Detecção de Toxicidade e Viés não intencional

Gabriel Kendy Faria Komatsu
University of São Paulo – (USP)
São Paulo, Brazil
gabriel.kendy@usp.br
10816711

Isabella Kuo
University of São Paulo – (USP)
São Paulo, Brazil
15483114.ik@usp.br
10783080

Rafael Rodrigues Braz
University of São Paulo – (USP)
São Paulo, Brazil
rafael.braz@usp.br
6482483

Vinicius Alves Matias
University of São Paulo – (USP)
São Paulo, Brazil
viniciusmatias@usp.br
10783052

***Abstract*—A popularidade das redes sociais aumentou significativamente os debates onlines, onde ocasionalmente vemos comentários tóxicos que ofendem um indivíduo ou grupo social. Numa tentativa de lidar com esse problema de forma escalável podemos utilizar a aprendizado de máquina para classificação, isso porém não é uma tarefa trivial, por poder apresentar um viés que reproduz a própria natureza ofensiva dos comentários aprendidos. Neste trabalho será utilizado o dataset Jigsaw Unintended Bias com o objetivo de treinar classificadores capazes de discernir entre comentários tóxicos e não tóxicos tentando minimizar o viés indesejado. Dos classificadores testados, obtivemos bons resultados com a Regressão Logística, LSTM e o BERT.**

***Index Terms*—toxicity, text classification, machine learning, data mining**

I. Introdução

A compreensão da língua humana é uma tarefa extremamente complexa, e portanto é válido atuar nessa questão com aprendizado de máquina. Um problema relevante que envolve o processamento de texto é a identificação de discursos ofensivos, isso devido à sua vasta presença em redes sociais e assim exigindo uma escalabilidade maior para tratar esse fenômeno que muitas vezes fere os direitos humanos. Tendo isso em mente, neste trabalho foi utilizado o conjunto de dados Jigsaw unintended bias, que possui comentários ofensivos e não ofensivos avaliados em uma escala de 0 a 10. Uma grande preocupação deste dataset é a presença de unintended bias (viés acidental). Todo modelo de aprendizado de máquina irá apresentar algum nível de viés; se a tarefa é classificar comentários tóxicos o modelo terá um viés intencional de forma que um comentário tóxico possa ter uma nota maior, porém se o modelo discriminar determinadas entidades que surgem nos comentários temos um viés não intencional, pois não é a intenção do modelo reproduzir preconceitos e discriminar entidades frequentemente atacadas.

Sendo assim, o objetivo deste trabalho é classificar comentários em tóxicos e não tóxicos considerando os classificadores de árvore de decisão; Naive Bayes; Regressão Logística; Convolutional Neural Networks (CNN); Long Short Term Memory (LSTM); e o Bidirectional Encoder Representations from Transformers (BERT). Para amenizar o unintended bias podemos utilizar a grande quantidade de comentários disponibilizados pelo dataset, que possui em sua maioria comentários não tóxicos, deixando o dataset desbalanceado, porém representando uma proporção semelhante às das redes sociais.

O acesso aos arquivos de exploração e aos modelos de redes neurais está acessível pelo Google Drive, com acesso por email USP [1], e atualmente somente os modelos baseados em Tf-idf e a implementação do BERT estão disponíveis em um repositório do Github [2].

II. Fundamentação teórica

A. Tokenização

Em processamento de linguagem natural, tokenização é o processo de subdividir um documento em partes menores chamadas de *tokens* que são utilizados para produzir características para um modelo de aprendizado de máquina. Tokens podem ser palavras, caracteres ou mesmo uma sequência de bytes. Neste trabalho foi utilizada a tokenização por palavras. No entanto, como os documentos são comentários produzidos para a internet, algum cuidado adicional é necessário. Assim, foi utilizado o *TweetTokenizer* da biblioteca scikit-learn, já preparado para lidar com alguns artefatos específicos da linguagem da internet, como emoticons, urls e e-mails.

B. Term frequency–inverse document frequency

O *Term frequency–inverse document frequency* (TF-IDF) é uma medida estatística que indica a relevância de um termo de um documento e é uma das estratégias para se produzir características numéricas a partir de um documento tokenizado. O TD-IDF é composto por duas partes: *term frequency* (TF) e *inverse document frequency* (IDF).

O TF é dado pelo número de vezes que o termo i aparece no documento j ($n_{i,j}$) dividido pelo total de número de termos no documento j, e, assim, é dado pela equação 1.

