UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

CAIO MAXXIMUS PEREIRA SOARES

VERIFICAÇÃO DE AUTORIA EM MENSAGENS DE TEXTO UTILIZANDO GRAFOS E APRENDIZAGEM DE MÁQUINA

CAMPINA GRANDE - PB

2023

CAIO MAXXIMUS PEREIRA SOARES

VERIFICAÇÃO DE AUTORIA EM MENSAGENS DE TEXTO UTILIZANDO GRAFOS E APRENDIZAGEM DE MÁQUINA

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador : Professor Dr. Herman Martins Gomes

CAMPINA GRANDE - PB

2023

CAIO MAXXIMUS PEREIRA SOARES

VERIFICAÇÃO DE AUTORIA EM MENSAGENS DE TEXTO UTILIZANDO GRAFOS E APRENDIZAGEM DE MÁQUINA

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA:

Professor Dr. Herman Martins Gomes

Orientador – UASC/CEEI/UFCG

Professora Dr. Patrícia Duarte de Lima Machado

Examinador – UASC/CEEI/UFCG

Francisco Vilar Brasileiro

Professor da Disciplina TCC – UASC/CEEI/UFCG

Trabalho aprovado em: 28 de JUNHO de 2023.

CAMPINA GRANDE - PB

RESUMO

A busca por extração de características em textos é uma área de interesse em aprendizagem de máquina devido às inúmeras possibilidades relacionadas, dentre elas a verificação de autoria é um tema relevante por suas aplicações e elevada complexidade. Neste contexto, o presente artigo faz uso de dados provenientes de mensagens de chat de servidores Discord com o propósito de verificar automaticamente a autoria das mensagens mediante um treinamento supervisionado. O processo inicia-se com um pré-processamento que busca reduzir ruído e viés nos dados, para então explorar a capacidade do modelo de aprendizagem em generalizar ao encontrar textos desconhecidos e defini-los como de sua autoria ou não. Desta forma são utilizados grafos como extratores de características em mensagens de texto, utilizando de redes neurais artificiais como modelos de aprendizagem de máquina para classificá-las . Palavras se tornam nós, e suas arestas capturam a intensidade referente à distância dos termos na frase, resultando na construção de um grafo que representa o vocabulário de um indivíduo e que tem como objetivo captar características relevantes no texto.Obtidas boas acurácias para o verdadeiros positivos e para os verdadeiros negativos ao se ajustar o limiar de ativação, os modelos conseguem alcançar resultados satisfatórios com reduzido custo de treinamento, permitindo uma facilidade maior para exploração de novos parâmetros.

AUTHORSHIP VERIFICATION IN TEXT MESSAGES USING GRAPHS AND MACHINE LEARNING

ABSTRACT

The search for feature extraction in texts is an area of interest in machine learning due to its numerous related possibilities, among them authorship verification is a relevant topic due to its applications and high complexity. In this context, this article uses data from Discord server chat messages with the purpose of automatically verifying the authorship of messages through supervised training. The process begins with preprocessing that aims to reduce noise and bias in the data, and then explores the learning model's ability to generalize by identifying unknown texts and classifying them as either authored or not. Graphs are used as feature extractors in text messages, leveraging artificial neural networks as machine learning models for classification. Words become nodes, and their edges capture the intensity related to the distance between terms in the sentence, resulting in the construction of a graph that represents an individual's vocabulary and aims to capture relevant characteristics in the text. By achieving good accuracies for true positives and true negatives when adjusting the activation threshold, the models can achieve satisfactory results with reduced training cost, allowing for easier exploration of new parameters.

Verificação de autoria em mensagens de texto utilizando grafos e aprendizagem de máquina

Caio Maxximus Pereira Soares
caio.soares@ccc.ufcg.edu.br
Universidade Federal de Campina Grande
Campina Grande, Paraíba, Brasil.

Orientador: Herman Martins Gomes
hmg@computacao.ufcg.edu.br
Universidade Federal de Campina Grande
Campina Grande, Paraíba, Brasil.

RESUMO

A busca por extração de características em textos é uma área de interesse em aprendizagem de máquina devido às inúmeras possibilidades relacionadas, dentre elas a verificação de autoria é um tema relevante por suas aplicações e elevada complexidade. Neste contexto, o presente artigo faz uso de dados provenientes de mensagens de chat de servidores Discord com o propósito de verificar automaticamente a autoria das mensagens mediante um treinamento supervisionado. O processo inicia-se com um pré-processamento que busca reduzir ruído e viés nos dados, para então explorar a capacidade do modelo de aprendizagem em generalizar ao encontrar textos desconhecidos e defini-los como de sua autoria ou não. Desta forma são utilizados grafos como extratores de características em mensagens de texto, utilizando de redes neurais artificiais como modelos de aprendizagem de máquina para classificá-las . Palavras se tornam nós, e suas arestas capturam a intensidade referente à distância dos termos na frase, resultando na construção de um grafo que representa o vocabulário de um indivíduo e que tem como objetivo captar características relevantes no texto.Obtidas boas acurácias para o verdadeiros positivos e para os verdadeiros negativos ao se ajustar o limiar de ativação, os modelos conseguem alcançar resultados satisfatórios com reduzido custo de treinamento, permitindo uma facilidade maior para exploração de novos parâmetros.

