utilizadas (KOWSARI *et al.*, 2019).

3 CONTEXTUALIZAÇÃO BIBLIOGRÁFICA

Com base nos trabalhos publicados nos últimos anos, é perceptível uma mudança acelerada no paradigma das arquitetura dos modelos de *NLG*. Os modelos mais recentes são tipicamente redes neurais profundas (do inglês, *deep neural networks – DNN*), ou seja, redes neurais com diversas camadas escondidas, treinadas em grandes *corpora* de texto escrito por humanos (CELIKYILMAZ; CLARK; GAO, 2020). As mudanças mais significativas iniciaram com os modelos recorrentes (do inglês, *Recurrent Neural Networks – RNN)* (CELIKYILMAZ; CLARK; GAO, 2020; CHO *et al.*, 2014) e *Gated Recurrent Units GRUs*) (CELIKYILMAZ; CLARK; GAO, 2020; GRAVES, 2013) no aprendizado de representação de linguagem.

Um problema característico das *DNNs* é que, inicialmente, só podiam ser aplicadas a problemas cujas entradas eram ajustáveis vetorialmente a uma dimensão fixa. Esta era uma limitação significativa porque, para muitos problemas, não é possível saber o tamanho da sequência de entrada antecipadamente (SUTSKEVER; VINYALS; LE, 2014). Para contornar esta limitação, foi proposta uma arquitetura denominada *seq2seq*, utilizando-se duas *Long Short Term Memory* (*LSTM*) que partem de uma arquitetura similar ao das *RNNs* e *GRUs*, dando origem a uma gama de novas aplicações baseadas nestes novos modelos (CELIKYILMAZ; CLARK; GAO, 2020). Os modelos *seq2seq* estão limitados ao tamanho do texto de entrada e, consequentemente, não conseguem identificar dependências entre as palavras quando a sequências é grande. Isto ocorre porque os estados ocultos (*hidden states*, em inglês) contabiliza toda a sequência de entrada antes de transmiti-la ao *decoder*. Assim, sequências que são grandes o suficiente podem ocasionar o problema de *vanishing gradient* ou tornar o modelo demasiadamente lento em suas previsões. Estas limitações estimularam o surgimento de *DNNs* com camadas de atenção (em inglês, *attention netwoks*) (CELIKYILMAZ; CLARK; GAO, 2020; BAHDANAU; CHO; BENGIO, 2014). O mecanismo de atenção permite que o modelo pese dinamicamente a importância de cada elemento na sequência de entrada. Calculando estes pesos de atenção, o modelo torna-se capaz de selecionar os elementos mais relevantes para a geração da próxima saída.

A arquitetura *transformer* incorpora o conceito de *encoder-decoders* com camadas de atenção e atualmente figura no estado-da-arte dentre os modelos de *NLG*. Recentemente, modelos foram pré-treinados em *corpora* massivos de texto para a representação de palavras contextualizadas, resultando em modelos densos especificamente utilizados para a geração de representação dinâmica das palavras de acordo com seu contexto. Os modelos contextualizados são tipicamente treinados em grandes *corpora* de dados como, por exemplo, *Wikipedia*, notícias, e artigos científicos. Os modelos mais conhecidos são baseados na arquitetura *transformer*, com destaque para: *GPT*, *BERT* e, seu aprimoramento, *RoBERTa*

(RADFORD *et al.*, 2019; DEVLIN *et al.*, 2018; LIU *et al.*, 2019).

GPT (*Generative Pretrained Transformer*) é um modelo desenvolvido pela *OpenAI*. A história dos modelos *GPT* inicia em 2018 com a introdução de sua primeira versão (RADFORD *et al.*, 2018). O *GPT-3*, uma das versões mais atuais disponível publicamente de forma gratuita, é um dos maiores modelos de linguagem já treinado, com centena de bilhões de parâmetros (175 Bilhões) (BROWN *et al.*, 2020). Ele foi disponibilizado para usuários finais através de um chat. A interação com usuários acontece através de um *prompt* de comandos. Ao enviar um comando, o modelo o interpreta e responde em tempo real. A semelhança com texto gerado por humano é bastante grande, o que acaba por levantar alguns questionamentos importantes.

Apesar de serem uma ferramenta que cria caminhos para livrar as pessoas de tarefas tediosas e repetitivas, existe um potencial de risco de uso mau intencionado da ferramenta. Uma vez integrado à sites, o ChatGPT, que é uma aplicação sustentada pelos *GPT-3.5* ou *GPT-4.0* otimizados para gerarem respostas no formato de *chat*, pode aprimorar a disseminação de desinformação e *fake news*, tornando-as mais apelativas através da análise de perfis de usuário. Por exemplo, é possível gerar um texto personalizado em tempo real no momento em que um usuário trafega em um portal de notícias, tornando mais fácil a tarefa de persuasão de eleitores em períodos de eleição, criando-se discursos específicos, apelativos e, acima de tudo, com um alto nível de personalização de acordo com perfil do usuário, levando a uma manipulação mais sofisticada da opinião pública.

No contexto da educação, a facilidade de uso dessas ferramentas pode levar a uma dependência excessiva na criação de conteúdos por parte dos alunos, o que pode prejudicar o desenvolvimento de habilidades importantes, como a capacidade de pesquisa escrita e a análise crítica. Além disso, o emprego frequente do uso dessas ferramentas pode levar a um aumento considerável na ocorrência de casos de plágio e fraudes, o que pode ter implicações negativas para a integridade acadêmica e a reputação das instituições de ensino.

Outro potencial uso destes modelos é a geração de repostas automáticas em sites ou fóruns para ganhar reputação. O *Stack Overflow*[1], por exemplo, tomou a iniciativa de proibir os usuários de enviarem respostas produzidas utilizando o ChatGPT, depois que o site passou a receber um número anormal de respostas incorretas[2]. Para não incentivar mudanças no padrão das respostas, o *Stack Overflow* não disponibiliza ou cita a forma de detecção adotada.

Em um campo em que a escrita criativa é um diferencial, a revista de ficção científica *Clarkesworld* cancelou seu processo de inscrição porque passaram a receber um número fora das expectativas de submissões desde dezembro de 2022. O pico de submissões

[1] Disponível em: www.stackoverflow.com. Acessado em: 13/02/2023

[2] Disponível em: www.meta.stackoverflow.com/questions/421831/temporary-policy-generative-ai-e-g-chatgpt-is-banned. Acessado em: 13/02/2023

acompanha a data de disponibilização do ChatGPT. A conclusão da revista é que pessoas estavam enviando histórias criadas pelo ChatGPT[3]. Como consequência, optou-se pela total paralisação das submissões[4].

Dentro dos contextos apresentados de potenciais ameaças, o desenvolvimento de detectores de textos sintéticos gerados por *IA* é a primeira linha defensiva. A criação destes detectores não é uma tarefa trivial e exige a comparação e análise de diferentes metodologias. Os detectores que têm apresentado os melhores resultados são baseados no modelo *RoBERTa* e são reconhecidos como os classificadores estado da arte na tarefa de detectar textos gerados pelo *LLM GPT-2* (SOLAIMAN *et al.*, 2019; IPPOLITO *et al.*, 2019; JAWAHAR; ABDUL-MAGEED; LAKSHMANAN, 2020). É valido mencionar que a especificidade para o modelo *GPT-2* acontece, pois a *OpenAI* disponibilizou uma base de dados em inglês, que doravante será referida como *gpt-2-output-dataset*. Essa base contém trechos do *corpus* denominado *WebText*, que é um compilado extraído do fórum *Reddit* selecionados seguindo critérios de relevância de conteúdo, e também trechos de textos gerados por diversas versões do modelo *GPT-2* com quantidade de parâmetros diferentes, sendo: 117M, 345M, 762M e 1542M parâmetros[5].