[1] https://drive.google.com/drive/folders/1LgutBWVg-zNEmTK2BUPGcA91YJJgFmE6?usp=sharing)

[2] https://github.com/rrbraz/toxicity-classification

$$\mathrm{tf}(i,j) = \frac{n_{i,j}}{\sum_k n_{k,j}} \tag{1}$$

Por outro lado, o IDF calcula o peso dos termos que aparecem no *corpus* (conjunto de todos documentos), considerando que termos menos frequentes são mais relevantes. O IDF é dado pela equação 2.

$$\mathrm{idf}(i) = \log\left(\frac{N}{n_i}\right) \tag{2}$$

Onde N é o número total de termos em todos os documentos do corpus, e n_i é o número de vezes que o termo i ocorre. Assim, o TF-IDF é dado pelo produto do TD e do IDF:

$$\text{tf-idf}(i,j) = \frac{n_{i,j}}{\sum_k n_{k,j}} \cdot \log\left(\frac{N}{n_i}\right) \tag{3}$$

C. N-gramas

Um n-grama é uma sequência contígua de elementos de um texto. Esses elementos podem ser caracteres, palavras, bytes, sílabas, fonemas ou qualquer outro elemento relevante para o texto. Dividir um documento em n-gramas é uma técnica comumente utilizada em processamento de linguagem natural. Este trabalho utiliza n-gramas de palavras, mais especificamente, n-gramas de tokens produzidos pelo TweetTokenizer.

Assim, a partir do comentário de um usuário, produziu-se sequências de tokens adjuntos de comprimento n. Por exemplo, para a sentença "Today is too hot" e $n = 2$, são obtidos os seguintes 2-gramas: "Today is", "is too", "too hot". Pontuações e *stop words* são removidas durante esse processo.

D. Classificadores

Um classificador é um algoritmo de aprendizado de máquina que tem por objetivo a partir de um conjunto de características numéricas predizer corretamente o rótulo de um dado. Neste trabalho os classificadores são utilizados para predizer se um comentário é tóxico ou não. Os algoritmos selecionados para esta tarefa estão dentre os mais simples e mais utilizados em processamento de linguagem natural: Árvore de decisão, Naive Bayes e Regressão Logística.

1) Árvore de Decisão: É um algoritmo que constrói um caminho em um grafo baseado nas características que mais contribuem para o modelo. A cada nó, um caminho é escolhido com base na característica que produz o maior ganho de informação. A implementação de árvore de decisão utilizada é fornecida pela biblioteca *scikit-learn* e é uma versão otimizada baseada do algoritmo CART (Classification and Regression Tree), que por sua vez é muito similar ao algoritmo C4.5, mas que suporta regressão numérica além de classificação.

2) Naive Bayes: É um modelo probabilístico baseado no teorema de Bayes que assume que pares de variáveis são condicionalmente independentes para fins de classificação. A partir disso, derivando o teorema de Bayes é possível chegar no seguinte estimador:

$$\hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i \mid y) \tag{4}$$

Diferentes versões do algoritmo diferem nas hipóteses sobre a distribuição de $P(x_i \mid y)$. Neste trabalho, a implementação utilizada foi a Multinomial, tipicamente utilizada para classificação de texto, e que assume que os dados seguem uma distribuição multinomial.

3) Regressão Logística: É um modelo linear de classificação cujo objetivo é produzir um modelo capaz de predizer valores a partir de um conjunto de observações, em função de uma ou mais variáveis contínuas ou binárias. Com o modelo obtido é possível classificar ou prever a probabilidade de um evento ocorrer dado uma observação aleatória.Essa técnica é comumente utilizada para predição de doenças, porém é interessante testá-la para o problema proposto que consiste numa classificação binária entre comentários tóxicos e não tóxicos. Para obter o modelo é utilizada a função logística:

$$\hat{p} = \frac{e^{\beta_0+\beta_1 x_1}}{1+e^{\beta_0+\beta_1 x_1}} = \frac{1}{1+e^{-(\beta_0+\beta_1 x_1)}} \tag{5}$$

Para otimizar a resolução da função foi utilizado o solver lbfgs, cujo objetivo é otimizar os parâmetros beta usando uma quantidade limitada de memória do computador.

4) Convolutional Neural Networks - CNN: Redes neurais convolucionais são arquiteturas específicas de redes neurais reconhecidas por promover bons resultados em tarefas de aprendizado relacionado a imagens, e mais recentemente para tarefas de classificação de texto. As maneira que são constituídas as camadas de CNNs as diferenciam de redes neural artificias como *MLPs* pelo fato de explorar o campo de entrada realizando convoluções (uma ideia semelhante a pequenas janelas deslizantes dentro de uma matriz maior - a entrada) de maneira a abstrair e computar as informações mais relevantes em cada passada da janela de que realiza as convoluções.

