UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
FACULDADE DE CIÊNCIAS - CAMPUS BAURU
DEPARTAMENTO DE COMPUTAÇÃO
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

JOÃO OTÁVIO RODRIGUES FERREIRA FREDIANI

IDENTIFICAÇÃO DE AUTORIA EM TEXTOS CURTOS UTILIZANDO TÉCNICAS DE PROCESSAMENTO DE LINGUAGEM NATURAL

BAURU
Fevereiro/2022

JOÃO OTÁVIO RODRIGUES FERREIRA FREDIANI

IDENTIFICAÇÃO DE AUTORIA EM TEXTOS CURTOS UTILIZANDO TÉCNICAS DE PROCESSAMENTO DE LINGUAGEM NATURAL

Trabalho de Conclusão de Curso do Curso de Bacharelado em Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências, Campus Bauru.
Orientador: Prof. Associado Aparecido Nilceu Marana

BAURU
Fevereiro/2022

João Otávio Rodrigues Ferreira Frediani Identificação de autoria em textos curtos utilizando técnicas de processamento de linguagem natural/ João Otávio Rodrigues Ferreira Frediani. – Bauru, Fevereiro/2022- 50 p. : il. (algumas color.) ; 30 cm.
Orientador: Prof. Associado Aparecido Nilceu Marana
Trabalho de Conclusão de Curso – Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Bacharelado em Ciência da Computação, Fevereiro/2022.
1. Inteligência Artificial 2. Processamento de Linguagem Natural 3. Atribuição de Autoria 4. BERT.

João Otávio Rodrigues Ferreira Frediani

Identificação de autoria em textos curtos utilizando técnicas de processamento de linguagem natural

Trabalho de Conclusão de Curso do Curso de Bacharelado em Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências, Campus Bauru.

Banca Examinadora

Prof. Associado Aparecido Nilceu Marana
Orientador
Departamento de Computação
Faculdade de Ciências
Universidade Estadual Paulista "Júlio de Mesquita Filho"

Profª. Drª. Simone das Graças Domingues Prado
Departamento de Computação
Faculdade de Ciências
Universidade Estadual Paulista "Júlio de Mesquita Filho"

Profª. Drª. Andréa Carla Gonçalves Vianna
Departamento de Computação
Faculdade de Ciências
Universidade Estadual Paulista "Júlio de Mesquita Filho"

Bauru, 04 de março de 2022.

Dedico este trabalho à todos os professores que fazem de suas vidas a educação e principalmente aos que participaram da minha.

Agradecimentos

Agradeço a minha família, meus pais e meus irmãos, que desde sempre incentivaram minha educação e curiosidade e que foram meu apoio de todas as maneiras possíveis durante a graduação.

Agradeço a meus amigos, os que me acompanham desde sempre e os que fiz durante a graduação, que estiveram ao meu lado e tornaram minha vida mais feliz.

Agradeço finalmente a meus professores, em especial meus orientadores Prof. Associado Aparecido Nilceu Marana e Profa. Dra. Tatiana Miguel Rodrigues, que não só me forneceram todo o conhecimento necessário como tornaram a graduação uma experiência prazerosa.

É um erro acreditar que é possível resolver qualquer problema importante usando batatas.

Douglas Adams

Resumo

As redes sociais criaram um ambiente compartilhado público, onde pessoas podem se comunicar, compartilhar informações, e conhecer outra pessoas independente de suas origens. Entretanto este espaço tem sido utilizado para propósitos maliciosos, seja para o compartilhamento de notícias falsas ou propagação de discurso de ódio, e com o auxilio de ferramentas modernas de omissão de identidade tais atos têm sido praticados sem que seja possível identificar o autor. Técnicas de de inteligência artificial já foram previamente utilizadas para atribuir a autoria de textos de autores desconhecidos, entretanto historicamente foram utilizadas em textos longos buscando muitas vezes a identificação de plágio, porém ao tratar de informações na internet é necessário considerar o curto numero de palavras utilizadas na comunicação online. Este trabalho testou técnicas já extensamente utilizadas para a atribuição de autoria ao serem aplicadas à textos retirados da internet utilizando métricas como a acurácia e matrizes de confusão. Além das técnicas clássicas foi testado também um modelo moderno de *deep learning* chamado BERT que tem sido aplicado a diferentes problemas devido à sua eficiência em lidar com linguagem natural. Após a testagem foi observado que o modelo BERT obteve os melhores resultados.

Palavras-chave: Inteligência artificial. Processamento de linguagem natural. Atribuição de autoria. BERT.

Abstract

The social networks created a shared public environment, where people can communicate with one another, share informations and meet other people regardless of their origins. However this space has been used for malicious purposes, being it sharing fake news or spreading hatred speech, and with the help of modern identity omission tools such acts have been practiced without it being possible to identify the author. Artificial Intelligence techniques have been previously used for attributing an authorship for texts with unknown authors, however historically they were used on long texts many times to identify plagiarism, but, when dealing with internet informations it's necessary to consider the small number of words used in online communication. This work tested techniques already extensively utilized for authorship attribution on texts taken from the internet, using metrics like accuracy and confusion matrices. Besides the classic techniques a modern deep learning model called BERT that has been applied to many different problems due to its efficiency when dealing with natural language was also tested. After the testing it was observed that the BERT model got the best results.

Key-words: Artificial intelligence. Natural language processing. Authorship attribution. BERT.

Lista de figuras

Lista de tabelas

Lista de abreviaturas e siglas

API	*Application Programming Interface*
BERT	*Bidirectional Encoder Representations from Transformers*
ML	*Machine Learning*
NB	*Naive Bayes*
RF	*Random Forest*
SVM	*Support Vector Machine*
TF-IDF	*Term Frequency - Inverse Document Frequency*

Sumário

1 Introdução

As redes sociais foram criadas com a intenção de gerar um ambiente virtual amigável onde conhecidos e estranhos pudessem interagir livremente. O *Facebook*, que segundo uma pesquisa feita pelo *site Statista*[1] em julho de 2021 é a rede social mais utilizada do mundo com quase 3 bilhões de usuários ativos, define sua missão em seu site oficial como "Dar às pessoas o poder de criar comunidades e aproximar o mundo".

Com o crescimento e avanço das redes sociais, indivíduos e organizações passaram a necessitar serem usuários presentes e ativos uma vez que a propagação de informação se tornou uma ferramenta essencial para competitividade. Então, apesar das razões nobres para o uso das redes sociais, é comum encontrar pessoas fazendo uso malicioso do ambiente virtual, divulgando informações falsas, praticando *cyberbulling*, propagando discursos de, além de diversos outros comportamentos nocivos (ABORISADE; ANWAR, 2018).

Um fator que colabora para a escalada do mau uso das redes sociais é a ascensão das técnicas de ocultação de identidade, que juntamente à popularização de celulares liberados, chips pré-pagos, redes Wi-Fi públicas, uso de técnicas como roteamento cebola e de softwares de navegação anônima, tornaram o rastreamento do pacote de dados malicioso muito mais complexo, garantindo a possibilidade de anonimato à pessoa que planeja realizar atos ilícitos na Internet (ROCHA et al., 2016).

Este problema tem tomado grandes proporções. Em 2015, o jornal *New York Times* revelou a existência de uma agência russa que supostamente seria focada no compartilhamento massivo de notícias falsas, e que teria campanhas financiadas por grandes órgãos e governos (CHENG, 2015). Eventos semelhantes levaram o Reino Unido, em 2017, a ameaçar grandes empresas como o Twitter e Facebook com sanções caso elas não auxiliassem na identificação de autores responsáveis por notícias falsas publicadas em suas redes sociais (HERN, 2017).

Mais recentemente, em outubro de 2021, um ex-funcionário anônimo do Facebook revelou ao jornal Washington Post que a empresa ignorava discursos de ódio e atividades ilegais, pois estas acabavam por gerar maior lucro (TIMBERG, 2021).

Neste contexto, a inteligência artificial com foco em processamento de linguagem natural surge como possível auxiliar na identificação de autores responsáveis por textos criminosos. A área de análise de autoria busca extrair informações de textos que revelem características de seu autor, e pode ser separada em três tarefas, a busca de característica sociolinguísticas de um autor, a autenticação de autoria e a que é objeto de estudo deste trabalho, a atribuição de autoria (FABIEN et al., 2020).

[1] https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/

Historicamente, a atribuição de autoria foi utilizada em casos de detecção de plágio ou na identificação de autores para textos históricos (FABIEN et al,2020), e em grande parte aplicada em textos longos que carregam muita informação. Entretanto, a comunicação na Internet se dá por mensagens e publicações formadas por textos curtos, dificultando, deste modo, o trabalho de extração de características (SCHWARTZ et al., 2013).

Assim, o objetivo deste trabalho é implementar técnicas de extração de características e classificadores já bem fundamentados para a resolução do problema de atribuição de autoria, além do modelo BERT, apresentado em 2018 e que tem obtido resultados exepcionais em tarefas de processamento de linguagem natural, e aplica-los em textos curtos retirados da Internet para avaliar e comparar suas acurácias na tarefa de atribuição de autoria.

Além deste capítulo introdutório, este trabalho contém os seguintes capítulos:

Capítulo 2: Apresenta a fundamentação teórica necessária para o desenvolvimento do trabalho;

Capítulo 3: Apresenta alguns trabalhos correlatos encontrados na literatura;

Capítulo 4: Apresenta a metodologia do trabalho, com suas etapas de desenvolvimento e as ferramentas utilizadas;

Capítulo 5: Apresenta detalhes do desenvolvimento do trabalho, com informações sobre dados, métricas, modelos e resultados;

Capítulo 6: Apresenta as conclusões e potenciais trabalhos futuros.

2 Fundamentação Teórica

Neste capítulo serão apresentados os conceitos teóricos necessário para a execução do trabalho, na sessão 2.1 é dada uma visão geral da área de processamento de linguagem natural e são apresentadas algumas técnicas de extração de características e representação de linguagem natural, na sessão 2.2 são discutidos alguns algoritmos de classificação que foram utilizados, e na sessão 2.3 é são apresentados alguns conceitos do algoritmo de *deep learng* BERT, um algoritmo introduzido recentemente focado em tarefas de processamento de linguagem natural.

2.1 Processamento de Linguagem Natural

Chowdhury (2003) define processamento de linguagem natural como a área de pesquisa que estuda a utilização de computadores para compreender e manipular linguagens humanas, como inglês e português. Esta área, portanto, busca simular as ferramentas utilizadas por pessoas em conversações em máquinas.

Utilizando ferramentas que misturam estudos de linguística, matemática, inteligência artificial e estatística, diversos pesquisadores buscam solucionar tarefas como a de reconhecimento de fala, tradução, análise de sentimento, sumarização e análise de autoria.

2.1.1 Bag of Words

Uma estratégia clássica para representação de palavras para o computador é a *Bag of Words*, que consiste na representação de textos como dicionários não ordenados de frequência de cada palavra presente no conjunto de treinamento. Estes dicionários podem indicar quantas vezes a palavra aparece no texto ou com um valor inteiro ou um valor booleano apenas indicando se ela aparece ou não (ABORISADE; ANWAR, 2018). Estas *Bag of Words* podem ser geradas para cada autor separadamente ou uma para todo o conjunto de autores analisado. Entretanto, apesar do potencial de captar mais características, gerar um dicionário para cada autor pode acabar tomando uma grande quantidade de tempo (ROCHA et al., 2016).

Como exemplo de seu funcionamento tome as seguintes frases curtas:

Frase 1: "Eu comi laranja"

Frase 1: "Eu tomei suco de laranja"

Inicialmente, cria-se um dicionário listando todas as palavras presentes no conjunto e as relacionando a um índice único, representado na tabela 1.

Tabela 1 – Dicionário gerado apartir das frases exemplo.