No estudo conduzido por Solaiman *et al.* (2019), onde o *dataset gpt-2-output-dataset* tornou-se público para fins de pesquisa, foram construídos classificadores utilizando diversos algoritmos, incluindo regressão logística como *baseline*, uma versão otimizada do *GPT-2*[6] para detecção de textos e o modelo *RoBERTa*. O estudo focou na análise de desempenho de detectores de textos gerados pelas diferentes versões do modelo *GPT-2* e concluiu que os classificadores criados a partir do modelo *RoBERTa* obtiveram melhor desempenho quando comparado com os demais classificadores, incluindo o classificador *GPT-2-detector*, atingindo acurácias de mais de 95%. Foi observado também que os trechos gerados pelas versões com maior número de parâmetros são mais difíceis de detectar em comparação com as versões menores do modelo *GPT-2*, indicando que quanto maior a quantidade de parâmetros do *LLM* mais próximo o estilo do texto gerado se aproxima do texto escrito por humanos, e portanto, dificultando a distinção.

Outro fator importante em determinar o sucesso de um *LLM* em evadir a detecção de classificadores e avaliadores humanos é a forma com que os *tokens* são selecionados a partir da lista de possíveis novos *tokens*. O estudo de Ippolito *et al.* (2019), que também utilizou um classificador baseado no modelo *RoBERTa* e a base de dados *gpt-2-output-dataset*, observa que o método de escolha do próximo *token* da sequência de palavras na

[3] Disponível em: www.fortune.com/2023/02/28/chatgpt-inaccuracies-causing-real-harm/ Acessado em: 18/02/2023

[4] Disponível em: www.observer.com/2023/02/science-fiction-magazine-clarkesworld-ban-submission-chatgpt. Acessado em: 18/02/2023

[5] Disponível em: https://github.com/openai/gpt-2-output-dataset Acessado em: 18/02/2023

[6] para evitar confusão entre o modelo detector e gerador de sentenças, o classificador baseado no GPT-2 será referido como *GPT-2-detector*

resposta dos *LLMs*, denominado método de amostragem, influencia de forma significativa o desempenho dos agentes classificadores. Caso o espaço amostral de *tokens* mais prováveis seja limitado, técnica denominada *top-k*, as sentenças geradas por *LLMs* são mais difíceis de serem detectadas por agentes humanos e mais fáceis de serem detectadas por detectores de textos. Isto acontece devido a anormalidades estatísticas na distribuição de probabilidades dos *tokens* gerados. Por outro lado, quando não há limitação do espaço amostral, técnica denominada *top-p*, os textos gerados pelos *LLMs* passam a ser mais facilmente detectados por agentes humanos enquanto que os modelos classificadores passam a apresentar maior taxa de erro. Isto pode ser explicado pela quebra lógica em uma sentença que um *token* de baixa probabilidade causa, sendo facilmente detectado por um avaliador humano devido à distorção na lógica de construção da frase. Este resultado indica falta de entendimento semântico das sentenças por parte dos modelos classificadores. Um resultado importante do estudo é que a mediana de acurácia de agentes humanos treinados para classificar textos é de 74%, resultado aquém dos classificadores baseados na arquitetura RoBERTa desenvolvidos no estudo, que chegou a 88% acurácia em sentenças geradas por LLMs com amostragem *top-k-40*.

A discussão de distorções nas distribuições estatísticas dos textos gerados quando os parâmetros do modelo *LLM* são alterados, seja por versões com diferentes quantidades de parâmetros ou a modificação no método de amostragem dos próximos *tokens* leva a um conceito importante em aprendizagem de máquina: a mudança de conceito (em inglês, *concept-drifting*). A mudança de conceito se refere à mudança nos padrões de dados subjacentes ao longo do tempo. Ou seja, os dados que foram utilizados para treinar os modelos classificadores apresentam distribuições estatísticas diferentes dos dados em que os classificadores estão sendo expostos para reconhecerem, fazendo com que o desempenho destes modelos se deteriorem com o tempo (SILVA; ALMEIDA, 2021; BIESIALSKA; BIESIALSKA; COSTA-JUSSÀ , 2020).

No estudo realizado por Silva e Almeida (2021), foi analisado o impacto que o *concept-drifting* acarreta em classificadores de notícias falsas treinados com dados do período pré-pandemia durante as eleições dos Estados Unidos em 2019 e no ano de 2020, período pandêmico. Os autores constataram que o surgimento repentino de notícias falsas sobre o tratamento de doenças impactou negativamente o desempenho de classificadores de notícias falsas treinados com os dados de 2019, sugerindo que a mudança de conceito, neste caso, surgimento repentino de palavras ligadas ao contexto da saúde nos dados de 2020, é determinante no desempenho de classificadores que não recebem atualizações (paradigma denominado como *offline-learning*). Corroborando com isto, os autores demonstram que, uma vez que os classificadores são incrementados com novos dados, e utilizados modelos habilitados a aprendizagem incremental, suas performances ao longo do tempo continuam a mesma, mitigando ou ao menos anulando o efeito negativo da mudança de conceito. Estes tipos de classificadores beneficiam-se do paradigma de *incremental-learnig* em que *datasets*

de treinamento atualizados são continuamente apresentados aos algoritmos classificadores.

De certa forma, discriminar textos gerados por humanos e por IA é um problema semelhante: novos *LLMs* são otimizados para criar textos convincentes aos leitores ao ponto de serem imperceptíveis a agentes humanos (IPPOLITO *et al.*, 2019). Desta forma, os textos gerados a cada nova geração de *LLMs* apresentam distribuições estatísticas diferentes das gerações anteriores, fenômeno semelhante ao efeito da mudança de conceito. Portanto, é necessário que os classificadores sejam construídos através de algoritmos que possibilitem o paradigma de treinamento incremental para evitar a abordagem ingênua de retreino, diminuindo as chances de ocorrência de um efeito conhecido como esquecimento catastrófico (em inglês, *cathasrophic-forgetiness*) (BIESIALSKA; BIESIALSKA; COSTA-JUSSÀ , 2020).

4 METODOLOGIA

Neste capítulo, são detalhados os materiais e métodos empregados para validar a hipótese levantada neste projeto. A Figura 1 sumariza o protocolo experimental proposto e executado. Na sequência, é descrita a base de dados e os métodos utilizados no tratamento dos documentos. Em seguida, é apresentada a modelagem adotada e os modelos treinados para classificação automática. Por fim, é descrito o protocolo experimental e critérios adotados para comparar os resultados.

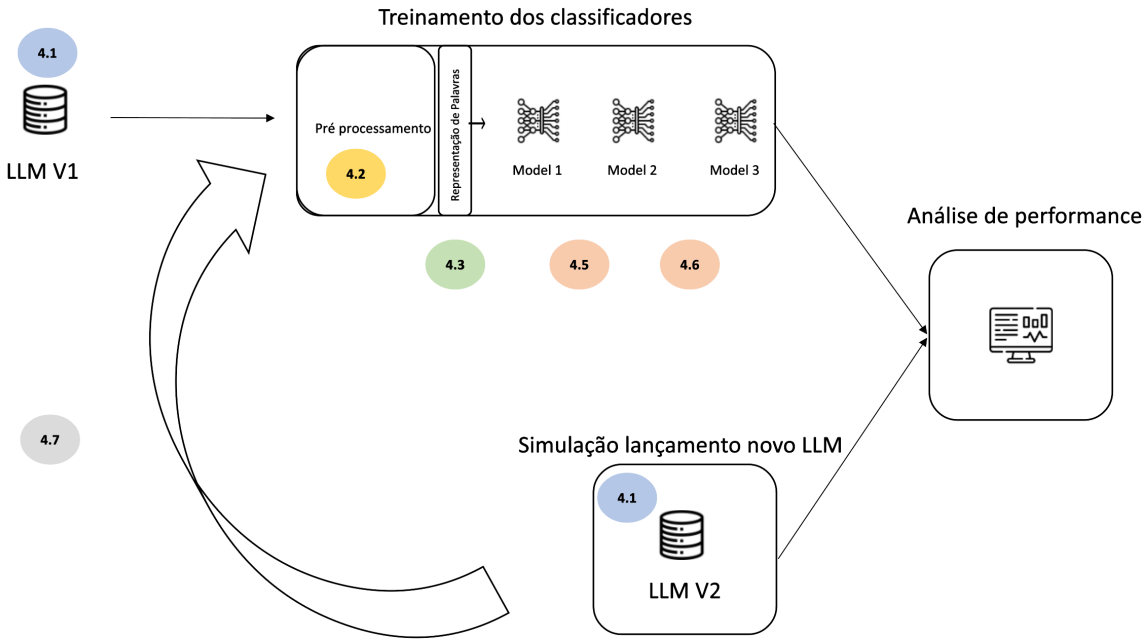


Figura 1 – Esquema genérico do protocolo experimental realizado. Os números destacados em balões coloridos representam as seções deste capítulo onde a respectiva etapa é explicada mais detalhadamente.