KEYWORDS

Grafos, machine learning, autoria, mensagens de texto

1. INTRODUÇÃO

Em um universo de serviços quase ou totalmente digitais, confiabilidade é um tema chave para garantir segurança na prestação de serviços [1]. Dados confidenciais podem ser tomados por meio de ataques de entidades que se passam por partes conhecidas e confiáveis de uma relação, através de credenciais roubadas [2] de pessoas tais como gerentes, secretários ou funcionários comuns.

Em posse dessas credenciais, é possível então extrair dados críticos de outros membros da organização. Como exemplo, um invasor rouba dados de credenciais de e-mail do funcionário de uma instituição e com esta conta, pode então solicitar a algum outro membro ocupando posição superior na empresa, dados aparentemente não tão importantes, porém privados, sobre clientes. Finalmente, de posse desta informação, o indivíduo invasor comete outros ataques ou mesmo golpes nos próprios clientes. O objetivo então é capturar a fraude no processo de comunicação das partes, levantando uma suspeita que deve ser então checada pelas entidades responsáveis.

Pesquisas na área de reconhecimento de autoria em textos utilizando Aprendizagem de Máquina não são algo recente[3] nas áreas de processamento de linguagem natural e até mesmo de serviços voltados para inteligência [4], contudo é ainda um desafio entender os parâmetros adequados a serem analisados, principalmente quando se refere a textos de menor tamanho, que são uma realidade nas formas mais comuns de comunicação virtual informal ou mesmo no âmbito de uma organização. O emprego de grafos em verificação de autoria é uma estratégia conhecida [4,5], transformando palavras em nós e explorando as suas conexões para extrair dados sobre seus relacionamentos, em que diversas técnicas são exploradas, como por exemplo, a mineração de dados para encontrar subgrafos[6]. Neste artigo a abordagem baseia-se em uma implementação mais simples, utilizando métricas distintas para medir a similaridade entre grafos que então, alimentam um classificador que deve aprender a classificar a autoria de uma mensagem como verdadeira ou falsa.

O restante do documento está estruturado da seguinte forma: na Seção 2 é descrita a fundamentação teórica do modelo, na Seção 3 a metodologia e arquitetura da aplicação, na Seção 4 são descritos os experimentos e os resultados; na seguinte uma descrição da experiência obtida durante a pesquisa e por último um relato pessoal dos benefícios da pesquisa realizada.

2. FUNDAMENTAÇÃO TEÓRICA

2.1 Grafos como fonte de informação

Um grafo é uma estrutura de dados composta por um conjunto de nós (ou vértices) e um conjunto de arestas que conectam esses nós. Os nós representam entidades individuais no grafo, como objetos, ou conceitos abstratos. As arestas são as conexões entre nós e podem ser direcionadas (quando possuem uma direção específica) ou não direcionadas (quando não têm uma direção definida). As arestas podem ter pesos ou custos associados, representando medidas como distância, tempo, capacidade, entre outros.Os grafos têm diversas características e propriedades, como grau (o número de arestas incidentes em um nó), caminhos

(sequências de nós conectados por arestas), ciclos (caminhos fechados formando circuitos), conectividade (existência de caminhos entre todos os pares de nós) e muitas outras [7].

2.2 Medidas de distância

Medidas de distância [7,8,9] são uma parte importante dos extratores de características do modelo, operando sobre as matrizes de adjacência do grafo que representa a entrada e o subgrafo obtido do modelo.

2.2.1 Distância Espectral

A distância espectral é uma métrica de distância utilizada para medir a diferença entre dois objetos com base em suas características espectrais. É comumente usada em processamento de sinais, processamento de imagens e reconhecimento de padrões. A distância espectral é calculada por intermédio da comparação das representações espectrais dos objetos, considerando suas frequências e amplitudes.

2.2.2 Distância Euclidiana

A distância euclidiana é uma das métricas de distância mais comuns. Ela é aplicada em espaços euclidianos para calcular a distância entre dois pontos, levando em consideração as coordenadas desses pontos. A distância euclidiana é calculada utilizando o teorema de Pitágoras, que determina a distância entre dois pontos em um plano ou espaço tridimensional.

2.2.3 Distância de Hamming

A distância de Hamming é uma métrica de distância usada para comparar strings ou sequências de elementos de mesmo comprimento. Ela mede a diferença entre duas sequências, contando o número de posições em que os elementos correspondentes são diferentes. A distância de Hamming é comumente usada em ciência da computação, codificação de dados e correção de erros em comunicações digitais.

2.2.4. Distância de Pearson

A distância de Pearson é uma métrica de distância que mede a correlação linear entre duas variáveis. Ela é baseada no coeficiente de correlação de Pearson, que indica a relação linear entre duas variáveis contínuas. A distância de Pearson é calculada como a diferença entre 1 e o coeficiente de correlação de Pearson, variando de 0 (correlação perfeita) a 2 (ausência de correlação).