1	2	3	4	5	6
Eu	comi	laranja	tomei	suco	de

Fonte: Elaborada pelo autor

Com o dicionário pronto, as frases podem ser representadas com vetores individuais da seguinte maneira:

Frase 1: [1, 1, 1, 0, 0, 0]
Frase 2: [1, 0, 1, 1, 1, 1]

Quando se utiliza a representação de *Bag of Words* é comum se retirar palavras como artigos, preposições, pronomes e outras palavras que são classificados na língua inglesa como "palavras funcionais", porém considerando a atribuição de autoria para textos curtos estas palavras podem trazer informações que não podem ser descartadas (ROCHA et al., 2016).

2.1.2 N-Grams

N-grams é uma técnica já bem fundamentada na área de processamento de linguagem natural, que consiste na separação do texto em fatias de n valores consecutivos, podendo ser definidas em nível de palavras ou caracteres visando encontrar similaridades na frequência de uso de certas sequências (FRANTZESKOU et al., 2006).

O uso de *N-grams* à nível de caractere é comum ao analisar textos de redes sociais, uma vez que este é capaz de captar o uso de emoticons ou o uso incomum de sinais de pontuação e letras. Outro fator vantajoso do uso de *N-grams* à nível de caractere é que pequenos erros ortográficos acabam sendo ignorados (ROCHA et al., 2016), tomando como exemplo a seguinte frase:

Frase 1:

```
''Bom diaaa !!''
```

Seria separada em *N-grams* de tamanho 2 e 3 da seguinte maneira:

2-grams: "Bo","om","m ", " d","di","ia","aa","aa","a "," !","!!"
3-grams: "Bom","om ","m d"," di","dia","iaa","aaa","aa ","a !"," !!"

Mesmo com a grafia incorreta da palavra "dia", a frase compartilha diversos *N-grams* com a frase corretamente escrita "bom dia", permitindo ao computador identificar a frase e o padrão do uso de repetição de letras como ênfase.

Já o uso de *N-grams* à nível de palavras permite a retirada de características semânticas ao unir diferentes palavras (ROCHA et al., 2016), tomando como exemplo as seguintes frases e as suas separações em *N-grams* de tamanho 2:

Frase 1: "A sua camiseta está com a manga amassada"

2-gram: "A sua","sua camiseta", "camiseta está", "está com", "com a", "a manga", "manga amassada"

Frase 2: "Ontem comi uma manga deliciosa"

2-gram: "Ontem comi", "comi uma", "uma manga", "manga deliciosa"

A palavra "manga" é claramente utilizada com significados diferentes em cada frase, porém ao unir a palavra "manga" com as palavras "amassada" ou "deliciosa" se torna mais fácil deduzir o seu significado na frase.

2.1.3 Term Frequency - Inverse Document Frequency (TF-IDF)

TF-IDF é uma medida de frequência para uma palavra específica. Este valor para uma determinada palavra aumenta proporcionalmente à medida que aumenta o número de ocorrências desta palavra no texto analisado. No entanto, esse valor é equilibrado pela frequência da palavra no *corpus* (conjunto total de textos). Desta maneira, palavras utilizadas por diversos autores assumem pesos inferiores por adicionarem pouca informação, enquanto que palavras menos utilizadas assumem maior influência, pois são de uso mais individual (BACCIU et al., 2019).

Dados um *corpus* (conjunto de textos) D, uma palavra p e um documento $d \in D$, o cálculo para este valor pode ser dado conforme definido na Equação 2.1:

$$p_d = f_{p,d} * log(\frac{|D|}{f_{p,D}}) \tag{2.1}$$

onde $f_{p,d}$ é o numero de vezes que a palavra p aparece em d, $|D|$ é o numero de documentos no corpus e $f_{p,D}$ é o numero de vezes que a palavra p aparece em D (RAMOS et al., 2003).

O conceito de TF-IDF apesar de ter sido introduzido para representar o valor de uma palavra de um *corpus* pode ser aplicado a outras característica, como por exemplo a *n-grams* como os apresentados na seção 2.1.2.

2.1.4 Word Embedding

A representação de palavras em muitas técnicas sofre com generalizações que afetam o desempenho final em tarefas de processamento de linguagem natural, por exemplo, é normal

que palavras como "futebol" e "vôlei" tenham representações completamente desconexas mesmo que possuam contextos significativamente semelhantes (LEVY; GOLDBERG, 2014).

A técnica de *word-embedding* busca solucionar essa questão, trazendo representações para palavras que incluam contextos sintáticos e semânticos, os termos são então representados por longos vetores em que cada posição apresenta com um valor numérico que relaciona a palavra com um contexto específico. Desta maneira, além de carregar grande quantidade de informações torna as operações computacionais fáceis uma vez que estas se tornam operações de matrizes de pequena dimensão (LEVY; GOLDBERG, 2014).

Para exemplificação segue um *embedding* fictício da palavra "rei":

$$\text{Rei} = [0.32\ 0.12\ 0.98\ 0.56\ 0.68]$$

Para simplificar o exemplo o *embedding* foi representado como um vetor de dimensão 5, porém, o comum é que este valor seja maior. É possível imaginar que o valor 0.98 esteja relacionado a gênero, desta maneira valores próximos de 1 significam que a palavra é masculina enquanto valores próximos de zero que a palavra é feminina, entretanto estes valores são calculados pela máquina, sendo assim impossível realmente compreender qual característica está sendo quantificada.

Em alguns modelos, como é o caso do BERT (*Bidirectional Encoder Representations from Transformers*), que será abordado posteriormente neste trabalho, é possível utilizar um *token* especial para cada texto a ser codificado, desta maneira a saída correspondente a este *token* traz reunidas informações correspondentes a frase inteira (DEVLIN et al., 2018).

2.2 Classificadores

Após a extração de características, as informações obtidas podem ser utilizadas para treinar classificadores para que estes aprendam a classificar padrões a partir dos textos. Na literatura, diversos classificadores diferentes têm sido utilizados para realizar a tarefa de atribuição de autoria. Alguns desses classificadores, os mais clássicos, são apresentados nas próximas subseções.

2.2.1 Naive Bayes

Naive Bayes (NB) é um algoritmo de aprendizado supervisionado baseado no teorema de Bayes, um teorema probabilístico criado a partir dos estudos do matemático inglês Thomas Bayes, muito utilizado na área de classificação de textos (ABORISADE; ANWAR, 2018). Este algoritmo, entretanto, se diferenciando do teorema original, assume que os vetores de características são independentes. Apesar dessa suposição raramente representar situações reais, o classificador NB já obteve ótimos resultados em categorizações de texto (ZHANG, 2004).

Assim, dada uma classe c e um documento X, a regra de Bayes pode ser dada da seguinte forma:

$$P(c|X) = \frac{P(X|c)P(c)}{P(X)} \tag{2.2}$$

Tendo então o conjunto de classes C, calcula-se então a relação de classe mais provavel:

$$F_b(X) = arg\max_{c \in C} P(X|c)P(c) \tag{2.3}$$

onde $F_b(X)$ é um classificador Bayesiano. Após a extração de características o documento X é decomposto em um conjunto de valores $(x_1, x_2...x_n)$, assim a relação deve ser reescrita:

$$F_b(X) = arg\max_{c \in C} P((x_1, x_2...x_n)|c)P(c) \tag{2.4}$$

Entretanto calcular a probabilidade de cada valor de característica do documento X pode levar à um problema de dados escassos uma vez que um documento não acumula todas as características, então utilizando a suposição de que os vetores são idenpendentes o classificador pode ser simplicado da seguinte maneira:

$$P((x_1, x_2...x_n)|c) = \prod_{i=1}^{n} P(x_i|C) \tag{2.5}$$

$$F_b(X) = arg\max_{c \in C} P(C)\prod_{i=1}^{n} P(x_i|C) \tag{2.6}$$

2.2.2 Random Forest

O classificador *Randon Forest* (RF), floresta aleatória, é um algoritmo de classificação famoso, que tem sido bastante aplicado à problemas de texto pela sua habilidade em lidar com informações de grande dimensão com boa performance. O seu funcionamento simplificadamente consiste na união de diversas árvores de decisão (XU et al., 2012).

Uma arvore de decisão é uma abordagem de classificação que utiliza do conceito de dividir e conquistar, em suma uma arvore de decisão é uma arvore binária em que cada nó representa uma característica, o algoritmo busca então o valor que separa uma classe da outra para esta característica, sendo o maior esforço do algoritmo então identificar os valores ótimos de partição (MYLES et al., 2004).

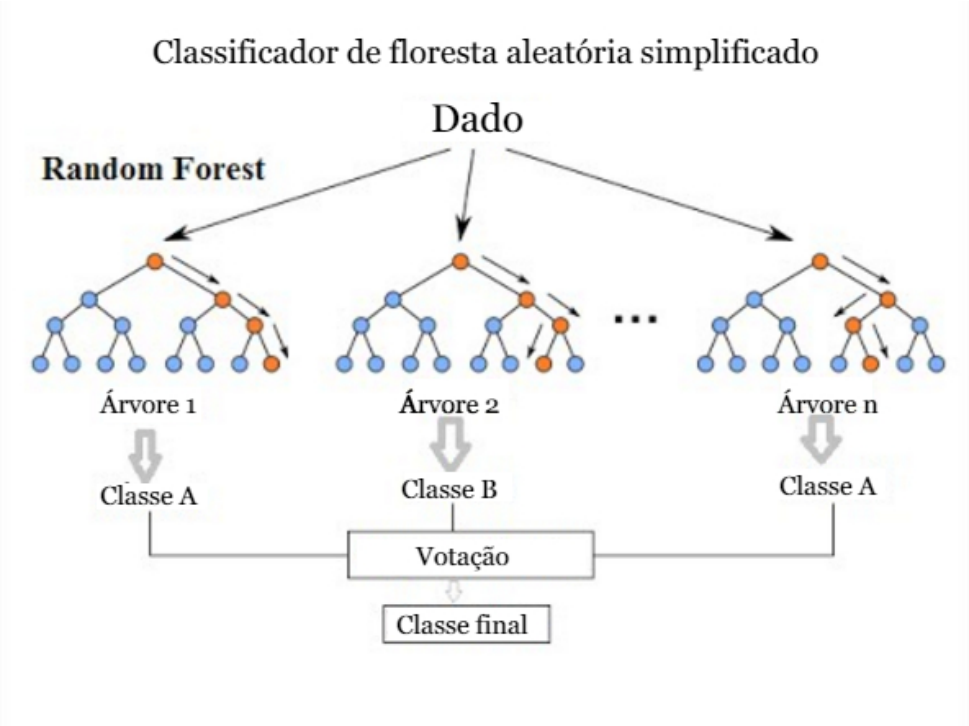
Tendo um conjunto de treinamento T, a criação de um classificador RF consiste na amostragem aleatória repetida de T, sendo que para cada amostragem é criada uma árvore de decisão para os valores escolhidos e esta é armazenada. Árvores de decisão têm como

objetivo prever a classe de um documento ao inferir regras simples com os valores de suas características.

Após a criação de um número determinado de árvores de decisão, o algoritmo passa a decidir qual a classe mais provável de um documento utilizando um sistema de votação, os dados são inseridos em todas as árvores e a classe mais votada em maioria simples é a escolhida pelo algoritmo (ROCHA et al., 2016), desta maneira buscando evitar problemas com *overfitting*, que são comuns para árvores de decisão.

A figura 1 representa graficamente um classificador de floresta aleatória simplificado.

Figura 1 – Representação simplificado do classificador de floresta aleatória.



Fonte: Jagannath (2020)

2.2.3 Support Vector Machine

Support Vector Machines (SVM), ou máquinas de vetores de suporte em português, é uma técnica de aprendizado de máquina amplamente aplicada em problemas de categorização de texto, análise de imagem e diversos outros. Baseada na teoria de aprendizado estático, busca separar dados em duas classes a partir da obtenção de um hiperplano que descreva a fronteira entre elas, obtendo uma margem máxima que as separa (LORENA; CARVALHO, 2007).