Fonte: Elaborado pelo autor.

4.1 *Base de dados*

Neste trabalho, foi utilizada a base de dados disponibilizada pela *OpenAI* para intuitos de pesquisas (SOLAIMAN *et al.*, 2019). Ela é constituída de duas partes. Na primeira, são disponibilizadas 250.000 amostras do conjunto textual *WebText*, introduzido por Radford *et al.* (2019), que originalmente contém 8 milhões de registros. O *WebText* é um *corpus* interno da *OpenAI* criado por meio de raspagem de páginas da web com ênfase na qualidade dos documentos. Os autores capturaram todos os *links* de resposta do

Reddit que receberam pelo menos 3 *karma*[1], utilizando isto como indicador heurístico para identificarem se os usuários percebem a postagem como relevante. O subconjunto derivado do *WebText* disponibilizado possui 250.000 amostras de treinamento, 5.000 amostras de validação e outras 5.000 amostras de teste. Na segunda parte, foram disponibilizados textos gerados sinteticamente por variações do modelo *GPT-2*. Foram apresentadas duas versões para cada quantidade de parâmetros: *k40* que faz referência a limitação do espaço amostral, ou amostragem de novos *tokens* mais prováveis. Neste caso, *top-k40* significa que os próximos *tokens* da sequência foram escolhidos a partir de uma lista com os 40 mais prováveis. Além desta, também foram empregadas as versões sem truncamento de espaço amostral, caracterizada pela **ausência** do *k40* em sua nomenclatura. Uma descrição resumida é fornecida na Tabela 1.

Tabela 1 – Descrição da quantidade de parâmetros e respectivo método de amostragem das versões dos modelos GPT-2 constituintes da base da dados. Na primeira coluna é exibida a quantidade de parâmetros na escala de milhões e o método de amostragem. As demais colunas mostram a quantidade de registros em cada uma das partições.

Versões disponíveis	Treino	Validação	Teste
117m-k40	250k	5k	5k
117m	250k	5k	5k
345m-k40	250k	5k	5k
345m	250k	5k	5k
762m-k40	250k	5k	5k
762m	250k	5k	5k
1542m-k40	250k	5k	5k
1542m	250k	5k	5k

Fonte: Elaborado pelo autor.

A junção destas duas partes (i.e, *WebText* e os dados sintéticos gerados pelas diferentes versões do modelo *GPT-2*) dá origem a base de dados denominada *gpt-2-output-dateset*. Os conjuntos de validação e teste são disponibilizados separadamente.

Neste trabalho, foram utilizados os registros gerados com amostragem *top-k40*, pois segundo o estudo de Ippolito *et al.* (2019), quando se limita o espaço amostral os *LLMs* têm mais chances de ludibriar agentes humanos e, portanto, é provável que os modelos disponíveis atualmente no mercado possuam algum truncamento do espaço amostral.

Foi utilizado o seguinte cenário de simulação: os modelos classificadores foram treinados com 20.000 amostras da base de dados *gpt-2-output-dataset*, sendo 10.000 do *WebText* e 10.000 do *GPT-2-117M-top-k40*. Foram escolhidos os primeiros 10.000 registros para ambos os casos por motivos de limitação computacional. Para o processo

[1] *Karma* reflete o quanto as contribuições de um usuário impactam a comunidade. Quando os *posts* ou comentários recebem *upvotes*, o autor ganha *karma*

de refinamento dos modelos, foi utilizado um conjunto de dados de validação com 10.000 amostras, sendo 5.000 textos extraídos do *WebText* e 5.000 gerados pelo *GPT-2-117M-top-k40*. Para teste, foi utilizado um conjunto de dados seguindo o mesmo padrão do conjunto de validação. Ambos os conjuntos de dados possuem textos distintos. No cenário de simulação de lançamento de um novo *LLM*, foi utilizado um conjunto de dados de 5.000 amostras do *GPT-2-1542m-top-k40* e 5.000 amostras do *Webtext*, totalizando 10.000 amostras disponíveis para realizar o ajuste dos parâmetros do modelo. As análises de desempenho no segundo cenário também foram realizadas utilizando um conjunto distinto do conjunto de atualização de parâmetros, com 10.000 amostras.

Tabela 2 – Bases de dados utilizadas nos experimentos. As colunas (Treino, Aprendizado incremental, Validação, Teste) apresentam as quantidades de amostras utilizadas em cada um dos processos. Na coluna Etapa, é descrito em que momento os dados foram utilizados: treinamento dos classificadores ou na simulação de um novo LLM.

Modelo	Treino	Aprendizado Incremental	Validação	Teste	Etapa
117M-k40	10k	-	5k	5k	Treinamento
1542M-k40	-	5k	-	5k	Simulação

Fonte: Elaborado pelo autor.

4.2 Pré-processamento do texto

O pré-processamento consistiu na limpeza dos dados, excluindo caracteres especiais e focando apenas em caracteres alfa-numéricos. Também, foram utilizadas técnicas de remoção de palavras de alta frequência que carregam pouco significado, através da remoção de *stop-words* utilizando como referência as palavras da biblioteca *nltk*. A normalização dos documentos utilizou a técnica de lematização, aplicada através do método *lemma* da biblioteca *scipy*.

4.3 Modelos de classificação

Foram escolhidos métodos de classificação disponíveis no *scikit-learn* com capacidade de aprendizado incremental para avaliação dos resultados. São eles:

- *Gaussian Naive Bayes*: do pacote *sklearn.naive-bayes*.
- *SDG Classifier*: do pacote *sklearn.linear-model*.
- *Passive Aggressive*: do pacote *sklearn.linear-model*.
- *Perceptron*: do pacote *sklearn.linear-model*.

4.3.1 Modelo *baseline*

Como *baseline*, é utilizado um modelo de regressão logística do pacote *sklearn.linear-model*. Este modelo visa manter a consistência teórica dos estudos realizados que utilizaram a mesma base de dados empregada neste projeto. Nestes, foram treinados classificadores baseados no modelo *RoBERTa* e, como *baseline*, foi empregado um modelo de regressão logística.

4.3.2 Estado-da-arte

Foi utilizado uma versão do modelo *RoBERTa*, especificamente o *distilroberta-base* da biblioteca *transformers*. Este classificador foi otimizado para classificar textos do *gpt-2-117m-top-k40*. Posteriormente, o classificador foi apresentado aos dados do cenário de simulação para ser comparado com o desempenho dos outros classificadores com capacidade de aprendizado incremental.

4.4 Representação computacional

Primeiramente, foi construído um *pipeline* considerando a representação de palavras utilizando *TF-IDF*. Em seguida, utilizando modelos pré-treinados de *word2vec* e, finalmente, um modelo pré-treinado *GloVe*, conforme detalhado a seguir.

4.4.1 Representação baseada em frequência - *TF-IDF*

A técnica de contagem de termos *TF-IDF* (ver Seção 2.3.2) cria uma representação de palavra fazendo uma distinção entre termos comumente utilizados no *corpus* e termos mais raros. Esta técnica pode ser ajustada para melhorar a representação textual variando o parâmetro de *n-gram*. Desta forma, as palavras deixam de ser contabilizadas unicamente e passam a ser contabilizadas de acordo com o parâmetro do *n-gram*. Este trabalho utiliza $n = 2$, ou seja, bigramas, e ignorou termos que aparecem em menos de 5 documentos através do parâmetro *min_df = 5*. Além disso, a quantidade máxima de *features* foi estabelecida em *max_features*= 2^{21}. Estes valores foram os mesmos empregados no estudo realizado por Solaiman *et al.* (2019).