2.2.5 Distância de Jaccard

É uma medida de dissimilaridade entre dois conjuntos. É calculada como 1 menos a razão entre o tamanho da interseção dos conjuntos e o tamanho da união dos conjuntos.

2.2.6 Distância de Manhattan

Também conhecida como distância de cidade ou distância L1, é a soma das diferenças absolutas entre as coordenadas dos pontos. É chamada de "Manhattan" porque se imagina que uma pessoa se mova apenas em ruas perpendiculares para ir de um ponto a outro, como nos quarteirões de Manhattan, Nova York.

3. METODOLOGIA

3.1 A Matriz de Adjacências - capturando informação temporal

Em um grafo, as conexões entre os nós representam associações, e estas podem ser bidirecionais e possuírem diferentes pesos, caracterizando assim diferentes intensidades . O objetivo é então utilizar essas características para transformar uma coleção de textos de um usuário em um grafo dirigido superconectado, utilizando diferentes métricas para calcular semelhanças, observando como as características extraídas são capazes de gerar representações únicas dos textos de um rótulo (usuário) alvo.

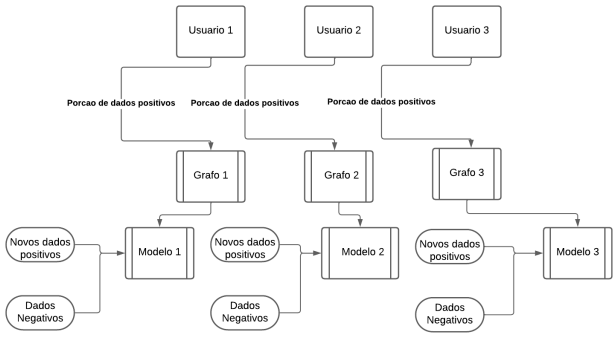


Figura 1.Esquema de construção dos modelos, utilizando o grafo como base para o cálculo das métricas.

A matriz de adjacências é então um representação bi-dimensional de um grafo, onde cada linha da matriz quadrada representa cada nó do grafo e cada coluna uma aresta que conecta este nó a outro. Sendo A uma matriz de adjacências, A[i][j] = x representa uma conexão entre o nó "i" e o nó "j" com intensidade x, que pode ser vista também como uma ausência de conexão caso x = 0, destaca-se que A[j][i] = y é também verdade, sendo x e y $\in R_+$.

A Figura 1 ilustra a implementação da solução, onde cada usuário possui um modelo próprio para predizer se uma mensagem é de sua autoria ou não.

Deste modo para cada mensagem dentro da base de dados do rótulo alvo (dados positivos), a criação do grafo que representa um usuário alvo se dá seguinte forma:

1) Para cada palavra encontrada no texto, utiliza-se de um contador que mapeia cada termo para um número inteiro (sua representação como um nó). Ao se encontrar um termo se verifica se este existe no dicionário, se sim, utiliza-se este valor como o valor do nó, se não, utiliza-se o valor do contador, adicionando o termo no dicionário, e incrementando o contador posteriormente.
2) Cria-se então uma conexão com intensidade proporcional à distância entre o primeiro termo e os seguintes, caso esta ainda não exista, ao contrário, é então feita uma soma da intensidade já existente com a intensidade atual.
3) A intensidade é calculada a partir de uma simples relação que utiliza da distância dos termos na frase, sendo "i" a posição do primeiro e "j" a posição do termo seguinte, a intensidade se dá da seguinte forma:

$$intensidade = 1 / (j - i)$$

Esta é então calculada entre um termo e todos os termos seguintes, como representado na Figura 2.

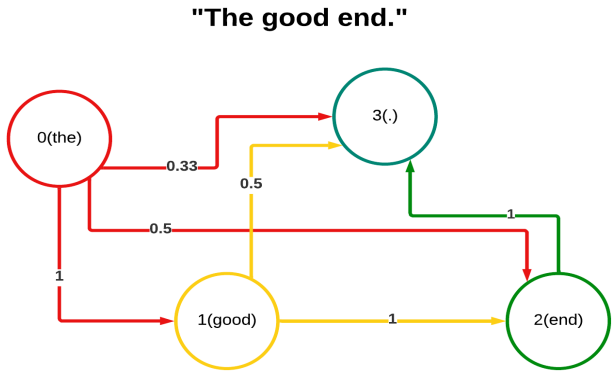


Figura 2. Exemplo de intensidades das arestas.

Então o grafo será representado como uma matriz de adjacências como na Tabela 1.

	0	1	2	3
0	0	1	0.5	0.33
1	0	0	1	0.5
2	0	0	0	1
3	0	0	0	0

Tabela 1. Matriz de adjacências correspondente à Figura 1.

Um tratamento adicional é feito sobre os dados, que consiste na inserção de caracteres especiais para representar o começo ($) e fim (#) da mensagem,por exemplo dessa forma, a mensagem : “The good end.” , se torna: “$ The good end . #” após o pré processamento.