Considerando um espaço de dados X, a equação de um hiperplano pode ser dada como:

$$w * x + b = 0 \tag{2.7}$$

onde w é o vetor normal ao hiperplano, $x \in X$ e b um número real. Esta equação divide o espaço de dados X em duas partes:

$$w * x + b > 0 \tag{2.8}$$

$$w * x + b < 0 \tag{2.9}$$

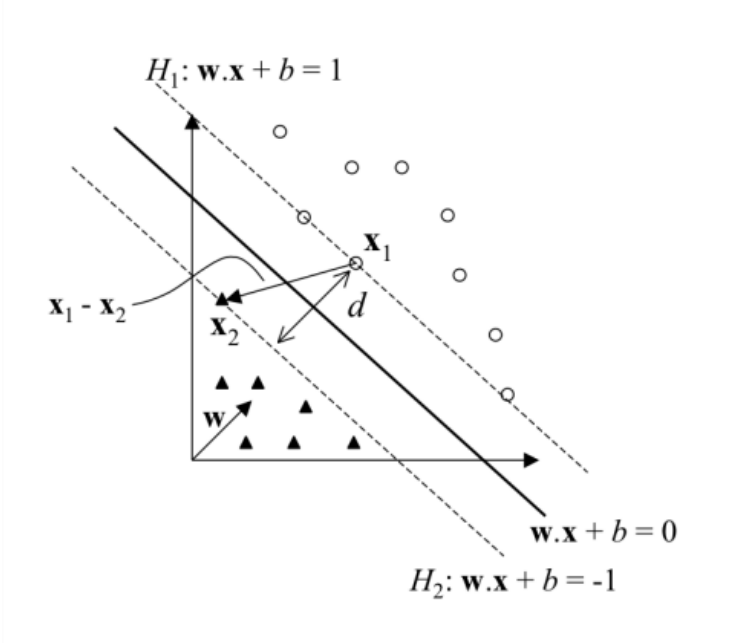
Os valores de w e b são escolhidos de maneira que os hiperplanos mais próximos de $w * x + b = 0$ respeitem a equação:

$$|w * x + b| = 1 \tag{2.10}$$

A Figura 2 ilustra a seguinte situação, tendo x_1 pertencente ao hiperplano H_2:$w*x+b = 1$ e x_2 pertencente ao hiperplano H_2:$w * x + b = -1$, a distância destes hiperplanos pode ser calculada pelo módulo da projeção de $x_1 - x_2$ na direção de w, dada pela equação:

$$(x_1 - x_2)(\frac{w}{|w|} * \frac{(x_1 - x_2)}{|(x_1 - x_2)|}) \tag{2.11}$$

Figura 2 – Cálculo da distância entre hiperplanos para definição da margem máxima.



Fonte: Lorena e Carvalho (2007)

Sabendo que a subtração das equações dos hiperplanos H_1 e H_2 fornece a relação:

$$w * (x_1 - x_2) = 2 \tag{2.12}$$

a equação de distância entre os hiperplanos H_1 e H_2 pode ser dada como:

$$\frac{2}{|w|} \tag{2.13}$$

assim a distância d que separa as duas classes do conjunto de dados do hiperplano separador inicial é:

$$d = \frac{1}{|w|} \tag{2.14}$$

esta então é chamada de margem geométrica do classificador.

Assim, o trabalho de maximização da margem é feita pela minimização de $|w|$ na seguinte função:

$$\underset{w\,,b}{min} : \frac{1}{2}|w|^2 \tag{2.15}$$

$$s.a : y_i(w * x_i + b) - 1 \geq 0,\, \forall i = 1, 2, ...n$$

onde y_i é o rótulo associado a x_i e n o número total de dados. Com as restrições determinadas é possível garantir que não haja dado na área da margem de separação.

Entretanto, esse método parte do princípio que o conjunto de dados é linearmente separável, sem que os dados se sobreponham, este modelo é conhecido como SVM de margens rígidas. Situações com dados linearmente separáveis são muito incomuns, seja pela natureza do problema ou ruidos, então para lidar com problemas que não respeitam essa regra um conjunto de variáveis ξ de folga é introduzido, então o problema de minimização assume a seguinte forma:

$$\underset{w\,,b\,,\xi}{min} : \frac{1}{2}|w|^2 + C(\sum_{i=1}^{t} \xi_i) \tag{2.16}$$

$$s.a : y_i(w * x_i + b) - 1 \geq 0,\, \forall i = 1, 2, ...n\,, \xi \geq 0$$

onde C é um fator de regulaziração a ser escolhido de maneira a melhor lidar com o problema trabalhado.

Desta maneira, com os valores w e b calculados, os dados podem ser separados em duas classes analisando o sinal do resultado obtido na aplicação da equação do hiperplano separador.

2.3 BERT

Bidirectional Encoder Representations from Transformers, ou simplesmente BERT, é um modelo de representação de linguagem introduzido por Devlin et al. (2018). Apesar de ser relativamente recente, o modelo obteve resultados de alta qualidade em diferentes tarefas de processamento de linguagem natural.

O modelo foi criado com o objetivo de obter melhores representações de palavras ao coletar contexto bidirecionalmente, assim a representação é afetada tanto por palavras que surgiram antes na frase quanto depois, adicionando dessa forma uma melhor compreenção de contexto para o algoritmo. O modelo é pré-treinado com uma grande quantidade de dados em tarefas baseadas em máscaras e depois pode ser especializado para uma outra tarefa qualquer de processamento de linguagem natural.

Nas próximas subseções serão descritos alguns conceitos relacionados ao modelo BERT.

2.3.1 Arquitetura

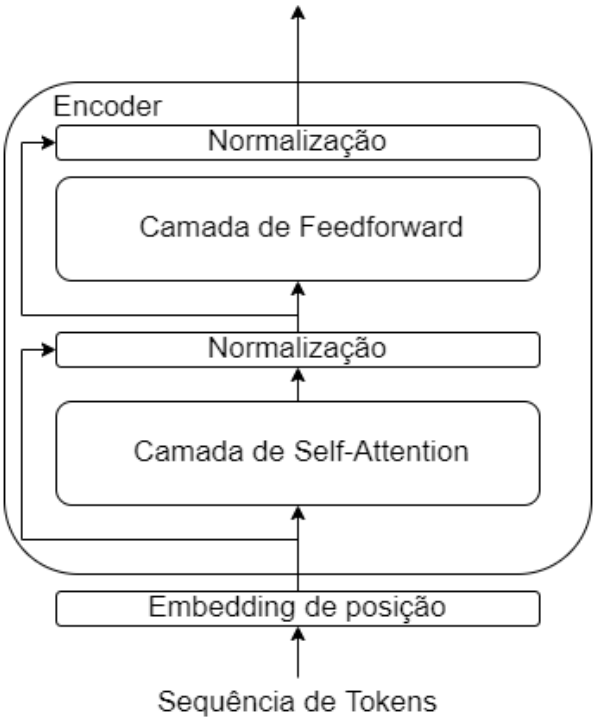
A arquitetura do modelo BERT se baseia fortemente na estrutura proposta por Vaswani et al. (2017) para o modelo chamado de *Transformer*, entretanto o BERT se utiliza somente do componente referenciado na literatura como *Encoder*, sendo que na sua versão básica são empilhados seis destes componentes.

O *Encoder* é essencialmente formado por duas camadas, uma é um mecanismo de *self-attention* com múltiplas cabeças, uma tecnologia que tem se tornado muito popular nos últimos anos na área de processamento de linguagem natural que permite a extração bilateral de contexto, e uma camada simples *feedforward* completamente conectada. Entre cada camada é realizado um processo de normalização onde o valor passado para a próxima camada é o resultado da normalização entre o resultado da camada atual somado à entrada da camada. A Figura 3 mostra as camadas de um *Encoder*.

2.3.2 Entrada

O modelo BERT aceita como entrada uma ou duas frases, para problemas como o de resposta a perguntas, transformadas em uma sequência de *tokens*. O sistema de *tokens* utilizado é o proposto por Wu et al. (2016) com o algoritmo *Wordpiece* que conta com um vocabulário de 30.000 *tokens*. Toda sequência de *tokens* é iniciada com um *token* especial, [CLS], que ao final do processamento agregará informações da sequência inteira podendo ser utilizado em problemas de classificação, e um *token* [SEP] que demarca o final de uma frase, como o modelo soluciona tarefas com multiplas frases este *token* é também utilizado para separar o inicio de uma e o final de outra.

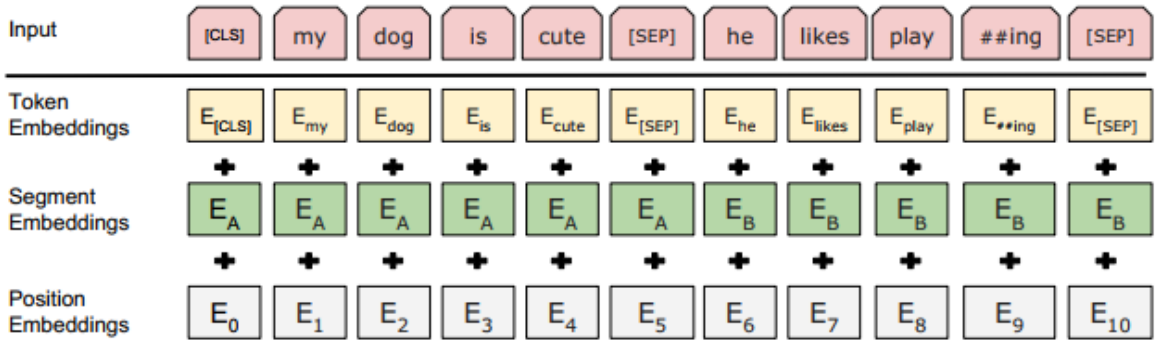
Figura 3 – Camadas do *Encoder*.



Fonte: Elaborada pelo autor.

A entrada é então formada com a soma dos *embeddings* de cada palavra, aos de segmento e posição, como representados na Figura 4.

Figura 4 – Representação de entrada para o modelo BERT.



Fonte: Devlin et al. (2018)

2.3.3 Pré-treinamento

O algoritmo BERT é pré-treinado para conseguir uma melhor compreenção do funcionamento de linguagens naturais, ele é treinado em duas tarefas não supervisionadas baseadas em máscaras.

A primeira tarefa, objetivando um treinamento para representações bilaterias, é chamada de modelagem de linguagem mascarada e consiste na substituição aleatória de quinze por cento dos *tokens* de todas as frases, com o objetivo de que esta palavra seja descoberta unicamente por contexto. Entretanto, para evitar que o *token* de máscara influencie o procedimento de *fine-tuning*, em dez por cento dos casos a palavra escolhida a ser mascarada será substituida por algum outro *token* aleatório e em dez porcento mantém o *token* original, assim usa-se o *token* de mascará em oitenta por cento.

Como exemplo segue a seguinte frase:

"[CLS] Deus ajuda quem [MÁSCARA] madruga [SEP]"

Então utilizando apenas o contexto dado pelas palavras em volta o algoritmo deve buscar o *embedding* para o *token* máscara e alcançar valores semelhantes ao do *embedding* da palavra ocultada.

Como o modelo busca solucionar problemas como o de resposta a perguntas, a segunda tarefa foca na compreensão da relação entre duas frases. São, então, fornecidas duas frases, sendo que em cinquenta por cento dos casos a segunda frase é sequência da primeira e nos outros cinquenta por cento dos casos a segunda frase é uma frase aleatória do conjunto de dados. Neste caso, a tarefa consiste em classificar a frase como sendo sequência ou não.

As duas frases a seguir representam esta tarefa:

Caso 1:"[CLS] Deus ajuda quem [SEP] cedo madruga"
Caso 2:"[CLS] Deus ajuda quem [SEP] não se olha os dentes"

O modelo deve então, utilizando seu conhecimento sobre linguagem natural, categorizar se as frases formam uma sequência, respondendo neste caso que no caso 1 as frases são uma sequência e que no caso 2 não.

3 Trabalhos Correlatos

Nos últimos anos, vários trabalhos procuraram resolver o problema de atribuição de autoria, diversos modelos e combinações de técnicas foram propostas para melhorar o desempenho dos algoritmos nesta tarefa (ABORISADE; ANWAR, 2018). Alguns trabalhos já procuraram focar os esforços na resolução do problema em casos de textos curtos que acumulam uma quantidade inferior de informações, obrigando o projetista a focar seus esforços na retirada da maior quantidade possível de informações (SCHWARTZ et al., 2013).