Para tarefas gerais de classificação de texto é tipico formular CNNs usando 4 camadas [1]: Uma camada de *Embeddings*, que é responsável por transformar o formato textual das entradas em representações numéricas de tamanho fixo (*n*-dimensões) utilizando técnicas que exijam mais tempo como *Word2Vec* ou que sigam estratégias mais curtas baseadas em redução de dimensionalidade de vetores *One-hot encoding* que é usada pela biblioteca *Keras*; Camadas convolucionais que exploram o campo de *embeddings* (onde cada sentença é representada por um vetor de tamanho único) realizando convoluções em todo o vetor por meio de pequenas janelas chamadas cujo tamanho é representado pelo conceito de *kernel* ou filtro, a cada passada da janela são geradas saídas para a próxima camada, representando as features detectadas pelas convoluções de cada sentença; Camadas de *Pooling* que reduzem a dimensionalidade dos resultados da camada de convoluções aplicando alguma técnica para sumarizar uma saída com base nos resultados vizinhos a fim de extrair apenas as informações relevantes (é comum utilizar o maior valor dos vizinhos para essa camada, sendo essa técnica conhecida como *Max Pooling* global caso olhe para todas as saídas, ou local caso olhe para apenas um intervalo; e por fim uma camada totalmente conectada para realizar o processo de *Feed-Forward*

de redes neurais artificias comuns, gerando as saída final por meio de uma função de ativação baseada em um método de perda. Um exemplo de rede que segue este processo é visto na figura 1.

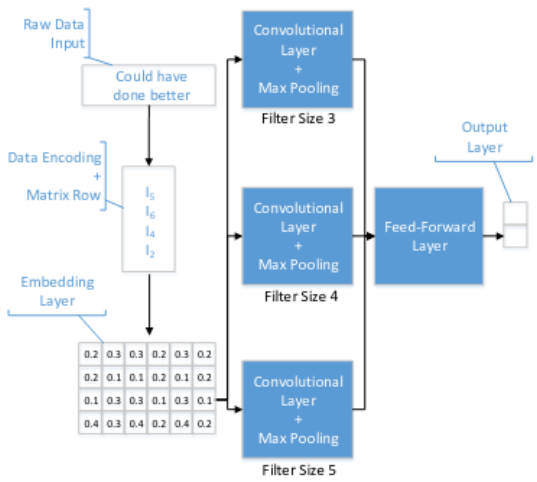

Fig. 1. Arquitetura de uma CNN para tarefas de texto [1]

5) Long Short Term Memory - LSTM: A arquitetura LSTM é um caso especial de redes neurais recorrents (RNNs). A arquitetura de redes recorrentes existe para problemas onde é importante persistir as informações para gerar predições mais próximas da representação real, caso de textos, onde diferentes contextos podem ser levados em conta para gerar uma predição que faça sentido. Utilizar a rede recorrente inteira para gerar os resultados pode, contudo, trazer problemas, pois diversas vezes a informação necessária está somente nas informações mais recentemente recebidas, mas em outros casos é necessário mais contexto, e consequentemente os resultados anteriores da RNN. As Redes de Memória de Curto Prazo Longo (*LSTM's*) surgem para suprir este problema.

Cada célula da LSTM terá três portões que determinarão o que deverá ser lembrado ou esquecido para a próxima célula da rede recorrente. O *Forget Gate* é responsável por zerar os pesos das features que devem ser esquecidas, e em conjunto com o *Input Gate* (que faz um processo semelhante ao portão anterior, mas dessa vez repassa apenas o que foi considerado como importante para ser lembrado, não zerando os atributos) as saídas dos dois portões são entregues ao portão de Output que realiza a ponderação do que deve ser passado para a próxima célula da rede.

6) Bidirectional Encoder Representations from Transformers - BERT: Os Transformers surgiram como uma evolução das redes neurais recorrentes para resolver problemas de processamento de linguagem natural. Modelos baseados em transformers podem realizar tarefas como tradução, extração de informação, jogos de questionários, produção de texto ou classificação.

Transformers são compostos por dois módulos: um *encoder* (codificador) e um *decoder* (decodificador), como representado na figura 2. O *encoder* é responsável por processar a sequência de entrada e transformá-la em uma representação do seu contexto. Já o *decoder* é responsável por produzir a sequência de saída. Em um problema de tradução, por exemplo, a entrada é o texto na língua de origem que é processado, produzindo uma representação de contexto livre independente de linguagem. Esse contexto é então passado para o *decoder*, que o transforma no texto da linguagem alvo, concluindo o processo de tradução.