4.4.2 Word2Vec

Word2Vec é um método de representação de palavras utilizado para construir um modelo de linguagem. O modelo pode ser criado utilizando uma implementação da biblioteca *gensim2*. Este trabalho utiliza um modelo pré-treinado disponibilizado pelo *Google*, denominado *GoogleNews-vectors-negative300*.

4.4.3 Glove

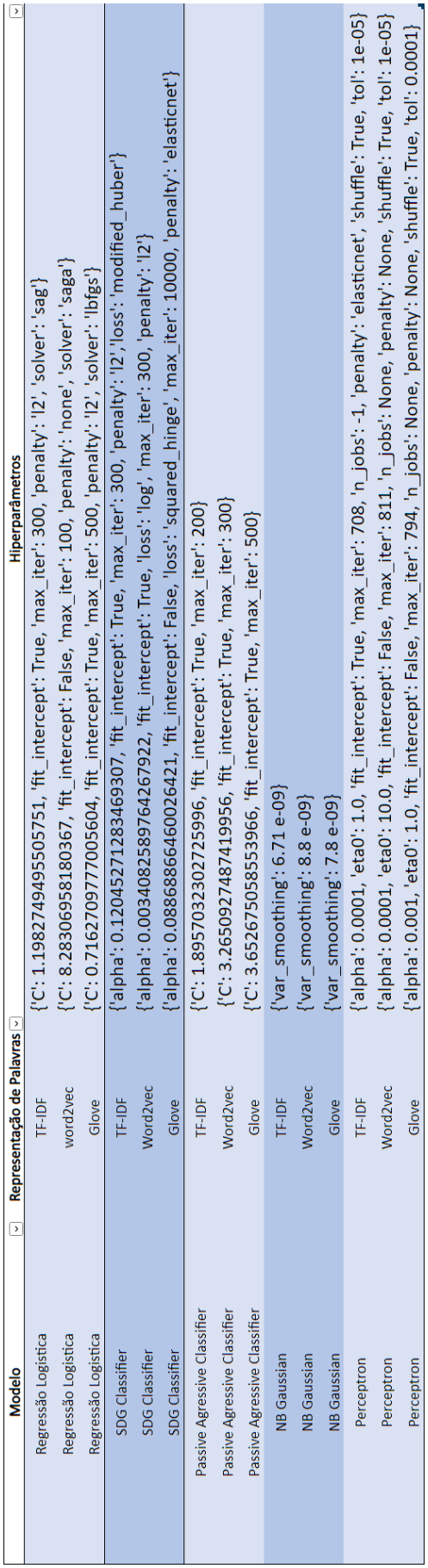
Foi utilizado o modelo pré-treinado com os dados textuais do *corpora* de treinamento *Wikipedia 2014* e *Gigaword 5*, denominado *glove.6B.50d*. Este modelo foi escolhido por ter uma grande variedade de palavras individuais (6B de *tokens*) e, consequentemente, um vocabulário mais vasto, pois o problema abordado por este trabalho não se limita à sentenças curtas (KOWSARI *et al.*, 2019).

4.5 Otimização dos modelos

Para os modelos com aprendizado incremental e o *baseline*, foi utilizado o buscador de espaço de parâmetros *RandomSearch*, disponível na biblioteca *scikit-learn*. Para as diferentes técnicas de representação de palavras, foram obtidos os valores de hiperparâmetros disponibilizados na Figura 2.

4.6 Aprendizado incremental

Durante a etapa de aprendizado incremental, foram empregadas 10.000 amostras, sendo 5.000 exemplos de textos sintéticos do modelo *GPT-2-1542m-top-k40* e 5.000 exemplos de textos escritos por agentes humanos, extraídos do *WebText*. O aprendizado incremental foi preparado de forma que, em cada época de treinamento são utilizadas 500 amostras, totalizando 20 épocas para todo o processo de atualização de parâmetros (Figura 3).



Modelo	Representação de Palavras	Hiperparâmetros
Regressão Logistica	TF-IDF	{'C': 1.1982749495505751, 'fit_intercept': True, 'max_iter': 300, 'penalty': 'l2', 'solver': 'sag'}
Regressão Logistica	word2vec	{'C': 8.283069581803 67, 'fit_intercept': False, 'max_iter': 100, 'penalty': 'none', 'solver': 'saga'}
Regressão Logistica	Glove	{'C': 0.7162709777005604, 'fit_intercept': True, 'max_iter': 500, 'penalty': 'l2', 'solver': 'lbfgs'}
SDG Classifier	TF-IDF	{'alpha': 0.12045271283469307, 'fit_intercept': True, 'max_iter': 300, 'penalty': 'l2','loss': 'modified_huber'}
SDG Classifier	Word2vec	{'alpha': 0.0034082589764267922, 'fit_intercept': True, 'loss': 'log', 'max_iter': 300, 'penalty': 'l2'}
SDG Classifier	Glove	{'alpha': 0.08868866460026421, 'fit_intercept': False, 'loss': 'squared_hinge', 'max_iter': 10000, 'penalty': 'elasticnet'}
Passive Agressive Classifier	TF-IDF	{'C': 1.895703230272 5996, 'fit_intercept': True, 'max_iter': 200}
Passive Agressive Classifier	Word2vec	{'C': 3.265092748741 9956, 'fit_intercept': True, 'max_iter': 300}
Passive Agressive Classifier	Glove	{'C': 3.652675058553966, 'fit_intercept': True, 'max_iter': 500}
NB Gaussian	TF-IDF	{'var_smoothing': 6.71 e-09}
NB Gaussian	Word2vec	{'var_smoothing': 8.8 e-09}
NB Gaussian	Glove	{'var_smoothing': 7.8 e-09}
Perceptron	TF-IDF	{'alpha': 0.0001, 'eta0': 1.0, 'fit_intercept': True, 'max_iter': 708, 'n_jobs': -1, 'penalty': 'elasticnet', 'shuffle': True, 'tol': 1e-05}
Perceptron	Word2vec	{'alpha': 0.0001, 'eta0': 10.0, 'fit_intercept': False, 'max_iter': 811, 'n_jobs': None, 'penalty': None, 'shuffle': True, 'tol': 1e-05}
Perceptron	Glove	{'alpha': 0.001, 'eta0': 1.0, 'fit_intercept': False, 'max_iter': 794, 'n_jobs': None, 'penalty': None, 'shuffle': True, 'tol': 0.0001}

Figura 2 – Valores dos hiperparâmetros dos modelos criados para as diferentes metodologias de representação de palavras

Fonte: Elaborado pelo autor.

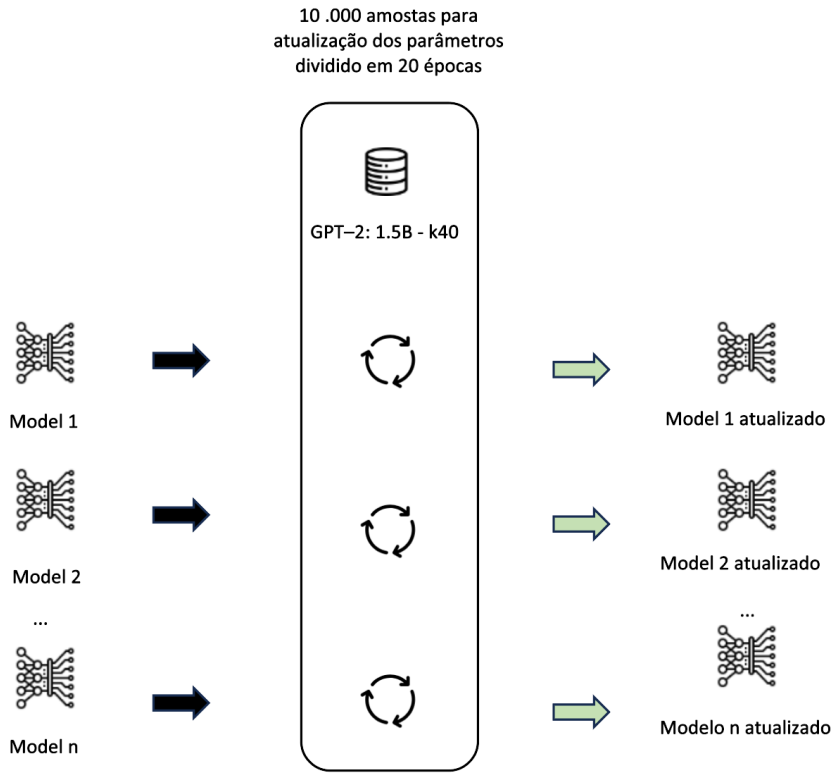


Figura 3 – São utilizadas 10.000 amostras para a atualização dos modelos para o cenário simulado. São utilizadas 20 épocas de atualização com 500 amostras em cada uma.