No cenário de implementação real do modelo, o nó de valor “0” é reservado para termos desconhecidos do dicionário do usuário, que é construído apenas com textos oriundos da parcela dos dados positivos, destinados a essa tarefa. Então um termo desconhecido é mapeado sempre para o nó zero. Importante notar que, ao se construir a matriz de adjacências, ocorre uma normalização da mesma, fundamentada nos valores de mínimo e máximo, agrupando-os entre [0,1], e este procedimento é também aplicado à matriz gerada por cada entrada que o modelo recebe,seja para previsão ou treinamento.

3.2 Rede Neural

Ao se utilizar das diferentes métricas para calcular as similaridades entre os subgrafos da entrada e do grafo base do rótulo alvo, estas medidas resultam então um pequeno vetor de características constituindo a natureza de uma comparação que se utiliza de diferentes aspectos para gerar uma extração de informação mais complexa.

Utilizou-se então de uma rede neural de pequena capacidade para processar esse vetor de características e gerar uma saída que opera no limiar [0,1], onde 0 é certeza que o rótulo é falso , e 1 a que ele é verdadeiro. Uma fronteira de decisão é então utilizada para definir como a saída gerada pela rede será classificada.

O trecho de código do modelo neural utilizado para a versão base do modelo é apresentado a seguir:

```
X = self.comparador.call(X)
model = self.model
model.add(Input(shape=6,))
model.add(Dense(30 , activation = "relu"))
model.add(Dense(15, activation = "relu"))
model.add(Dense(1, activation = "sigmoid"))
model.compile(optimizer=Adam(lr=0.1),
loss='binary_crossentropy'                ,
metrics=['accuracy'])
model.fit(X, y, epochs=100, batch_size=6,
verbose = True)
```

No código anterior, a entrada de tamanho seis, corresponde à quantidade de métricas geradas para cada exemplo de treinamento a partir do comparador, que é responsável por armazenar o grafo do modelo e transformar os dados de mensagens para as medidas de similaridade .

3.3 Base de Dados

A base de dados [9] utilizada é uma pequena fração de uma coleção de mensagens do software do chat Discord[1].Esta se encontra disponível através da plataforma Kaggle[2], para ser baixada ou utilizada diretamente em seus ambientes. As mensagens são rotuladas com o *nickname* do seu autor e o pré processamento se deu através da exclusão de ruído nos dados, remoção de stopwords, e de termos como: “\n” ou “\\\n”, assim como de palavras que começavam com “@” como “@algumnickname” para prevenir algum viés, também números ou símbolos (“ª%^º~*”), foram removidos. Filtraram-se mensagens com menos de 10 e com mais de 20 termos, almejando capturar características em textos de tamanho reduzido que são mais comuns na base de dados.

[1] https://discord.com

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

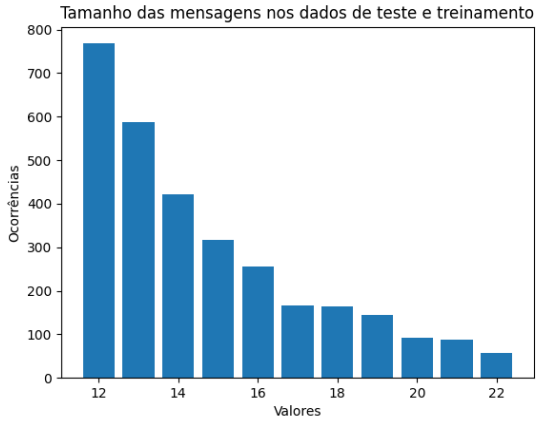


Figura 3. Histograma dos tamanhos das mensagens na porção da base de dados utilizada.

Nota-se que os tamanhos apresentados na Figura 3, estão entre 12 e 22, que representam a soma do tamanho da mensagem pré processada original com a adição dos termos de início : "$" ,e fim: "#" de mensagem.

3.4 Subgrafos Como Representantes Locais

Posteriormente à criação da matriz de adjacência do modelo, temos agora uma representação das interações entre os termos da mensagem do rótulo alvo mapeada para um grafo. O passo seguinte busca como utilizar isto para extrair os parâmetros necessários para gerar informações para o modelo.

A resposta então está nos subgrafos da matriz de adjacências; as métricas, já mencionadas anteriormente serão então empregadas sobre uma entrada que é transformada em um grafo, de bem menor escala, seguindo o mesmo algoritmo utilizado para a construção do grafo do usuário alvo porém neste momento não existe criação de novos nós, apenas o mapeamento dos termos da entrada para os nós correspondentes a partir do dicionário do usuário. Como dito anteriormente, termos inexistentes no dicionário são mapeados para o nó reservado "0".

Após a criação da matriz da entrada,é criada uma nova matriz de mesmo tamanho, porém com os valores recuperados da matriz de adjacências do usuário, um subgrafo, que carrega as informações para aqueles conjuntos de nós da entrada, dos pesos das arestas.