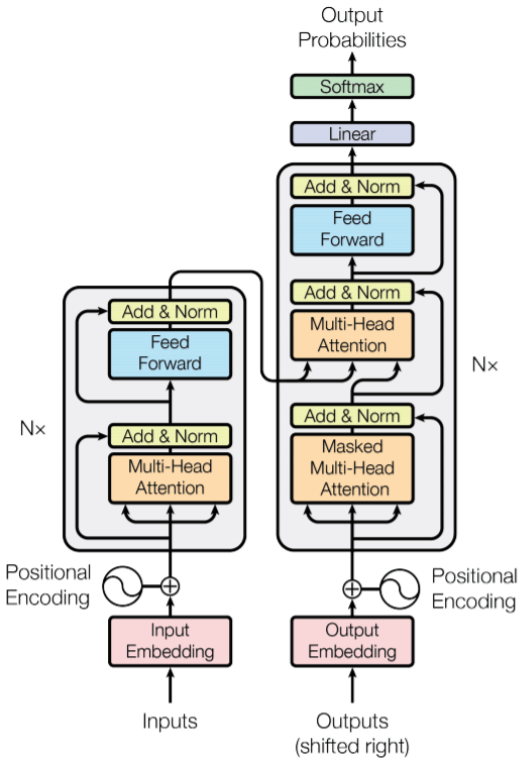

Fig. 2. A arquitetura encoder-decoder de um transformer [2]

Bidirectional Encoder Representations from Transformers, também conhecido como BERT, é um modelo de representação de linguagem que utiliza apenas os módulos *encoder* dos transformers. Esse modelo possui a vantagem de poder ser pré-treinado para reconhecer uma linguagem e então posteriormente ajustado (*fine-tuning*) para uma variedade de tarefas distintas [3]. No BERT, cada palavra é relevante para o contexto, que é construído bidirecionalmente, olhando os contextos tanto pela esquerda quanto pela direita da palavra.

Dada a capacidade de transferência de aprendizado do BERT, diversos modelos pré-treinados em uma variedade de linguagens são disponibilizados publicamente. Neste trabalho os textos processados são na língua inglesa, assim, optou-se por utilizar o modelo *bert-base-uncased* disponibilizado pelo google. Este modelo conta com 12 camadas e 110 milhões de parâmetros. O Google também disponibiliza um outro modelo maior, *bert-large-uncased*, com 24 camadas e 336 milhões de parâmetros, mas que não foi utilizado devido ao alto custo computacional.

Esse BERT pré-treinado é considerado "sem cabeça" e não

é treinado para nenhuma tarefa específica. Ao receber um texto de entrada, esse modelo primeiro faz a tokenização do texto e então produz *embbedings* - representações vetoriais do texto e seu significado. Para fazer o ajuste fino desse modelo, treinando-o para uma tarefa de classificação, é necessário adicionar uma "cabeça" de classificação, tipicamente uma camada linear que recebe de entrada um *embedding* especial denominado [CLS] seguida por uma função *softmax*. Após isso, é usada uma variação do algoritmo de gradiente descendente para treinar o modelo com os dados específicos do problema de classificação a ser resolvido.

III. Trabalhos consultados

A principal abordagem para lidar com o problema apresentado foi escolher modelos de classificação de boa reputação para lidar linguagem ofensiva e unintended bias, e modelos mais simples para ter de baseline. Temos também uma grande gama de benchmarks devido ao fato de que o problema foi disponibilizado em uma competição da plataforma Kaggle. Dos modelos mais próximos do estado da arte temos o BERT, que como explicado no artigo [4] é um modelo com um mecanismo de auto-atenção que permite um melhor entendimento das frases e se atenta em perceber quais palavras estão conectadas entre si.

A escolha de redes neurais também foi bastante interessante, como visto em [5] a rede LSTM possui a capacidade de aprender textos compridos proficientemente, sabendo escolher bem o que deve ser aprendido e esquecido em cada camada. Uma outra rede que vem ganhando notoriedade na classificação de testo é a CNN, que é tradicionalmente utilizada para o processamento de imagens, mas como mostrado em [1] os resultados da CNN são são até superiores que alguns métodos tradicionais de classificação de texto.