Neste capítulo serão apresentados alguns trabalhos feitos na área de processamento de linguagem natural relacionados a atribuição de autoria e ao modelo BERT.

3.1 Trabalhos na área de atribuição de autoria

Frantzeskou et al. (2006) procuram solucionar o problema de atribuição de autoria para códigos de programa, com o objetivo de encontrar o autor de um código malicioso ou em uma questão de disputa legal. O trabalho busca propor um modelo capaz de identificar o correto autor de um código de um conjunto predefinido para linguagens C++ e Java, provando ainda que a ausência de comentários não afeta o desempenho.

Schwartz et al. (2013) propõem um modelo para atribuição da autoria para micro-mensagens, se utilizando de *n-grams* para representação das palavras, introduzindo também o conceito de *K-signature*, que seria uma característica que aparece em uma quantidade considerável de textos de um autor, mas que não se repete no conjunto de dados.

Rocha et al. (2016) analisam técnicas utilizadas na atribuição de autoria com foco na análise forense de textos publicados em redes sociais, especificamente o Twitter. O trabalho estuda a acurácia de diferentes combinações de técnicas e classificadores e propõe que uma união de algoritmos automatizados de atribuição de autoria possam ser uma ferramenta para solução de crimes virtuais.

Aborisade e Anwar (2018) também propõem uma comparação de modelos para atribuição de autoria para textos retirados do Twitter, porém eles focam na separação dos dados entre autores conhecidos ou desconhecidos e na acurácia dos classificadores de *Naive Bayes* e de regressão logística neste contexto.

3.2 Trabalhos relacionados ao BERT

O modelo de *Transformers* proposto por Vaswani et al. (2017) foi de grande impacto no campo do processamento de linguagem natural, obtendo resultados excepcionais na tarefa

de tradução. O trabalho propôs um modelo mais simples do que as redes convolucionais que eram dominantes na área de tradução se baseando unicamente na tecnologia de *attention*, que buscar colher influências de palavras presentes na frase inteira.

Devlin et al. (2018) apresentou o modelo BERT, baseado no modelo de Vaswani et al. (2017), que busca uma representação de palavras com influência bidirecional apartir de um pré-treinamento com textos não rotulados, podendo assim ser especializado para outras tarefas de processamento de linguagem natural treinando uma última camada de saída adicional.

Fabien et al. (2020) propuseram sua utilização na tarefa de atribuição de autoria devido a sua alta performance na classificação de sequências de palavras, sendo que os autores unem o modelo com outras arquiteturas clássicas em um esquema de votação e testam seu desempenho para diferentes conjuntos de dados.

4 Metodologia

Neste capítulo são apresentadas na seção 4.1 os principais passos para o desenvolvimento do trabalho, e na seção 4.2 são introduzidas as principais ferramentas e bibliotecas utilizadas para a execução do trabalho.

4.1 Etapas

Este trabalho iniciou com uma pesquisa bibliográfica para primeiramente compreender a área de processamento de linguagem natural e seus problemas. Foram estudados também os modelos clássicos para a resolução do problema de atribuição de autoria. Durante esta fase foram escolhidas as técnicas, classificadores e modelos implementados posteriormente.

Na sequência, foram pesquisadas bibliotecas e ferramentas que seriam de auxílio para implementação dos modelos, tendo sido escolhida, após tais pesquisas, a linguagem Python para programação devido ao grande número de pacotes e serviços com foco na áreas de aprendizado de máquina e os métodos foram então implementados. Foram também colhidos os dados a serem utilizados para a avaliação de acurácia.

Finalmente, foram efetuados então o treinamento dos diferentes modelos e obtidas as medidas de acurácia para os diferentes cenários.

4.2 Ferramentas

4.2.1 SciKit Learn

Scikit Learn é uma biblioteca *Open Source* para Python que implementa diversos métodos de IA (KRAMER, 2016). A biblioteca oferece algoritmos de classificadores como de florestas aleatórias e SVM com uma interface de fácil uso (PEDREGOSA et al., 2011).

Os estimadores fornecidos pela biblioteca possuem métodos simples *fit* e *predict* que facilitam o treinamento e a predição para conjuntos de dados fornecidos pelo programador. A biblioteca também possui métodos de extração de informação para diversas tarefas e funções para o cálculo de diversas métricas de avaliação para algoritmos de inteligência artificial.

4.2.2 Numpy

NumPy é uma biblioteca *Open Source* focada em computação científica para a linguagem Python que fornece métodos e estruturas que permitem cálculos mais poderosos e mais rápidos. Dentre as funcionalidades mais importantes implementadas pelo pacote está o objeto *array*

de n dimensões e todas as operações aritméticas e lógicas associadas a ele e a capacidade de *broadcasting* que permite operações entre *arrays* de diferentes dimensões, que são essenciais para diversos algoritmos de aprendizado de máquina(NUMPY, c2008).

Devido à qualidade de suas implementações as funcionalidades disponibilizadas pelo pacote são também fundamentais para o funcionamento de métodos de outras bibliotecas como a *Scikit Learn* e *Hugging Face*.

4.2.3 Hugging Face

Hugging Face é um biblioteca *Open Source* para Python que traz diversas ferramentas para facilitar a implementação de modelos de aprendizado de máquina atuais. O pacote possui dezenas de modelos pré-treinados em grandes conjuntos de dados especializados para diferentes tarefas principalmente na área de processamento de linguagem natural (WOLF et al., 2019).

A biblioteca também possui métodos para adaptar conjuntos de dados de outras bibliotecas para os seus modelos e ferramentas de treinamento para aperfeiçoamento dos modelos para outras tarefas com hiperparâmetros e métricas personalizáveis.

4.2.4 Pandas

Pandas é uma biblioteca *Open Source* para Python que fornece estruturas de dados flexíveis e métodos que facilitam a manipulação de dados relacionais ou rotulados. O pacote funciona como auxiliar a outros pacotes de dados fornecendo suas estruturas como objetos a serem manipulados (PANDAS, c2008).

A biblioteca disponibiliza uma estrutura chamada "Dataset" que permite diversas operações e funcionalidades como leitura e escrita de dados externos, manipulação de colunas e linhas, uso de dados nulos, entre outras.

4.2.5 Google Colab

Google Colab[1] é um serviço disponibilizado pela Google que permite o acesso por nuvem de máquinas com alto poder computacional. A ferramenta possui planos gratuitos e pagos, ambos dando acesso à GPU's sem a necessidade de configuração.

O serviço funciona como um ambiente acessível por um navegador de células de código, muito semelhante ao *Jupyter Notebook*, que permitem execução segmentada de códigos em Python. A principal vantagem da utilização da ferramenta é a possibilidade de lidar com grandes quantidades de dados e utilizar algoritmos com cálculos pesados sem exigir da máquina local, permitindo assim a execução de algoritmos no estado da arte mesmo em computadores mais fracos.

[1] https://colab.research.google.com/

4.2.6 Twitter API

A Twitter API é uma API disponibilizada pelo Twitter que permite acesso às funções do aplicativo a partir de códigos. A ferramenta não é disponibilizada para o público geral, sendo necessário um pedido de aprovação para uma conta do tipo desenvolvedor em um curto processo. Uma das funções mais importantes da ferramenta é a possibilidade de busca de *tweets* utilizando filtros de usuário, palavras-chave, localização e diversos outros (TWITTER, c2022).

Para utilizar a API com a linguagem Python é necessário utilizar uma biblioteca auxiliar chamada Tweepy que atua como intermediária entre o código e a API, o pacote também implementa de maneira intuitiva todo o processo de autenticação e diversos métodos para a coleta de dados.

5 Desenvolvimento

Neste capítulo são detalhados os passos do desenvolvimento do trabalho, na seção 5.1 é apresentado o processo para a obtenção de dados, são discutidas caracaterísticas do corpus e descritos os procedimentos de pré processamento, na seção 5.2 são apresentadas as métricas utilizadas para avaliação dos modelos, na seção 5.3 são detalhados os modelos que foram implementados no trabalho, na seção 5.4 são retratados os parâmetros do treinamento dos modelos e finalmente na seção 5.5 são incluídos e discutidos os resultados dos teste.

5.1 Dados

5.1.1 Obtenção dos Dados

Como foi previamente descrito, os dados utilizados neste trabalho são postagens da rede social *Twitter*, sendo o conjunto de dados resultado da união entre arquivos encontrados na *Kaggle*[1], uma das mais conhecidas plataformas para competições de Ciência de Dados fundada em 2010 por Anthony Goldbloom e adquirida pelo Google (Alphabet), em 2017.

Para obtenção dos dados foi necessário a utilização da *Twitter API*, uma API criada pelo *Twitter* que permite acesso a diversas ferramentas para a captação e análise de informações da plataforma. Para ter acesso a ela foi necessário submeter um pedido de aprovação para a empresa, que consistia em um questionário acerca das intenções de uso dos dados. Com o acesso concedido pela empresa foi utilizado também a biblioteca *Tweepy* para facilitar o acesso a API.

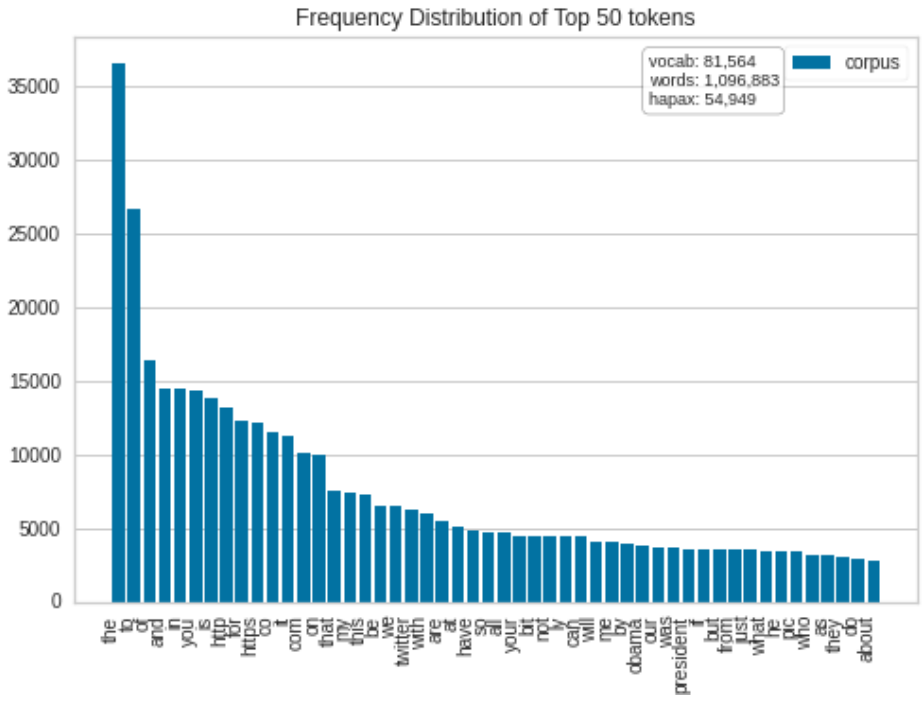
Como o problema abordado neste trabalha consiste na identificação de autoria, o conjunto de dados necessita de apenas duas características, o texto e o autor. Devido ao grande número de artigos e bibliotecas preparadas para lidar com a lingua inglesa foi definido que as publicações utilizadas fossem somente nesta língua, descartando as feitas em outras linguas. Foi necessário também excluir os textos compostos por menos de quatro palavras, uma vez que estes trazem poucas informações e podem resultar em ruído (ABORISADE; ANWAR, 2018). Outra característica que exigiu atenção durante a obtenção dos dados foi a presença de *retweets*, que são publicações feitas por outros autores mas que o autor analisado publica em seu próprio perfil, então como estes são escritos por outros usuários foi necessário excluir todos aqueles que apresentavam a marcação de *retweet*.