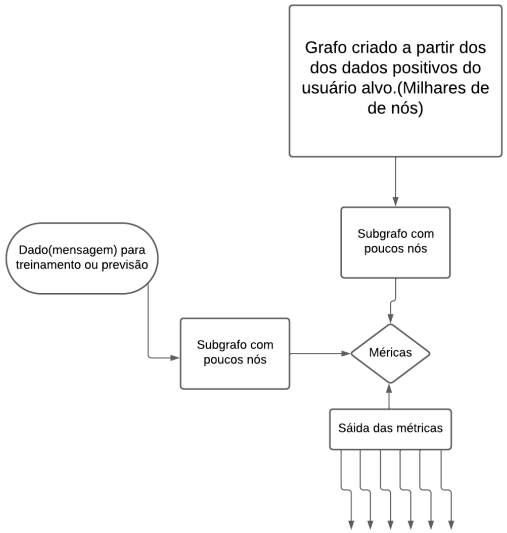


Figura 4. Histograma dos tamanhos das mensagens na fração da base de dados utilizada.

Para evitar a criação de matrizes esparsas e grandes demais, é feito um mapeamento dos valores, para se criar então duas matrizes que são do mesmo tamanho da quantidade de termos distintos na entrada, como ilustra a Figura 4.

Por exemplo para a entrada = [10,20,200,10], é feito um mapeamento para: [0,1,2,0].

0	1	0.5
0	0	1
1	0	0

Tabela 2. Vetor da entrada remapeado para uma matriz de tamanho reduzido

0	3	0.7
0	0	0.9
7	0	0

Tabela 3. Matriz que representa o subgrafo extraído do grafo total do usuário

Os valores da primeira matriz na Tabela 2, são preenchidos através do mesmo algoritmo usado para a construção do grafo na seção 3.1, utilizando a distância entre os termos como forma para calcular as intensidades das arestas.

Na segunda matriz na Tabela 3, os valores são obtidos recuperando estes do grafo do usuário alvo.Então no trecho: [10,20 ,...], se recupera o grafo os valores naquela posição no grafo: *grafoUsuario[10][20],* e mapeados para a posições adequadas.Neste exemplo o termo "10" está mapeado para a posição "0" e "20" para a posição "1", então: *matriz2[0][1]* =

grafoUsuario[10][20]. Estas matrizes são a entrada para o cálculo das métricas que retornarão as saídas para o modelo neural.As matrizes aqui estão demonstradas sem a normalização.

3.5 A Abordagem Completa

A construção do modelo final acontece então da seguinte forma:

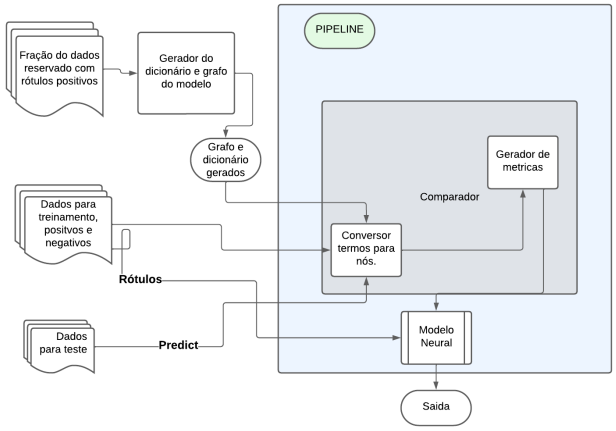


Figura 5. Pipeline do modelo

Na Figura 5 é apresentada a composição da aprendizagem e previsão do modelo. Uma fração dos dados cria o grafo que representa o vocabulário do usuário, após isso é feito o treinamento do modelo neural. Onde os dados positivos e negativos são enviados para o conversor de termos para nós, onde o vetor de strings, se torna um vetor de valores inteiros, estes vetores são enviados para o gerador de métricas que transforma estas entradas em matrizes e produz a seis métricas de similaridade como descrito na seção 3.4. Uma matriz de n linhas, onde n é o número de dados de teste e seis colunas, será a entrada de treinamento do modelo neural.Este mesmo algoritmo é empregado para as previsões, transformando as strings e gerando as métricas para o modelo neural gerar uma ativação.

4. RESULTADOS DOS EXPERIMENTOS

4.1 Dados para treinamento e validação

4.1.1 Autores e proporção de dados para treinamento e validação.

Para o primeiro momento do experimento, foram escolhidos seis usuários da base de dados, onde uma transformação é aplicada sobre o volume de mensagens, de forma a padronizar a amostragem. Com 510 exemplos cada, as categorias: “Mikeforall”, “NyeasB0ss”, “Chris Sama “, “mneme” , “Zhareth” e “DeadlyDays” têm seus dados divididos de forma que 80 exemplos de cada são reservados para teste de verdadeiro positivo e acurácia cruzada. Tais dados não participam da construção do grafo de nenhum usuário, nem do seu treinamento, muito menos do treinamento de qualquer outro modelo, sendo totalmente desconhecidos para qualquer um dos modelos.