Fonte: Elaborado pelo autor.

5 RESULTADOS

Para todos os resultados reportados, os modelos foram treinados utilizando a base de dados *117m-k-40-train* e avaliados na base *117m-k40-test*, com excessão da simulação do cenário de lançamento de um *LLM* maior, em que os modelos foram avaliados sobre o *dataset 1542m-k40-test*. A medida de análise de desempenho adotada é a *Medida F1*.

5.1 Offline learning

Na Tabela 3, são apresentados os resultados dos modelos treinados utilizando diferentes técnicas de representação de palavras. É possível destacar que, para a representação *TF-IDF*, os modelos *baseline* e o *Passive Aggressive* obtiveram desempenhos de classificação bastante próximos, 0,83 e 0,85, respectivamente, sendo que o *Passive Aggressive* apresentou um desempenho ligeiramente superior, atingindo inclusive o desempenho do classificador baseado na arquitetura *RoBERTa*. Considerando a representação *Word2Vec*, *SGD Classifier* com a função de perda *modified-huber* apresentou um resultado ligeiramente inferior ao *baseline* e, consequentemente, ao modelo *RoBERTa*. Para a representação de palavras utilizando *Glove*, todos os modelos alcançaram um desempenho significativamente inferior ao *baseline*, com destaque para o *Gaussian Naive Bayes* (referido como *Gaussian* NB), que apresentou o melhor resultado dentre os modelos com esta representação de palavras. O modelo *Perceptron* não se destacou em nenhuma das abordagens utilizadas de representação de palavras, ficando foram do quadro de destaques.

Tabela 3 – Modelos com diferentes representação de palavras, com os melhores resultados de cada grupo destacado em negrito

Método de representação	Modelo de classificação	Medida-F_1
Autoencoder	RoBERTa	**0,85**
TF-IDF	Regressão Logística *(baseline)*	0,83
	Passive Aggressive	**0,85**
Word2vec	Regressão Logística *(baseline)*	**0,84**
	SGD	0,83
Glove	Regressão Logística *(baseline)*	**0,83**
	Gaussian NB	0,78

Fonte: Elaborado pelo autor.

5.2 Cenário de lançamento de um *LLM* maior

Os resultados da Tabela 4 foram obtidos através do pradigma de *off-line learning*, que na prática significa que foi apresentado um *dataset* para treinamento dos modelos,

um *dataset* de validação para o processo de escolha dos melhores hiperparâmetros e um *dataset* de teste para avaliar a performance final do modelo. Entretanto, ao introduzir o conjunto de dados *gpt-2-1542M-k40-test*, que, dentro do contexto deste trabalho, simula o cenário do lançamento de um modelo *LLM* maior e mais complexo, há uma degradação significativa no desempenho de todos os modelos de classificação conforme demonstrado na Tabela 4. Observa-se uma queda média de 12,5% na Medida-*F1* quando os modelos foram apresentados ao novo conjunto de dados, considerando apenas os modelos não neurais. Considerando o classificador *RoBERTa*, nota-se uma piora na performance, porém abaixo da média dos demais modelos, 11,76%. Apesar dos modelos *Passive Aggressive* e *RoBERTa* terem apresentado o mesmo desempenho no cenário anterior (Tabela 3), existe uma diferença significativa na perda de desempenho neste cenário, sendo uma queda de 14,12% e 11,76%, respectivamente. Isto indica que o modelo *RoBERTa* apresentou uma melhor capacidade de generalização no segundo cenário.

Tabela 4 – Simulação do cenário de lançamento de um LLM maior e mais complexo

Método de representação	Modelo de classificação	Medida-F_1	Resultado anterior	**Variação (%)**
autoencoder	RoBERTa	0,75	0,85	-11,76
TF-IDF	Regressão Logística	0,72	0,83	-13,25
	Passive Aggressive	0,73	0,85	-14,12
Word2Vec	Regressão Logística	0,74	0,84	-11,90
	SGD	0,73	0,83	-12,05
Glove	Regressão Logística	0,73	0,83	-12,05
	Gaussian NB	0,69	0,78	-11,54

Fonte: Elaborado pelo autor.

As Figura de 4 a 6 apresentam as matrizes de confusão de cada modelo. As imagens à esquerda, identificadas como os itens A, C e E, são as as matrizes de confusão para o conjunto de teste *gpt-2-117m-top-k40-test* e, as imagens B, D e F, as matrizes de confusão considerando o cenário simulação de lançamento de um novo *LLM*, (*gpt-2-1542M-k40-test*). Os pares de matrizes de confusão A - B (*Passive Aggressive - TF-ID*), C - D (*SGD Classifier - Word2vec*) e E - F (*Gaussian - Glove*) demonstram um comportamento de dificuldade em discriminar os textos gerados pelo modelo *gpt-2-1542M-k40*. Isto fica evidenciado pelo deslocamento de verdadeiros-positivos (quadrante inferior direito) para falsos-negativos (quadrande inferior-esquerdo), refletindo diretamente na métrica de *recall*, com uma queda média de 26,3%.

As Tabelas 5 a 7 mostram as medidas de acurácia, precisão e *recall* das figuras com respectiva referência.

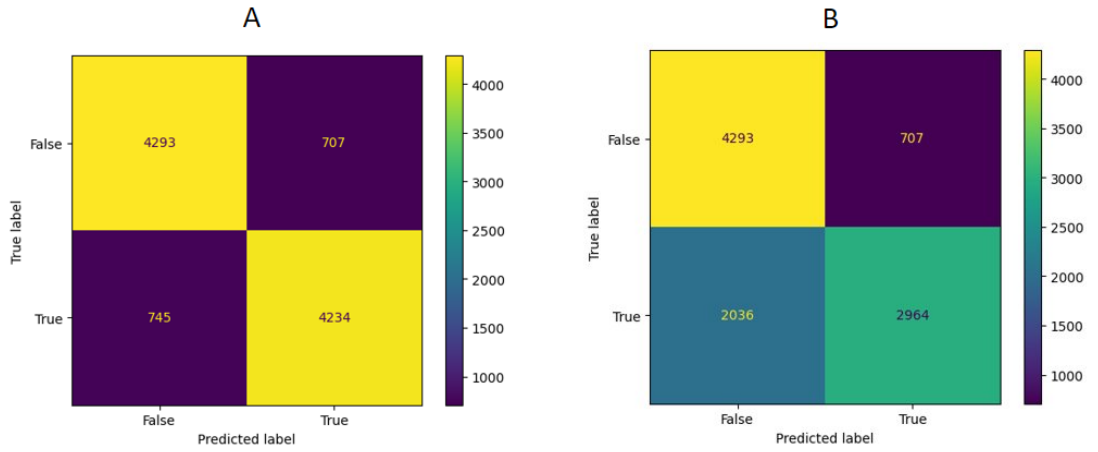


Figura 4 – Matrizes de confusão do modelo *Passive Aggressive* utilizando *TF-IDF* como método de representação de palavras. Em **A**, a matriz refere-se aos dados de teste do modelo gerador *gpt-2-117m-top-k40-test* e, em **B**, utilizando *gpt-2-1542M-k40-test*.