A Figura 5 apresenta o cojunto de *tokens* mais frequentes no conjunto de dados. Nela é possível verificar que as palavras mais presentes são artigos, conjunções e preposições, conhecidas

[1] https://www.kaggle.com

em inglês como "palavras de função". A Figura 6, por sua vez, traz os *tokens* mais utilizados excluindo as "palavras de função". Nesta figura é possível observar que palavras relacionadas a *links* aparecem nas primeiras posições e serão lidadas durante o pré-processamento, e em seguida aparecem as palavras de cunho politico como "obama" e "president".

Figura 5 – Distribuição de frequência dos 50 Tokens mais comuns no conjunto de dados.



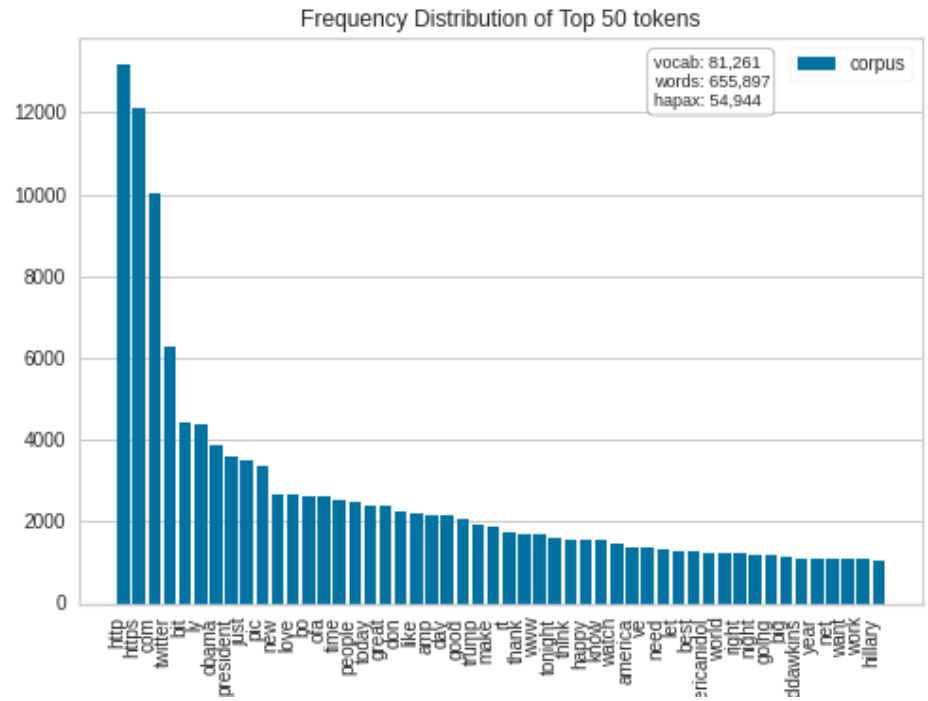
Fonte: Elaborada pelo autor.

Entretanto, ao observar a distribuição de textos por autor, apresentada na Figura 7, é possível perceber que alguns autores possuem uma quantidade muito maior de textos e acabam criando um viés no conjunto, isto ocorre pois os conjuntos de dados obtidos na plataforma *Kaggle* superam em muito o limite imposto pela *Twitter API*. Então, durante o treinamento foram selecionados a mesma quantidade de dados de cada autor para evitar vieses.

5.1.2 Pré-processamento dos Dados

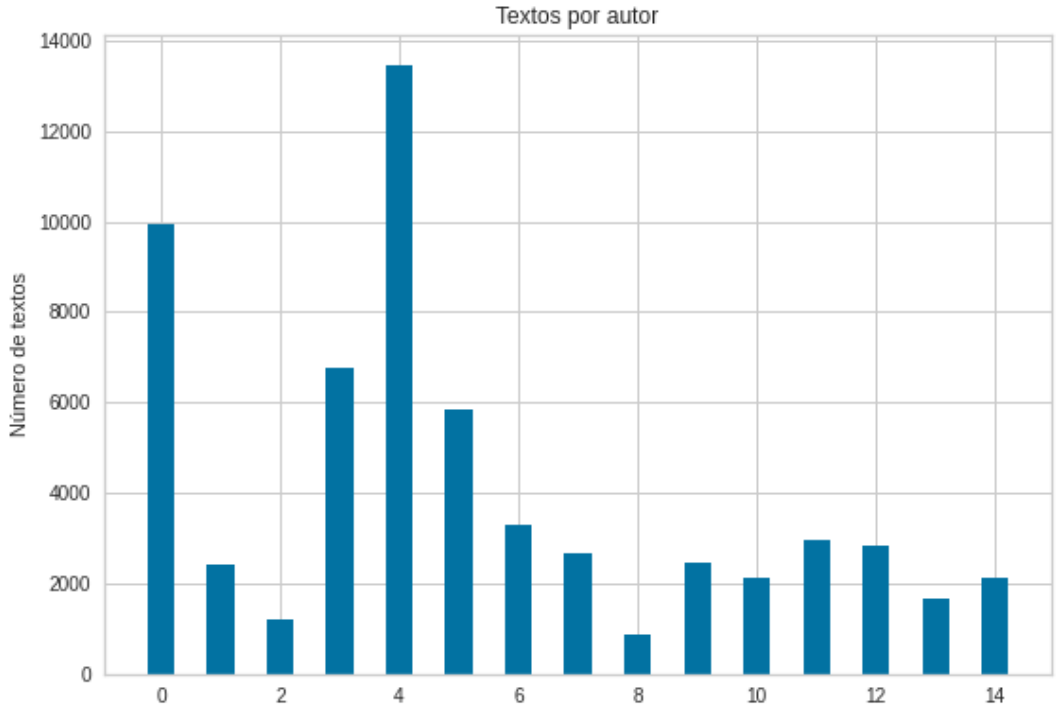
Antes de encaminhar os dados obtidos para a obtenção de características textuais foi necessário um estágio de pré-processamento para evitar que fatores de pouca importância se propaguem e possam causar problemas no estágio de atribuição do autor. Varios trabalhos na área de processamento de linguagem natural fazem pré-processamento focado na retirada de erros ortográficos, *emojis* ou capitalização de letras, entretanto tais fatores podem trazer características importantes no momento de definição do autor, como um erro ortográfico muitas vezes repetido, o uso de gírias ou abreviações ou a preferência por repetição de pontuações (ROCHA et al, 2016).

Figura 6 – Tokens mais comuns no conjunto de dados excluindo as "palavras de função".



Fonte: Elaborada pelo autor.

Figura 7 – Distribuição de textos por autor.



Fonte: Elaborada pelo autor.

Então, o pré-processamento foi feito focado na retirada de palavras que trazem poucas informações ou que poderiam resultar em classificação incorreta e na substituição destas por marcadores. Assim, referências à datas, horários e números foram subtituidos pelas marcações "DATE", "TIME" e "NUM", respectivamente, uma vez que é improvável que um autor as repita, no Twitter um usuário pode responder à publicações de outro ou pode marca-lo por meio de uma citação à seu nome acompanhado do caractere "@", estas referências a outros usuários e endereços de sites foram trocados por "REF" e "URL", referências e *links* podem ser longos e ocupar um grande numéro dos caracteres da limitação que todo *tweet* está sujeito com palavras que não são da escolha do autor. É interessante também notar que no caso de análise de autoria para razões criminais um usuário que utiliza uma conta falsa dificilmente iria se comunicar por ela com seus contatos usuais (LAYTON et al, 2010).

Alguns exemplos de *tweets* antes e depois do pré-processamento.

Antes do pré-processamento:

Tweet 1: "Entrei pelo menos 10 vezes nesse site https://www2.unesp.br/"
Tweet 2: "Não se esqueçam da SICOMP dia 04/10/2021"
Tweet 3: "@Ana estou aqui desde as 08:00 !!"

Depois do pré-processamento:

Tweet 1: "Entrei pelo menos NUM vezes nesse site URL"
Tweet 2: "Não se esqueçam da SICOMP dia DATE"
Tweet 3: "REF estou aqui desde as TIME !!"

5.2 Métricas

Após o treinamento dos modelos, é necessário avalia-los. Para quantificar a eficiência de um modelo são utilizadas métricas. Nesta seção são descritas as métricas utilizadas neste trabalho.

5.2.1 Acurácia

A acurácia é o método mais simples e mais utilizado de avaliação, podendo ser também chamada de taxa de acerto. Ela é calculada pela divisão simples do número de dados associados pelo algoritmo à classe correta, d, sobre o número total de dados testados, T, fornecendo assim a porcentagem de dados que tiveram suas classes corretamente calculadas.

$$A = \frac{d}{T} \tag{5.1}$$

5.2.2 Matriz de confusão

A matriz de confusão é uma métrica que fornece a relação entre as classes reais e classes calculadas dispostas em uma matriz de tamanho $L \times L$, onde L é o número de classes. Segue na tabela 2 um exemplo de uma matriz de confusão para de classificação binária, com uma classe positiva e uma classe negativa.

Tabela 2 – Matriz de confusão.

real/calculado	Positivo	Negativo
Positivo	X	Y
Negativo	Z	W

Fonte: Elaborada pelo autor

Desta maneira, uma matriz de confusão ideal possui os maiores valores localizados na diagonal principal, uma vez que esta relaciona a classe real com a calculada. A matriz de confusão pode ser utilizada para calcular diversas outras métricas, incluindo a acurácia com a relação:

$$A = \frac{X + W}{X + Y + Z + W} \tag{5.2}$$

5.2.3 Pontuação F1

A pontuação é uma métrica que utiliza de outras duas métricas, a precisão e a sensitividade. Utilizando a tabela 2 a precisão é dada pela relação:

$$P = \frac{X}{X + Z} \tag{5.3}$$

e a sensibilidade pela relação:

$$S = \frac{X}{X + Y} \tag{5.4}$$

A pontuação F1 se dá então pela média harmônica das duas medidas:

$$F1 = \frac{2 * P * S}{P + S} \tag{5.5}$$

Quando aplicada à problemas com múltiplas classes é obtido uma pontuação F1 para cada classe, esses valores podem ser unidos em um único valor simplesmente somando todos e dividindo pelo numero de classes ou , em casos com dados desbalanceados, utilizando o numero de dados usados para o teste como peso.

5.3 Modelos

Para este trabalhado foram implementados treze modelos, resultados da combinação de técnicas de extração de características textuais com classificadores, descritos nas seções 2.1 e 2.2 mais o modelo BERT. As técnicas de extração de características são:

- TF-IDF + Unigram (n-gram a nível de palavra de tamanho 1).
- TF-IDF + 4-gram a nível de caractere.
- TF-IDF + 4-gram a nível de palavra.
- TF-IDF + 4-gram a nível de caractere + n-gram a nível de palavra de tamanho 1 á 5.

e os classificadores são:

- Random Forest.
- Naive Bayes.
- Suport Vector Machine.

Para todos foram utilizadas as versões disponibilizadas pela biblioteca *Scikit-learn*.

Finalmente, foi implementado o modelo BERT disponibilizado pela biblioteca *Hugging Face* em sua versão pré-treinada para classificação de sequências. Este último modelo por utilizar a representação de palavras por *Word Embedding* não foi alinhado com nenhuma outra técnica de extração de características.

5.4 Treinamento

Inicialmente, para avaliação dos modelos, todos foram treinados com o conjunto de dados preparados. Para evitar o viés causado pela diferença na quantidade de textos disponíveis de cada autor os modelos foram treinados com aproximadamente 50, 100, 200, 500 e 1000 textos de cada autor.

Considerando a diferença de comportamento de diferentes pessoas na Internet, é conhecido que pessoas publicam em quantidades diferentes, assim conhecer a acurácia dos modelos para diferentes quantidades de dados disponíveis pode ser útil para compreender qual modelo possui o melhor desempenho para a quantidade de dados disponível no problema.