Para treinamento de cada modelo independente, com os dados restantes, se escolhem 4 usuários (excluindo o usuário alvo) de forma aleatória escolhendo 120 mensagens de cada, estes serão os exemplos negativos para o treinamento. Um outro usuário, também escolhido de forma aleatória, e este será o teste de verdadeiro negativo, isto é, a capacidade de identificar exemplos negativos provenientes de um usuário que o modelo não conhece algum exemplo. O objetivo é observar então a capacidade de generalização para dados que não possuem semelhança com o que foi treinado como negativo.

Para a construção do grafo, são separados 290 mensagens positivas, que não participarão do treinamento do modelo neural, restando então 432 exemplos negativos e 126 positivos para essa tarefa. A distribuição do tamanho das mensagens de cada treinamento ou classe é aleatória.

Ao final do treinamento de cada modelo, é feito um teste para verificar a acurácia de verdadeiro positivo, a partir dos dados que foram reservados, como explicado anteriormente, neste momento, cada modelo tenta prever a acurácia em mensagens positivas inéditas.

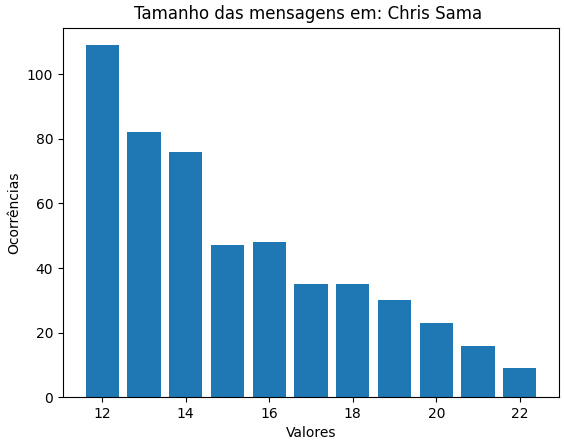


Figura 6. Histograma Chris Sama

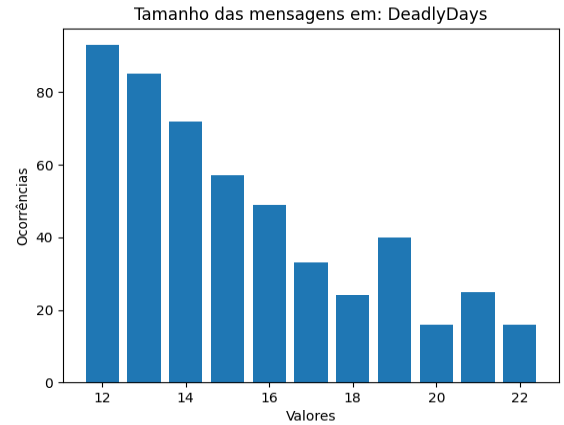


Figura 7. Histograma DeadlyDays

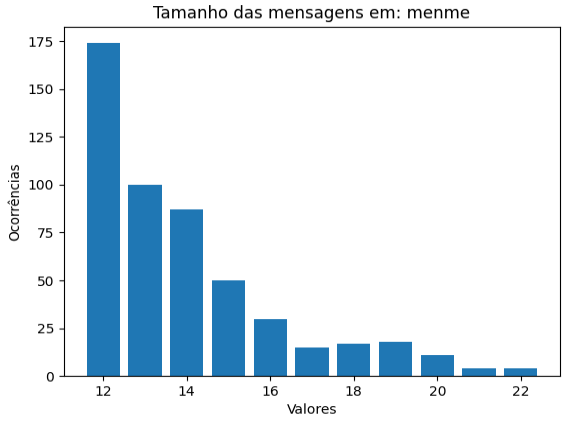


Figura 8. Histograma menme

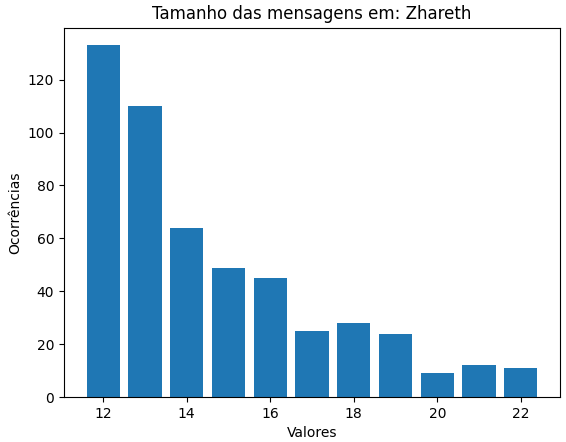
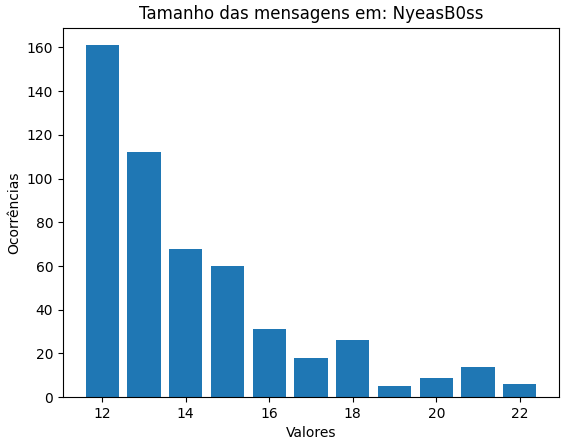
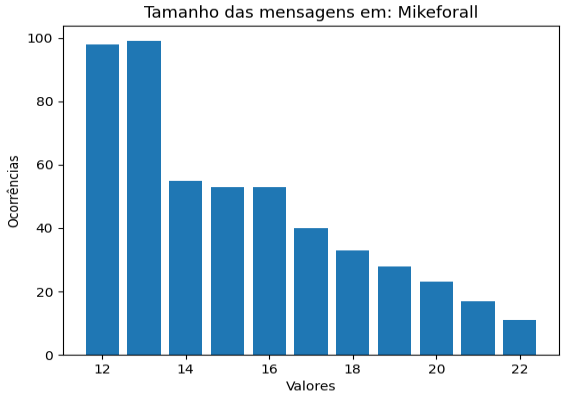