Fonte: Elaborado pelo autor.

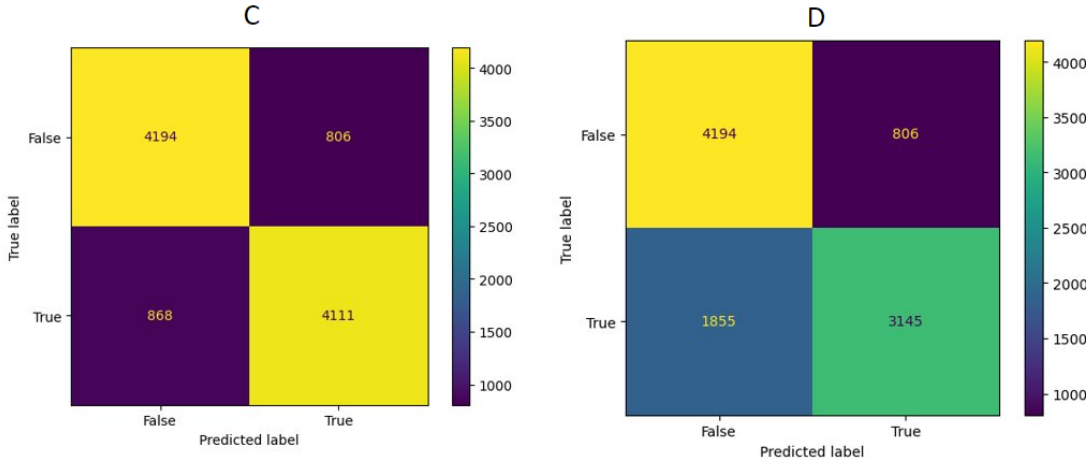


Figura 5 – Matrizes de confusão do modelo *SGD* utilizando *Word2vec* como método de representação de palavras. Em **C**, a matriz refere-se aos dados de teste do modelo gerador *gpt-2-117m-top-k40-test* e, em **D**, utilizando *gpt-2-1542M-k40-test*.

Fonte: Elaborado pelo autor.

5.3 Online learning

Na tabela 8, apresenta o desempenho dos modelos de classificação após o processo de ajuste. Na primeira linha, é apresentado o resultado do modelo *Passive Aggressive* com representação *TF-ID*. Houve um incremento da Medida-F1 de aproximadamente 2,75%, partindo de 0,73 e atingindo 0,75. Em seguida, o classificador *SGD* com representação de palavras *Word2Vec* atinge um valor máximo ao final de 0,74, um aumento de 1,3% em comparação com o valor inicial de 0,73. O classificador *Gaussian Naive Bayes* não

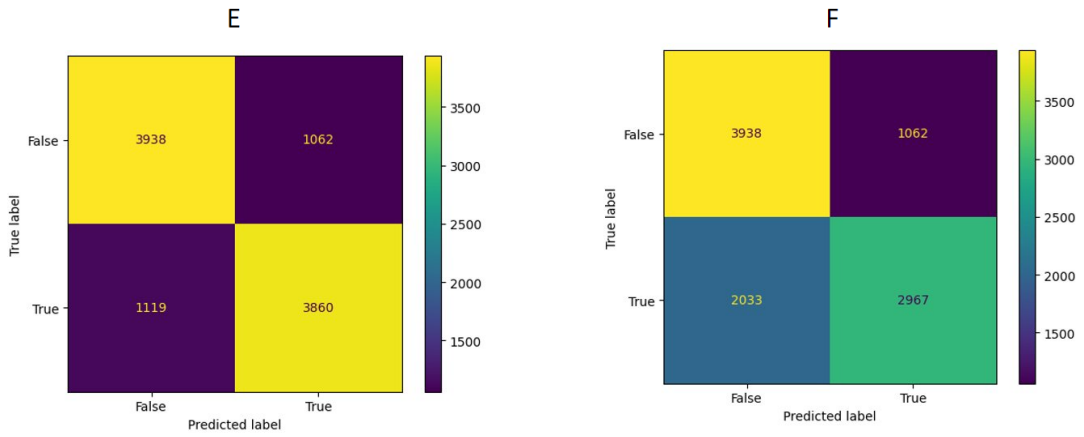


Figura 6 – Matrizes de confusão do modelo *Gaussian Naive Bayes* utilizando *Glove* como Método de representação de palavras. Em **E**, a matriz refere-se aos dados de teste do modelo gerador *gpt-2-117m-top-k40-test* e, em **F**, utilizando *gpt-2-1542M-k40-test*.

Fonte: Elaborado pelo autor.

Tabela 5 – Desempenho obtido pelo método *Passive Aggressive*

Cenário	Base de dados	Acurácia	Precisão	*Recall*
A	gpt-2-117m-top-k40-test	0,83	0,84	0,83
B	gpt-2-1542M-k40-test	0,73	0,80	0,63
Variação (%)	-	-12,05	-4,76	-24,10

Fonte: Elaborado pelo autor.

Tabela 6 – Desempenho obtido pelo método *SGD*

Cenário	Base de dados	Acurácia	Precisão	*Recall*
C	gpt-2-117m-top-k40-test	0,78	0,78	0,78
D	gpt-2-1542M-k40-test	0,69	0,74	0,59
Variação (%)	-	-11,54	-5,13	-24,36

Fonte: Elaborado pelo autor.

apresentou melhora utilizando a técnica de representação *GloVe*.

A seguir, é ilustrado o processo de aprendizado incremental dos modelos. Na Figura 7, referente ao modelo *Passive Aggressive* (PA), é possível observar um aumento não linear da Medida-F1 conforme o processo de ajuste. Nota-se que em alguns momentos há uma queda acentuada da Medida-F1, em especial nas regiões de 3.000, 4.000 e 8.000 amostras. Na Figura 8, referente ao modelo *SGD*, é apresentado um comportamento de aumento linear da Medida-F1, porém com pouca variação entre as épocas. Por fim, o modelo *Gaussian NB*, representado na Figura 9, não apresenta melhoras significativas.

Tabela 7 – Desempenho obtido pelo método *Gaussian NB*

Cenário	Base de dados	Acurácia	Precisão	*Recall*
E	gpt-2-117m-top-k40-test	0,85	0,86	0,85
F	gpt-2-1542M-k40-test	0,73	0,81	0,59
Variação (%)	-	-14,12	-5,81	-30,59

Fonte: Elaborado pelo autor.

Tabela 8 – Modelos de classificação após serem atualizados *online*.

Modelo de representação	Modelo de classificação	Medida F_1
TF-IDF	Passive Aggressive	0,75
Word2Vec	SGD	0,74
Glove	Gaussian NB	0,69

Fonte: Elaborado pelo autor.

5.3.1 *Modelo RoBERTa com retreinamento*

Na última etapa, foi realizado o retreino completo do modelo *RoBERTa* com o intuito de analisar o impacto do lançamento de novos *LLMs* no método considerado o estado-da-arte. Nesta etapa, foram utilizadas as amostras de treinamento do modelo *gpt-2-117m-top-k40-treino* e *gpt-2-1542M-top-k40-treino*, que foram utilizadas durante o processo de aprendizagem incremental, totalizando 30.000 amostras, sendo 20.000 do primeiro e 10.000 do segundo. O desempenho foi aferido utilizando as 10.000 amostras do modelo *gpt-2-1542M-top-k40-teste*, considerando apenas o segundo cenário. Resgatando os dados apresentados na Tabela 9 e acrescentado os resultado do **retreinamento** do modelo *RoBERTa*, é possível observar que esse modelo apresenta uma recuperação de desempenho superior aos demais. Apesar desta diferença, o retreino do modelo *RoBERTa* exige uma capacidade computacional maior, sendo necessário uma *GPU* para realização deste processo em tempo hábil.

Tabela 9 – Classificadores após serem atualizados

Modelo de representação	Modelo de classificação	Medida-F_1 pré	Medida-F_1 pós	**Variação (%)**
autoencoder	**RoBERTa**	0,75	**0,78**	4,0
TF-IDF	Passive Aggressive	0,73	0,75	2,7
Word2Vec	SGD	0,73	0,74	1,4
Glove	Gaussian NB	0,69	0,69	0,0

Fonte: Elaborado pelo autor.