O classificador SVM foi ajustado para utilizar uma função de separação linear com o fator de regularização $C = 1$, o classificador Random Forest utilizou 500 estimadores sem limitações para numero de folhas ou profundidade das árvores, a versão disponibilizada pela

biblioteca *scikit-learn* utiliza o método de amostragem *Bootstrap* para criar cada árvore de decisão, em uma média o método utiliza $2/3$ dos dados totais. o classificador Naive Bayes foi utilizado em sua versão multinomial disponibilizada pela biblioteca *scikit-learn*.

A versão utilizada neste trabalho do modelo BERT foi pré-treinada pela biblioteca *"Hugging Face"* em 16 chips de TPU com um milhão de passos para lotes de tamanho 256 e otimizada utilizando o método estocástico ADAM, foi treinada com taxa de aprendizado $2 * 10^{-5}$.

Para todos os treinamentos os dados foram separados em proporção de 70% para treinamento e 30% para avaliação. As matrizes de confusão foram geradas unicamente para o conjunto de características que leva ao melhor desempenho do classificador.

5.5 Testes e avaliação

Os modelos foram avaliados utilizando os 30% dos dados separados. As Figuras 8, 10, 9 e 11 apresentam as curvas de acurácia obtidas pelos classificadores *Naive Bayes*, *Random Forest*, SVM e BERT, respectivamente, para cada um dos quatro descritores (*unigram*, *char 4-gram*, *word 4-gram* e *todos os n-grams*), para os cinco conjuntos de dados cosntituídos por aproximadamente 50, 100, 200, 500 e 1000 *tweets* por autor.

Todos os classificadores obtiveram os melhores desempenhos nos conjuntos de dados contendo a maior quantidade de *tweets* por autor. Dentre os classifiadores clássicos, dois tiveram o melhor desempenho com todas as técnicas de extração de características e o classificador *Random Forest* obteve seu melhor desempenho com somente o *4-gram* a nível de caractere.

Dentre os classificadores clássicos, o SVM que utiliza todas as técnicas de extração de características obteve a melhor acurácia de 70.77%, porém abaixo da acurácia obtida pelo BERT que, após o procedimento de *fine-tuning*, obteve acurácia de 75.78%.

As Figuras 12, 13, 14 e 15 apresentam as matrizes de confusão para os melhores modelos de cada classificador. Nelas é possível observar que algumas autorias apresentam maior dificuldade de serem corretamente atribuídas, como por exemplo o modelo que utiliza o classificador Naive Bayes teve dificuldade em atribuir corretamente os textos da classe 11, atribuindo corretamente apenas 102 textos e atribuindo equivocadamente 84 textos à classe 9.

Devido à diferenças de acurácias para as diferentes classes evidenciadas pela matriz de confusão foi decidido utilizar a métrica de pontuação F1. As figuras 8, 10, 9 e 19 representam a pontuação F1 de cada classificador em seu modelo de melhor desempenho para cada um dos diferentes autores, como o numero de amostras para testagem para cada autor é muito semelhante foi escolhido utilizar uma média aritmética para o calculo da pontuação F1 em um valor único apresentado na tabela 3.

Com estes valores é possível observar que o modelo BERT atinge os melhores valores

Tabela 3 – Pontuação F1.

real/Classificador	Pontuação
Naive Bayes	0.6408
Random Forest	0.6261
SVM	0.7099
BERT	0.7590

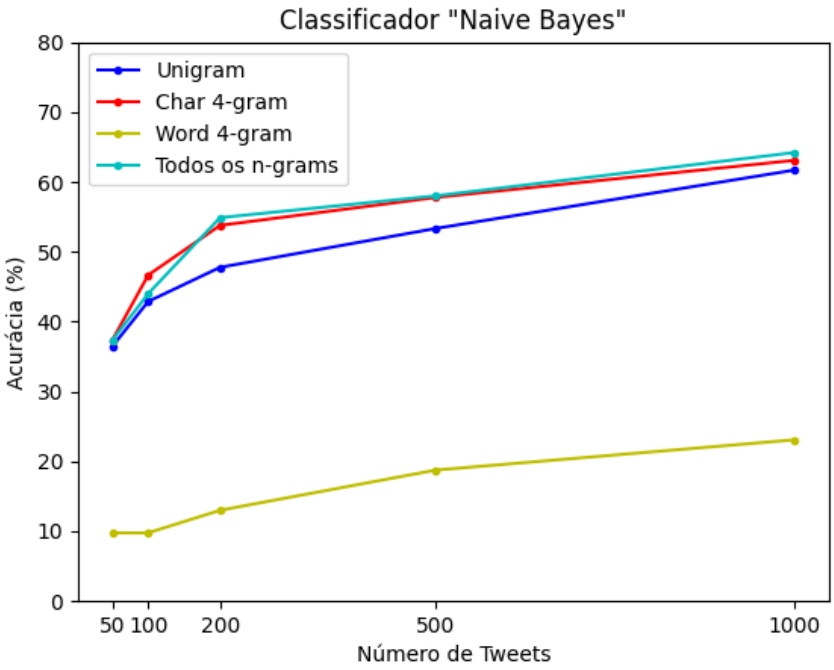
Fonte: Elaborada pelo autor

para acurácia e pontuação F1, enquanto somente entre os modelos clássicos o que utiliza do classificador SVM também atinge os melhores valores para ambas métricas.

É interessante observar como a acurácia de todos os modelos que utilizam tecnologias clásssicas utilizando somente do *4-gram* á nível de palavra é muito baixa, isso provavelmente se dá pela limitação de caracteres imposta pelo *Twitter* em suas postagems, que normalmente são compostas por um numero muito pequeno de palavras, tornando improvavel a repetição dos *4-grams*. Essa limitação pode ser também responsável pela baixa subida na acurácia quando são adicionados os *n-grams* de palavra ao de caractere, outro fator à se observar nesse caso é a escalada no numero de dados, que pode levar á um problema de *overfitting*.

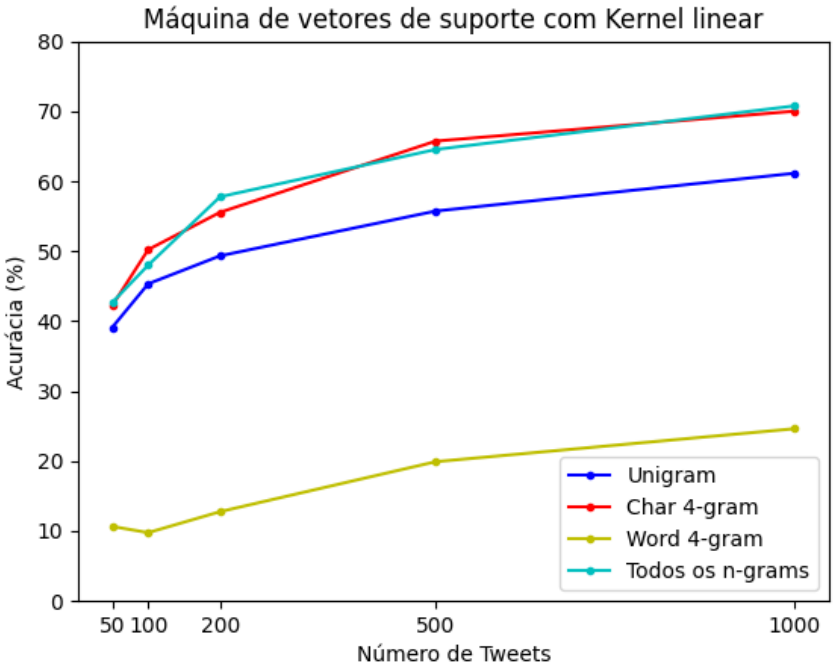
Enquanto é possível discutir e teorizar em relação aos modelos classicos, é muito difícil conseguir uma intuição em relação ao modelo BERT. É possível imaginar que os modelos classícos busquem padrões nas frequências alimentadas aos classificadores, já o modelo *deep learning* não permite uma vizualização clara do que está sendo feito, é muito dificil até mesmo compreeender as características sendo codificadas no *word embedding* criado por este.

Figura 8 – Acurácias obtidas pelo classificador Naive Bayes.



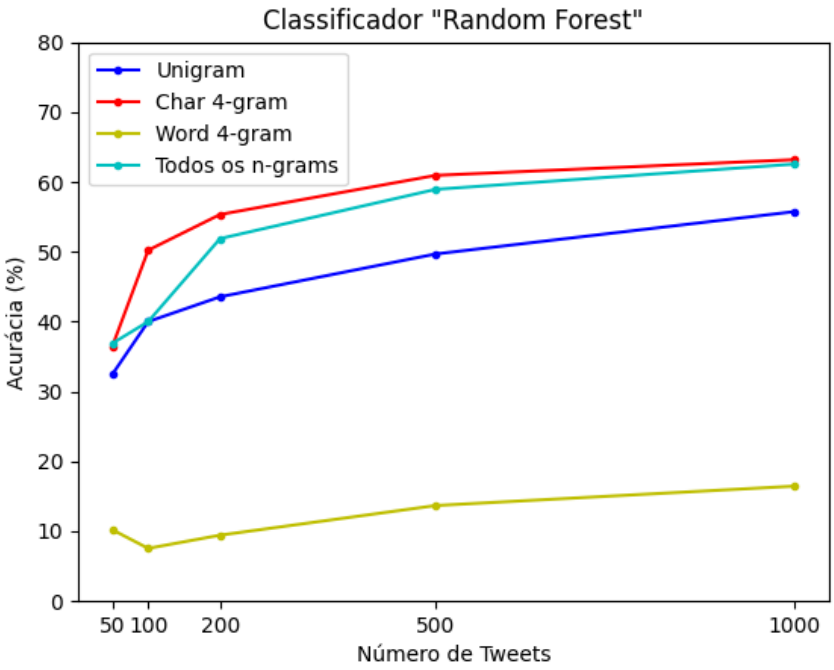
Fonte: Elaborada pelo autor.

Figura 9 – Acurácias obtidas pelo classificador SVM.



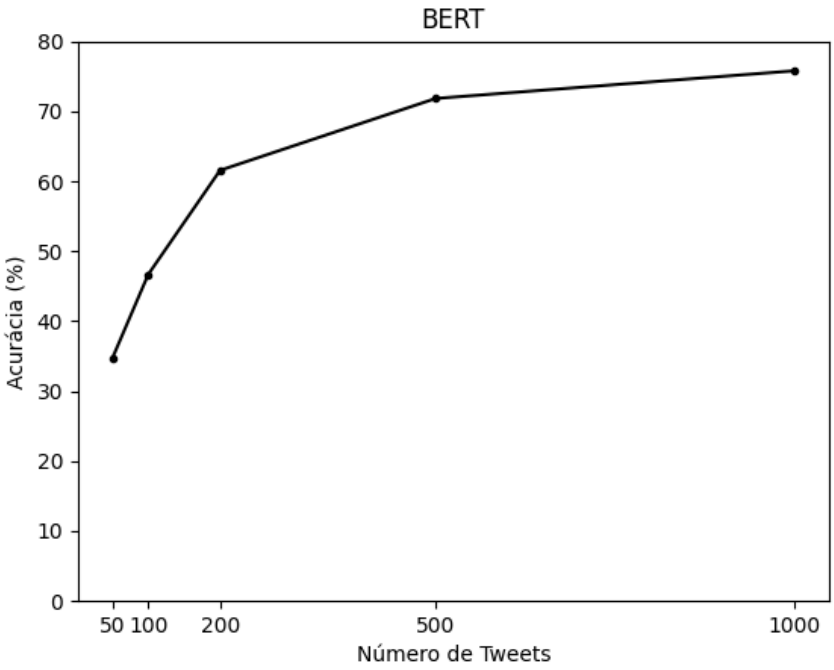
Fonte: Elaborada pelo autor.

Figura 10 – Acurácias obtidas pelo classificador Random Forest.