Figura 9. Histograma Zareth





Figuras 10 e 11 respectivamente. Histograma de NyeasB0ss e Mikeforall

4.2 Treinamento do modelo

Numa rodada de treinamento para um limiar de ativação de 0.3, são retornados os seguintes resultados (Tabela 4):

Modelo treinado	**Colunas exemplos negativos (120 amostras cada)**	**Alvo para teste VN (430 amostras)**	**Acurácia VP (80 amostras)**	**Acurácia VN**
‘Chris Sama ’	'Mikeforall', 'NyeasB0ss', 'menme', 'DeadlyDays'	Zhareth	0.75	0.99
NyeasB0 ss	'menme', 'Chris Sama ', 'DeadlyDays', 'Mikeforall'	Zhareth	0.53	1.0
Zhareth	'DeadlyDays', 'NyeasB0ss', 'Mikeforall', 'menme'	'Chris Sama '	0.86	0.99
menme	'Mikeforall', 'Chris Sama ', 'NyeasB0ss', 'Zhareth'	'DeadlyDays'	0.57	1

DeadlyDays	'NyeasB0ss', 'menme', 'Zhareth', 'Mikeforall'	'Chris Sama '	0.56	1
‘Mikeforall’	'menme', 'Chris Sama ', 'Zhareth', 'NyeasB0ss'	'DeadlyDays'	0.71	1

Tabela 4. Resultados dos testes iniciais

O próximo passo então é a verificação através da matriz de confusão das intensidades das ativações de cada modelo, para todos os dados de teste - as 80 mensagens de cada categoria-, que foram reservadas, como explicado anteriormente. Neste teste o que se está medindo é intensidade de ativação de cada modelo para os cada teste, o modelo com maior ativação “pontua ” naquele exemplo específico.A última coluna “NOTFOUND”, representa o momento em que um exemplo, não é capaz de ser predito com um valor acima do limiar por nenhum dos modelos. No exemplo abaixo, o limiar estabelecido foi de 0.1, deve-se observar o trade-off entre aumentar a acurácia dos VP e diminuir as dos VN.

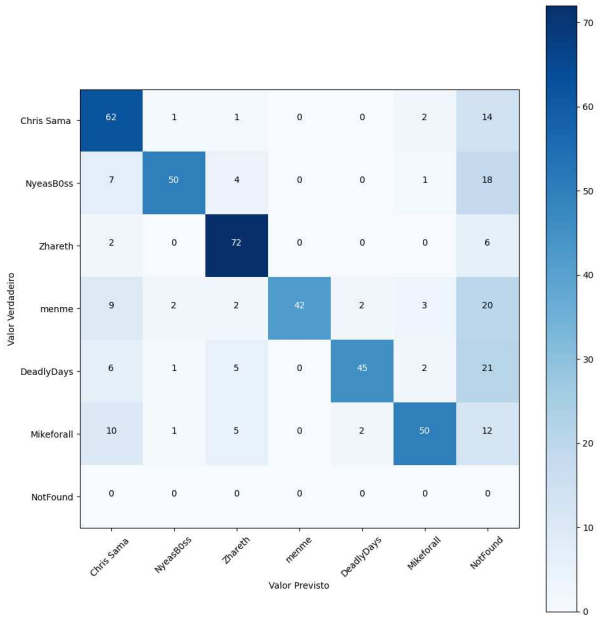


Figura 12. Matriz de validação cruzada

O modelos conseguem estabelecer uma relativa boa acurácia (Figura 12), com alguns se destacando com sua capacidade de representatividade ao evitar falsos positivos, com o “Zhareth” possuindo uma acurácia de 90% em seus verdadeiros, idealmente os modelos devem estar sempre gerando distâncias maiores entre os os valores para cada rótulo, criando uma fronteira segura para se mover o limiar sem perder a acurácia dos verdadeiros positivos.