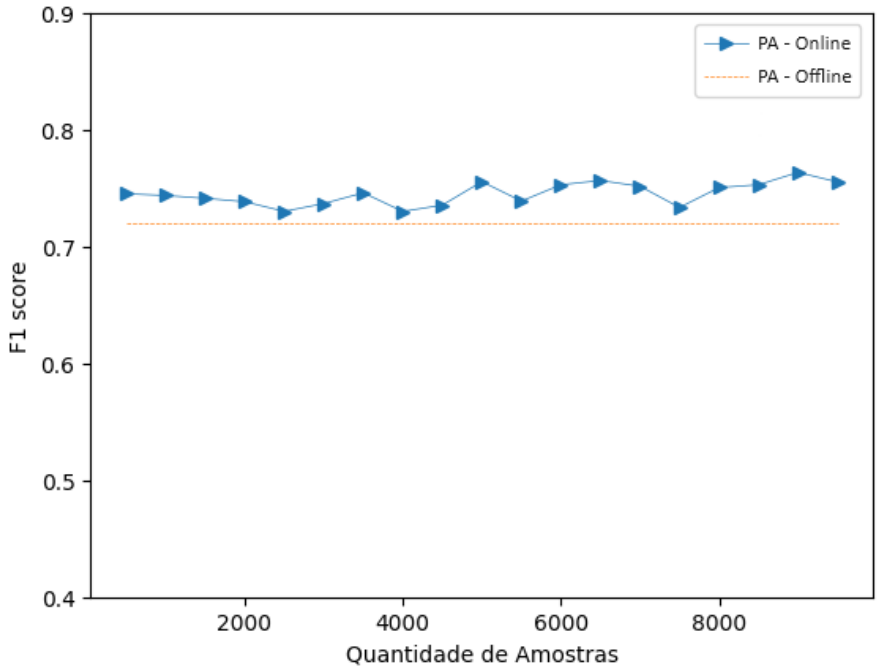


Figura 7 – Modelo *Passive Aggressive* (PA) com representação *TF-IDF* durante o processo de aprendizado incremental. No eixo X, é possível observar o acumulado de amostras utilizadas para a atualização do modelo. A linha tracejada laranja mostra o desempenho antes do aprendizado *online*. Em azul, o desempenho do modelo conforme as épocas de treinamento avançam.

Fonte: Elaborado pelo autor.

5.3.2 *Backtest*

A Tabela 10 apresenta os resultados dos classificadores que passaram pelo processo de atualização. Na última coluna, observa-se os resultados dos classificadores aferidos utilizando o mesmo *dataset* de treino utilizado no processo de aprendizagem *offline*. Neste cenário, é possível observar uma perda de performance significativa do *Passive Aggressive* (Medida-F1 = 0,85) no paradigma *offline* para Medida-F1 = 0,8 no cenário *online*. O modelo *SGD* apresentou uma queda de performance menos significativa de 0,83 para 0,82, respectivamente, e o classificador *Gaussian NB* manteve o mesmo resultado, indicando que o modelo não foi atualizado.

Tabela 10 – Performance no data set *117m-k40-teste* antes (*offline*) e depois da atualização dos parâmetros *online*

Modelo de representação	Modelo de classificação	Medida F_1 *offline*	Medida F_1 *online*	**Variação (%)**
TF-IDF	Passive Aggressive	0,85	0,80	-5,9
Word2vec	SGD	0,83	0,82	-1,2
Glove	Gaussian NB	0,78	0,78	0,0

Fonte: Elaborado pelo autor.

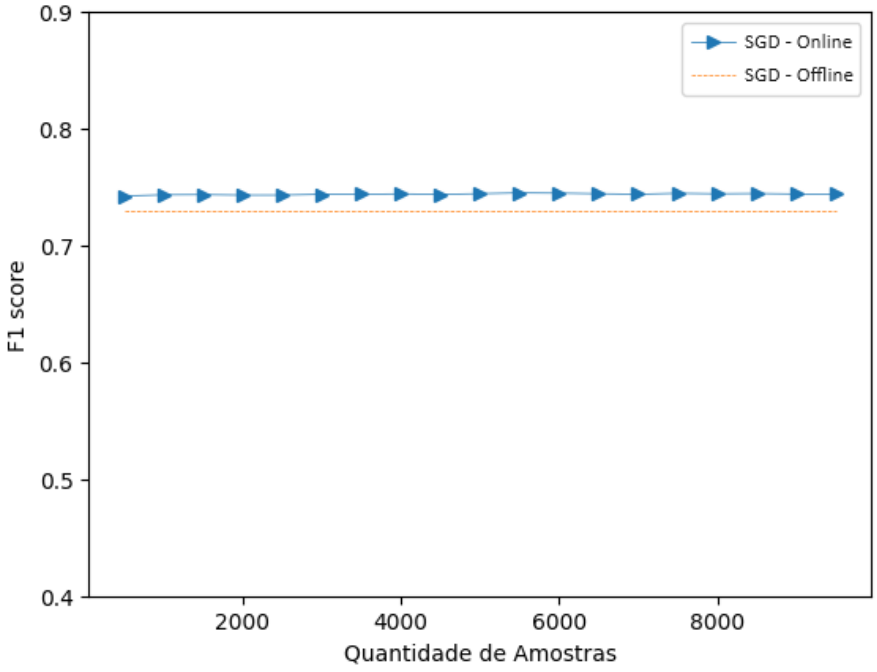


Figura 8 – Modelo *SGD* com representação *word2vec* durante o processo de aprendizado incremental. No eixo X, é possível observar o acumulado da quantidade de amostras utilizadas para a atualização do modelo. A linha tracejada laranja mostra o desempenho antes do aprendizado *online*. Em azul, o desempenho do modelo conforme as épocas de treinamento avançam.

Fonte: Elaborado pelo autor.

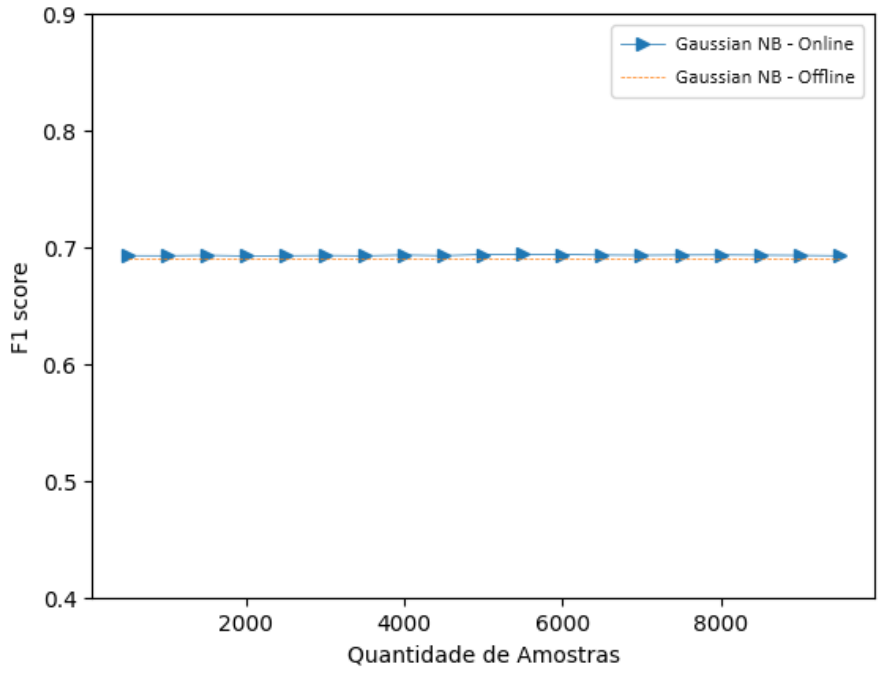


Figura 9 – Modelo *Gaussian classifer* com representação *Glove* durante o processo de aprendizado incremental. No eixo X, é possível observar o acumulado de amostras utilizadas para a atualização do modelo. A linha tracejada laranja mostra o desempenho antes do aprendizado *online*. Em azul, o desempenho do modelo conforme as épocas de treinamento avançam.