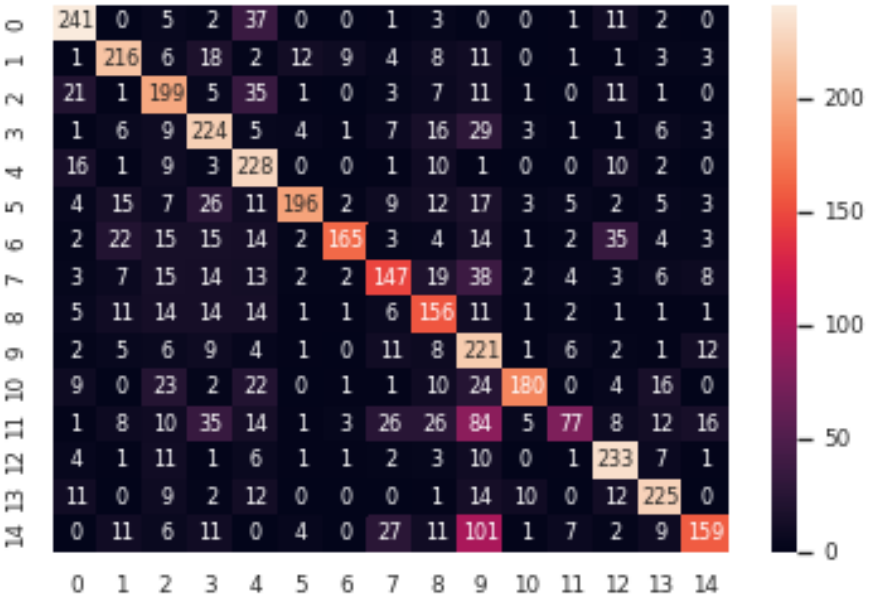
Fonte: Elaborada pelo autor.

Figura 11 – Acurácias obtidas pelo BERT.



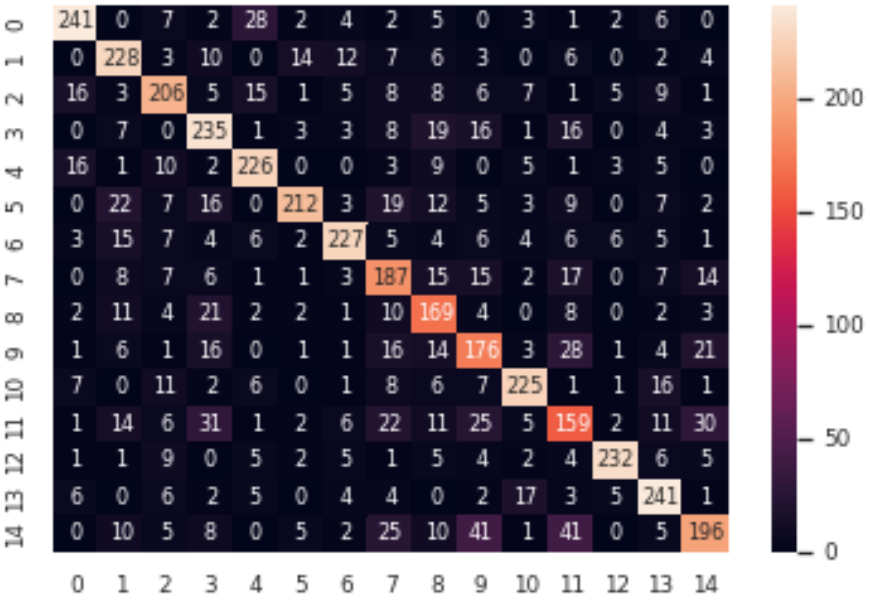
Fonte: Elaborada pelo autor.

Figura 12 – Matriz de confusão do modelo Naive Bayes + TF-IDF + 4-gram a nível de caractere + n-gram a nível de palavra de tamanho 1 á 5 treinado com 700 textos por autor e testado com 300 textos por autor.



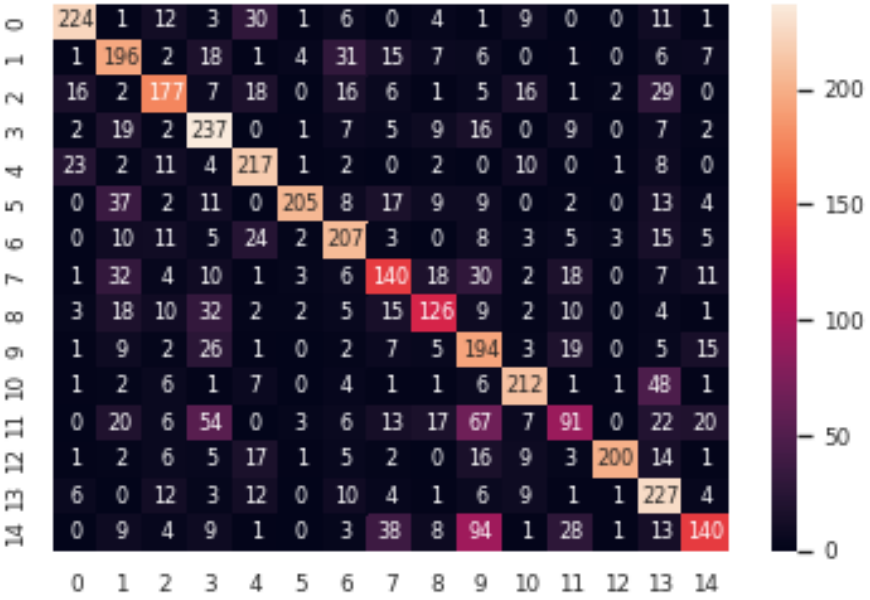
Fonte: Elaborada pelo autor.

Figura 13 – Matriz de confusão do modelo SVM + TF-IDF + 4-gram a nível de caractere + n-gram a nível de palavra de tamanho 1 á 5 treinado com 700 textos por autor e testado com 300 textos por autor.



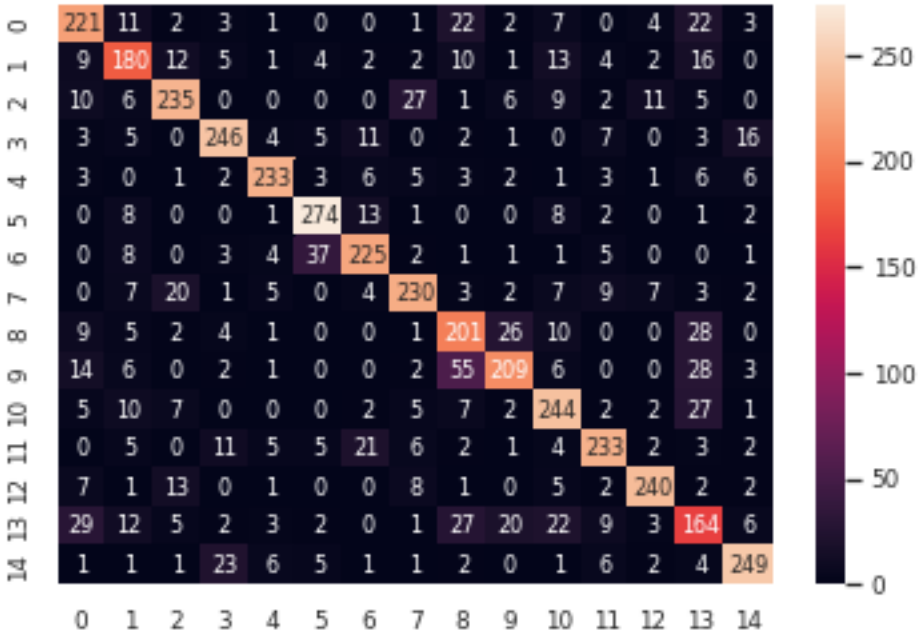
Fonte: Elaborada pelo autor.

Figura 14 – Matriz de confusão do modelo Random Forest + TF-IDF + 4-gram a nível de caractere treinado com 700 textos por autor e testado com 300 textos por autor.



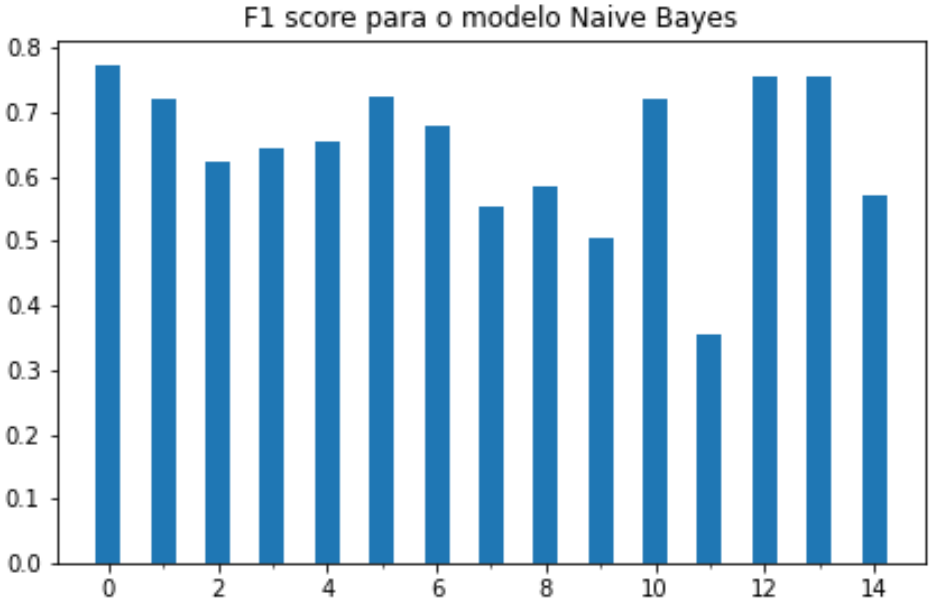
Fonte: Elaborada pelo autor.

Figura 15 – Matriz de confusão do modelo BERT treinado com 700 textos por autor e testado com 300 textos por autor.



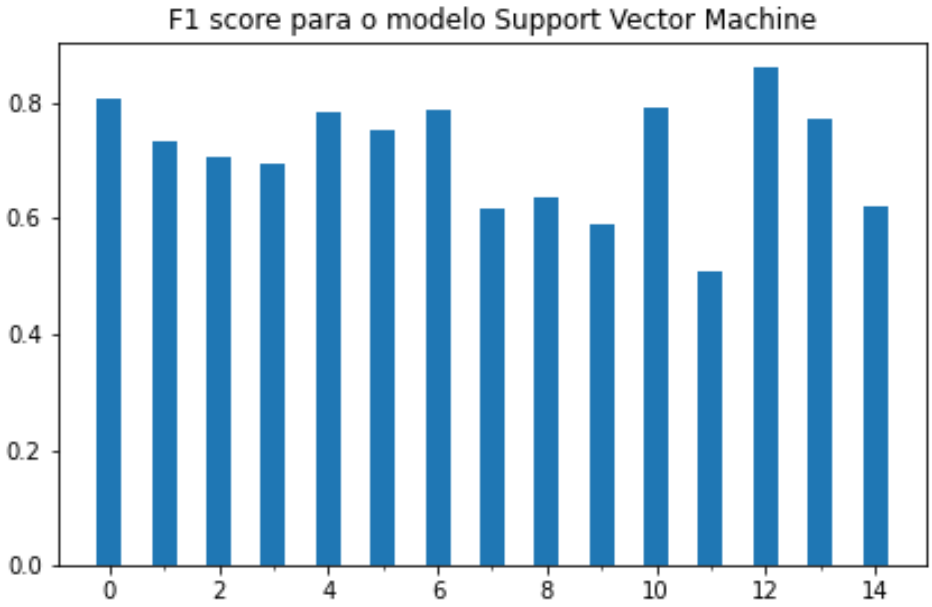
Fonte: Elaborada pelo autor.

Figura 16 – Pontuação F1 por classe do modelo Naive Bayes + TF-IDF + 4-gram a nível de caractere + n-gram a nível de palavra de tamanho 1 á 5 treinado com 700 textos por autor e testado com 300 textos por autor.