4.3 Resultado teste adicional de VN

Um teste adicional é executado sobre um conjunto totalmente novo de mensagens, que passam pelo mesmo pré-processamento, estas não aparecem em momento algum nos processos anteriores.

Utilizando das mensagens dos usuários: "TDXD" e "jgfan27", um teste agora é executado com uma porção de 5584 mensagens.

‘Chris Sama ’	0.7
‘NyeasB0ss’	1.0
‘Zhareth’	0.94
‘menme’	1.0
‘DeadlyDays’	0.98
“Mikeforall”	0.89

Tabela 5. Testes finais VN

Neste teste na Tabela 5 , onde o limiar é 0.1, nota-se que a maioria dos modelos conseguiu boa precisão nos verdadeiros negativos, sendo ‘Chris Sama ’ o de menor precisão, o mesmo conseguiu uma ótima acurácia nos testes do verdadeiros positivos anterior, o que poderia indicar que os modelos de maior precisão VP estão gerando saídas muito fortes com frequência porém ‘Zhareth’, possui precisão melhor ainda no teste anterior, e junto com ‘NyeasB0ss’ e ‘‘Mikeforall’ conseguiram precisões altas neste último teste de VN.

4.4 Resultados

Os modelos demonstraram consistência ao manter boa acurácia de verdadeiros negativos, no entanto a acurácia para verdadeiros positivos ainda acaba flutuando dentro de um considerável limiar. A rede neural de pequena complexidade permite um treinamento muito rápido, que permite uma fácil opção para ajustes de parâmetros.

5. EXPERIÊNCIA

Nesta seção serão descritas experiências proporcionadas pelo processo de desenvolvimento do modelo, bem como os desafios, limitações e propostas para a continuidade e futuro.

5.1 Pipeline de um projeto em aprendizagem de máquina

Todo o processo de desenvolvimento de um projeto em aprendizagem de máquina envolve uma acurada análise e criticismo sobre os dados e as escolhas a serem feitas para modelar o problema. Escolhida a plataforma para realizar o trabalho a coleta dos dados buscou um cenário onde estes refletissem uma visão natural do problema.

A ideia de utilizar grafos surge do objetivo de tentar algo diferente de abordagens mais comuns utilizando redes recorrentes, por exemplo.

5.2 Desafios e limitações

Organizar as etapas de uma pipeline de um modelo, envolve a análise sobre as tentativas e erros proporcionadas pelos problemas, estas etapas constroem uma visão mais acurada do comportamentos dos dados, e como a busca por soluções mais simples e criativas devem guiar todo o processo.

5.3 Trabalhos futuros

Existe um grande espaço para experimentação de métricas e soluções novas, se apoiando em ambientes distintos para os testes realizados. Novas métricas dentro do tema “grafos” certamente

podem ser aplicadas para gerar observações mais acuradas das relações entre os nós.

6. CRESCIMENTO PESSOAL

Durante o projeto foram utilizados diversos conhecimentos obtidos durante o curso de Ciência da Computação, com destaque para disciplinas como Redes Neurais e Percepção Computacional.Esta experiência permitiu adquirir novas habilidades técnicas assim como um reforço para as capacidades analista e criativa.

7. REFERÊNCIAS

[1] https://dictionary.cambridge.org/pt/dicionario/ingles/reliability

[2] https://www.aser.com.br/conheca-os-perigos-do-roubo-de-credenciais-para-as-empresas/

[3] Zheng, R., Li, J., Chen, H. & Huang, Z. A framework for authorship identification of online messages: Writing-style features and classification techniques. J. Am. Soc. Inform. Sci. Technol. 57, 378–393 (2006)

[4] Vilariño, D., Pinto, D., Gómez-Adorno, H., León, S., & Castillo, E. (2013). Lexical-syntactic and graph-based features for authorship verification. CLEF 2013 Evaluation Labs and Workshop, Online Working Notes, Valencia, Spain, CEUR Workshop Proceedings

[5] CASTILLO, Esteban; CERVANTES, Ofelia y VILARINO, Darnes. Text Analysis Using Different Graph-Based Representations. *Comp. y Sist.* [online]. 2017, vol.21, n.4 [citado 2023-06-14], pp.581-599. Disponible en: <http://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S1405-55462017000400581&lng=es&nrm=iso>. ISSN 2007-9737.

[6] [Graph-based Text Representations: Boosting Text Mining, NLP and Information Retrieval with Graphs](https://aclanthology.org/D17-3003) (Malliaros & Vazirgiannis, EMNLP 2017)

[7] https://www.britannica.com/topic/graph-theory

[8] https://www.analyticsvidhya.com/blog/2020/02/4-types-of-distance-metrics-in-machine-learning/

[9] https://mathworld.wolfram.com/MatrixSpectrum.html

[10] https://www.kaggle.com/datasets/jef1056/discord-data/code

Sobre o autor:

Caio Maxximus Pereira Soares é aluno do curso de Ciência da Computação na Universidade Federal de Campina Grande.