Fonte: Elaborado pelo autor.

6 CONCLUSÃO

A geração de linguagem natural é um campo em constante evolução, com aplicações significativas em diversos cenários, como tradutores automáticos, classificação de tópicos, análise de sentimentos textuais e *chatbots*. Os grandes modelos de linguagem (LLMs) têm o potencial de facilitar as atividades cotidianas, porém, eles, também, apresentam riscos associados ao uso mal intencionado. A detecção de textos sintéticos gerados por estes modelos é um desafio constante, pois se tornam mais sofisticados a cada geração e com textos cada vez mais convincentes. Isso faz com que os modelos de detecção de texto atuais, mesmos os neurais, considerados estado-da-arte, rapidamente se tornem obsoletos no relativo curto período entre o lançamento de novas gerações, pois grande parte dos estudos que propõem estes tipos de classificadores considera o paradigma de aprendizagem *offline* em seus treinamentos, sem que haja atualizações em seus parâmetros em meia vida.

Este trabalho baseou-se na hipótese de que os modelos treinados via o paradigma de aprendizado *offline* são eficazes na discriminação de textos sintéticos gerados por LLMs atuais, mas a mudança para uma nova geração pode comprometer significativamente seus desempenhos, demandando a execução do treinamento novamente. Este trabalho também considera a hipótese de que é possível treinar classificadores de detecção automática de textos com abordagem clássica com aprendizado incremental, visando obter classificadores que se adaptem rapidamente aos lançamentos mais recentes de LLMs.

Durante o experimento, observou-se que o desempenho dos classificadores treinados de forma *offline* tende a ser otimista se considerado apenas a geração de LLMs aos quais tiveram acesso aos dados de treinamento. Isto fica evidente quando são apresentados ao conjunto de dados do cenário transitório, em que foi observado uma degradação significativa de desempenho devido, principalmente, à sequência semântica mais refinada do modelo gerador de nova geração, evidenciando o efeito do *concept-drifting*.

No entanto, durante os experimentos de treinamento *online*, a atualização dos classificadores via o aprendizado incremental, recuperou parte do desempenho, fazendo com que os classificadores tradicionais se equiparassem ao desempenho do modelo RoBERTa no conjunto do cenário de transição. Há, entretanto, uma ligeira perda de desempenho considerando a etapa de *backtests*, em que os classificadores foram submetidos, novamente, ao conjunto de dados do LLM menor.

Este trabalho não aprofundou-se no modelo *RoBERTa*, mas sim nos classificadores com aprendizado incremental. Consequentemente, trabalhos futuros podem dar ênfase maior em modelos neurais, propondo diferentes modelos pré-treinados, uma etapa de refinamento mais robusta e também uma maior quantidade de dados. Devido as limitações

computacionais, foi utilizada uma quantidade de dados limitada e isso pode não ter levado ao modelo *RoBERTa*, aqui treinado, ao seu máximo desempenho.

REFERÊNCIAS

BAHDANAU, D.; CHO, K.; BENGIO, Y. Neural machine translation by jointly learning to align and translate. **arXiv preprint arXiv:1409.0473**, 2014.

BIESIALSKA, M.; BIESIALSKA, K.; COSTA-JUSSÀ , M. R. Continual lifelong learning in natural language processing: A survey. *In*: **Proceedings of the 28th International Conference on Computational Linguistics**. International Committee on Computational Linguistics, 2020. Available at: <https://doi.org/10.18653%2Fv1%2F2020.coling-main.574>.

BROWN, T. *et al.* Language models are few-shot learners. **Advances in neural information processing systems**, v. 33, p. 1877–1901, 2020.

CELIKYILMAZ, A.; CLARK, E.; GAO, J. Evaluation of text generation: A survey. **arXiv preprint arXiv:2006.14799**, 2020.

CHO, K. *et al.* Learning phrase representations using RNN encoder–decoder for statistical machine translation. *In*: **Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)**. Doha, Qatar: Association for Computational Linguistics, 2014. p. 1724–1734. Available at: <https://aclanthology.org/D14-1179>.

CRAMMER, K. *et al.* Online passive aggressive algorithms. **Journal of Machine Learning Research 7 (2006) 551–585**, 2006.

DEVLIN, J. *et al.* Bert: Pre-training of deep bidirectional transformers for language understanding. **arXiv preprint arXiv:1810.04805**, 2018.

GALLANT, S. I. *et al.* Perceptron-based learning algorithms. **IEEE Transactions on neural networks**, v. 1, n. 2, p. 179–191, 1990.

GRAVES, A. Generating sequences with recurrent neural networks. **arXiv preprint arXiv:1308.0850**, 2013.

IPPOLITO, D. *et al.* Human and automatic detection of generated text. **CoRR**, abs/1911.00650, 2019. Available at: <http://arxiv.org/abs/1911.00650>.

JAWAHAR, G.; ABDUL-MAGEED, M.; LAKSHMANAN, L. V. S. **Automatic Detection of Machine Generated Text: A Critical Survey**. 2020.

KOWSARI, K. *et al.* Text classification algorithms: A survey. **Information**, v. 10, n. 4, 2019. ISSN 2078-2489. Available at: <https://www.mdpi.com/2078-2489/10/4/150>.

LANE, H.; HOWARD, C.; HAPKE, H. **Natural Language Processing in Action**. [*S.l.: s.n.*]: Manning, 2019.

LIU, Y. *et al.* Roberta: A robustly optimized bert pretraining approach. **arXiv preprint arXiv:1907.11692**, 2019.

MIKOLOV, T. *et al.* Efficient estimation of word representations in vector space. **arXiv preprint arXiv:1301.3781**, 2013.

PENNINGTON, J.; SOCHER, R.; MANNING, C. GloVe: Global vectors for word representation. *In*: **Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)**. Doha, Qatar: Association for Computational Linguistics, 2014. p. 1532–1543. Available at: <https://aclanthology.org/D14-1162>.

RADFORD, A. *et al.* Improving language understanding by generative pre-training. OpenAI, 2018.

RADFORD, A. *et al.* Language models are unsupervised multitask learners. **OpenAI blog**, v. 1, n. 8, p. 9, 2019.

SEBASTIANI, F. Machine learning in automated text categorization. **ACM computing surveys (CSUR)**, ACM New York, NY, USA, v. 34, n. 1, p. 1–47, 2002.

SILVA, R.; ALMEIDA, T. How concept drift can impair the classification of fake news. *In*: **Anais do IX Symposium on Knowledge Discovery, Mining and Learning**. Porto Alegre, RS, Brasil: SBC, 2021. p. 121–128. ISSN 2763-8944. Available at: <https://sol.sbc.org.br/index.php/kdmile/article/view/17469>.

SKLEARN-SGDCLASSIFIER. 2023. Available at: <https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html>.

SOLAIMAN, I. *et al.* Release strategies and the social impacts of language models. **CoRR**, abs/1908.09203, 2019. Available at: <http://arxiv.org/abs/1908.09203>.

SUSMAGA, R. Confusion matrix visualization. *In*: SPRINGER. **Intelligent Information Processing and Web Mining: Proceedings of the International IIS: IIPWM '04 Conference held in Zakopane, Poland, May 17–20, 2004**. [*S.l.: s.n.*], 2004. p. 107–116.

SUTSKEVER, I.; VINYALS, O.; LE, Q. V. Sequence to sequence learning with neural networks. **Advances in neural information processing systems**, v. 27, 2014.

WEBB, G. I.; BOUGHTON, J. R.; WANG, Z. Not so naive bayes: aggregating one-dependence estimators. **Machine learning**, Springer, v. 58, p. 5–24, 2005.

ZHANG, T. Solving large scale linear prediction problems using stochastic gradient descent algorithms. *In*: **Proceedings of the twenty-first international conference on Machine learning**. [*S.l.: s.n.*], 2004. p. 116.