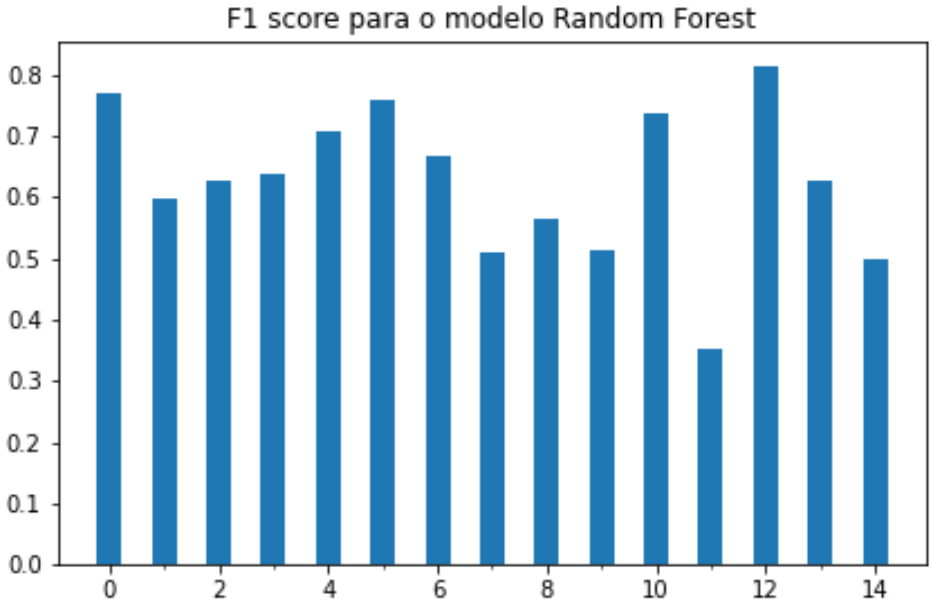
Fonte: Elaborada pelo autor.

Figura 17 – Pontuação F1 por classe do modelo SVM + TF-IDF + 4-gram a nível de caractere + n-gram a nível de palavra de tamanho 1 á 5 treinado com 700 textos por autor e testado com 300 textos por autor.



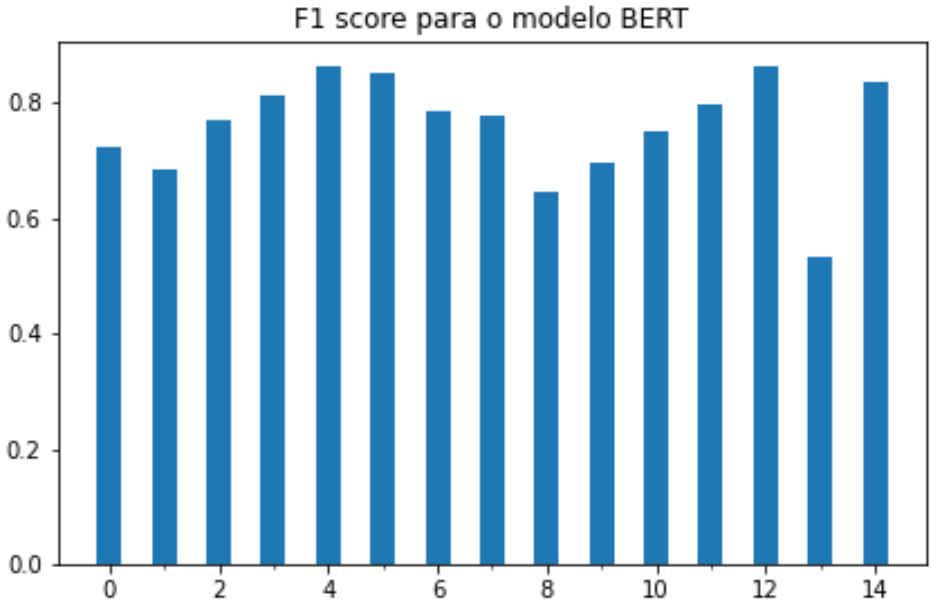
Fonte: Elaborada pelo autor.

Figura 18 – Pontuação F1 por classe do modelo Random Forest + TF-IDF + 4-gram a nível de caractere treinado com 700 textos por autor e testado com 300 textos por autor.



Fonte: Elaborada pelo autor.

Figura 19 – Pontuação F1 por classe do modelo BERT treinado com 700 textos por autor e testado com 300 textos por autor.



Fonte: Elaborada pelo autor.

6 Conclusão

Neste trabalho buscou-se compreender a tarefa de atribuição de autoria, testando diferentes modelos que explorando técnicas, classificadores e um modelos de estado da arte de aprendizado de máquina aplicados à um corpus formado por dados retirados do *Twitter*. A tarefa vem associada a diversas questões atuais como a divulgação de noíicias falsas, propagação de discurso de ódio, *cyberbulling* e outras que têm sido objeto de interesse de diversas pesquisas nos últimos anos.

Foi necessário coletar dados para complementar o *corpus* disponibilizado *online*, exigindo a utilização de uma API específica. Uma dificuldade apresentada pela tarefa foi a extração de características de textos formados por uma quantidade muito limitada de palavras, exigindo um passo de pré-processamento e um estudo de diferentes técnicas de extração de características de linguagem natural, como o TF-IDF e *n-grams*, que estão classicamente associadas ao problema de atribuição de autoria.

Além de técnicas de extração de características foi feito um estudo em relação aos classificadores utilizados em problemas de processamento de linguagem natural, foi escolhido implementar os classificadores de *Naive Bayes*, *Random Forest* e *Support Vector Machine* e associa-los as técnicas previamente selecionadas.

Após a implementação dos modelos que utilizam de tecnologias clássicas, decidiu-se implementar um modelo mais recente que lida com processamento de linguagem natural, o modelo escolhido foi o BERT, um modelo que utiliza da tecnologia de *Transformers* que tem sido extremamente influente na área de aprendizado de máquina.

Os modelos então foram treinados e testados para diferentes quantidades de dados para cada autor, o que revelou que a baixa quantidade de textos disponíveis afeta severamente o desempenho dos modelos, mas que na melhor condição o Modelo BERT se mostrou mais eficaz na tarefa atingindo acurácia de 75.78%.

Durante o desenvolvimento foi possível estudar diferentes modelos e tecnologias da área de processamento de linguagem natural, sendo possível observar diferentes abordagens para o mesmo problema. Ainda, foram também estudadas diferentes bibliotecas e tecnologias fundamentais para o desenvolvimento e estudos de inteligências artificiais.

Como possibilidade para trabalhos futuros existe a melhoria no *corpus*, realizar o teste com uma maior quantidade de autores ou experimentar com diferentes línguas. É válido também que se utilize de outros métodos de avaliação como avaliação cruzada para garantir maior confiabilidade aos testes.

Em relação aos modelos um processo mais longo de busca de hiperparâmetros poderia

garantir um melhor desempenho. Para o modelo BERT uma alternativa que já vem sendo explorada em outros trabalhos é uma análise de explicabilidade, a obscuridade em relação aos processos do modelo pode ser uma barreira na sua utilização em ambientes como o criminal que pode utilizar da tarefa de atribuição de autoria.

Referências

ABORISADE, O.; ANWAR, M. Classification for authorship of tweets by comparing logistic regression and naive bayes classifiers. In: IEEE. *2018 IEEE International Conference on Information Reuse and Integration (IRI)*. [S.l.], 2018. p. 269–276.

BACCIU, A.; MORGIA, M. L.; MEI, A.; NEMMI, E. N.; NERI, V.; STEFA, J. Cross-domain authorship attribution combining instance based and profile-based features. In: *CLEF (Working Notes)*. [S.l.: s.n.], 2019.

CHENG, A. *The Agency*. 2015. Https://www.nytimes.com/2015/06/07/magazine/the-agency.html Acessado em 09/02/2022.

CHOWDHURY, G. Natural language processing. *Annual Review of Information Science and Technology*, v. 37, n. 1, p. 51–89, jan. 2003. ISSN 0066-4200.

DEVLIN, J.; CHANG, M.-W.; LEE, K.; TOUTANOVA, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

FABIEN, M.; VILLATORO-TELLO, E.; MOTLICEK, P.; PARIDA, S. Bertaa: Bert fine-tuning for authorship attribution. In: *Proceedings of the 17th International Conference on Natural Language Processing (ICON)*. [S.l.: s.n.], 2020. p. 127–137.

FRANTZESKOU, G.; STAMATATOS, E.; GRITZALIS, S.; KATSIKAS, S. Source code author identification based on n-gram author profiles. In: SPRINGER. *IFIP International Conference on Artificial Intelligence Applications and Innovations*. [S.l.], 2006. p. 508–515.

HERN, A. *Facebook and Twitter threatened with sanctions in UK 'fake news' inquiry*. 2017. Https://www.theguardian.com/media/2017/dec/28/facebook-and-twitter-threatened-with-sanctions-in-uk-fake-news-inquiry Acessado em 09/02/2022.

JAGANNATH, V. *Random Forest template for Tibco Spotfire®*. 2020. Https://community.tibco.com/wiki/random-forest-template-tibco-spotfire Acessado em 14/03/2022.

KRAMER, O. Scikit-learn. In: *Machine learning for evolution strategies*. [S.l.]: Springer, 2016. p. 45–53.

LEVY, O.; GOLDBERG, Y. Dependency-based word embeddings. In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. [S.l.: s.n.], 2014. p. 302–308.

LORENA, A. C.; CARVALHO, A. C. D. Uma introdução às support vector machines. *Revista de Informática Teórica e Aplicada*, v. 14, n. 2, p. 43–67, 2007.

MYLES, A. J.; FEUDALE, R. N.; LIU, Y.; WOODY, N. A.; BROWN, S. D. An introduction to decision tree modeling. *Journal of Chemometrics: A Journal of the Chemometrics Society*, Wiley Online Library, v. 18, n. 6, p. 275–285, 2004.

NUMPY. *NumPy user guide*. c2008. Https://numpy.org/doc/stable/user/index.htmluser Acessado em 09/02/2022.

PANDAS. *User Guide*. c2008. Https://pandas.pydata.org/docs/user_guide/index.html Acessado em 09/02/2022.

PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V. et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, JMLR. org, v. 12, p. 2825–2830, 2011.

RAMOS, J. et al. Using tf-idf to determine word relevance in document queries. In: CITESEER. *Proceedings of the first instructional conference on machine learning*. [S.l.], 2003. v. 242, n. 1, p. 29–48.

ROCHA, A.; SCHEIRER, W. J.; FORSTALL, C. W.; CAVALCANTE, T.; THEOPHILO, A.; SHEN, B.; CARVALHO, A. R.; STAMATATOS, E. Authorship attribution for social media forensics. *IEEE transactions on information forensics and security*, IEEE, v. 12, n. 1, p. 5–33, 2016.

SCHWARTZ, R.; TSUR, O.; RAPPOPORT, A.; KOPPEL, M. Authorship attribution of micro-messages. In: *Proceedings of the 2013 Conference on empirical methods in natural language processing*. [S.l.: s.n.], 2013. p. 1880–1891.

TIMBERG, C. *New whistleblower claims Facebook allowed hate, illegal activity to go unchecked*. 2021. Https://www.washingtonpost.com/technology/2021/10/22/facebook-new-whistleblower-complaint/ Acessado em 09/02/2022.

TWITTER. *Docs*. c2022. Https://developer.twitter.com/en/docs Acessado em 09/02/2022.

VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, Ł.; POLOSUKHIN, I. Attention is all you need. *Advances in neural information processing systems*, v. 30, 2017.

WOLF, T.; DEBUT, L.; SANH, V.; CHAUMOND, J.; DELANGUE, C.; MOI, A.; CISTAC, P.; RAULT, T.; LOUF, R.; FUNTOWICZ, M. et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

WU, Y.; SCHUSTER, M.; CHEN, Z.; LE, Q. V.; NOROUZI, M.; MACHEREY, W.; KRIKUN, M.; CAO, Y.; GAO, Q.; MACHEREY, K. et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

XU, B.; GUO, X.; YE, Y.; CHENG, J. An improved random forest classifier for text categorization. *J. Comput.*, Citeseer, v. 7, n. 12, p. 2913–2920, 2012.

ZHANG, H. The optimality of naive bayes. *Aa*, v. 1, n. 2, p. 3, 2004.