Sobre a obtenção de baselines, foi utilzado tf-idf com modelos mais simples de classificação, além de já terem demonstrado algum sucesso em outros trabalhos relacionados, como em [6] e [7], vemos que por serem mais simples e portanto mais rápidos, servem como um bom comparativo para analisar os modelos mais complexos.

IV. Definição do problema

O objetivo deste trabalho é construir modelos de detecção de toxicidade em comentários produzidos em discussões online. Neste contexto, toxicidade é definida como qualquer coisa rude, desrespeitosa ou que possa de alguma outra forma fazer alguém abandonar a discussão. Os modelos produzidos tem que ser capazes de produzir bons resultados em diversos cenários ao mesmo tempo que evitam reproduzir viés dos dados. Como os dados foram extraídos de fontes disponíveis, onde, infelizmente, certas identidades são amplamente mencionadas de maneiras ofensivas, os modelos podem carregar esse viés e aprender a associar incorretamente termos associados a esses grupos com toxicidade. Um exemplo possível é a palavra "gay", que pode ser associada da toxicidade, mesmo em comentários não tóxicos (como em "I am a gay woman"). Os modelos treinados tem de ser avaliados sob essa perspectiva de modo a evitar replicar esse viés para os usuários.

A. Conjunto de dados

O dataset utilizado neste trabalho foi produzido pela Jigsaw, uma unidade dentro do Google que explora ameaças à sociedades abertas, e disponibilizado em https://www.kaggle.com/c/jigsaw-unintended-bias-in-toxicity-classification/data. Ele é constituído por 1.999.516 comentários realizados na plataforma *Civil Comments* até 2017, quando esta encerrou suas atividades. A Jigsaw então enriqueceu esses dados com avaliadores humanos anotando diversos atributos relacionados a toxicidade.

O texto do comentário individual está contido na coluna comment_text. Cada comentário possui um rótulo de toxicidade a ser predito. A toxicidade, e todos demais atributos anotados por humanos, é representada nesses dados por um valor fracional que representa a fração de avaliadores humanos que acreditaram que o rótulo se aplica ao comentário. Para fins de classificação, comentários com toxicity ≥ 0.5 foram considerados da classe positiva (tóxico).

Além disso, os dados possuem atributos que dizem respeito às identidades mencionadas no texto do comentário (não necessariamente de forma tóxica). Esses atributos não devem ser utilizados como características na classificação, pois comentários novos produzidos em discussões online não viriam com esses rótulos, mas podem ser utilizados para avaliar a presença viés nos classificadores. As identidades analisadas foram apenas as que possuem mais de 500 exemplos no conjunto de testes e são:

- male
- female
- homosexual_gay_or_lesbian
- christian
- jewish
- muslim
- black
- white
- psychiatric_or_mental_illness

Esses dados exigiram uma limpeza inicial para o treinamento dos classificadores, removendo linhas com valores faltantes e linhas com texto de comentário duplicado (foi mantida apenas a primeira ocorrência encontrada). Após remover essas linhas, o conjunto de dados permaneceu com 1.971.915 linhas.

Como o atributo *toxicity* possui valor fracional, foi necessário adicionar um novo atributo categórico para a toxicidade com classe positiva para $toxicity \geq 0.5$, que foi usado como alvo de treinamento, descartando o atributo original. Como podemos ver nos gráficos 3 e 4, o conjunto de dados resultante é desbalanceado, com apenas 8% dos dados com a classe positiva.

Um outro ajuste que foi necessário foi remover alguns caracteres especiais dos textos que não contribuíam com o valor semântico, como *soft hyphens* e outros caracteres que dizem respeito apenas à codificação do texto, e não ao conteúdo mensagem.

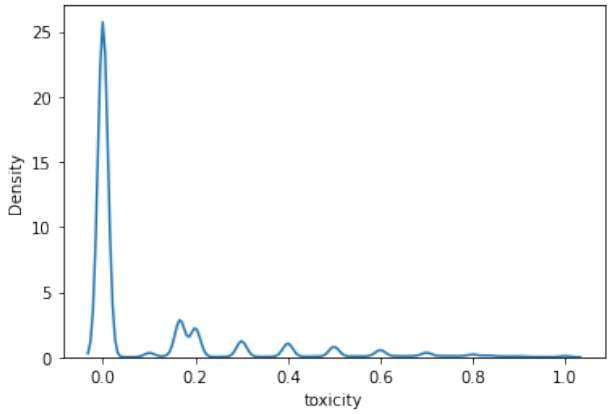

Fig. 3. Estimativa de densidade kernel da *toxicity* nos dados

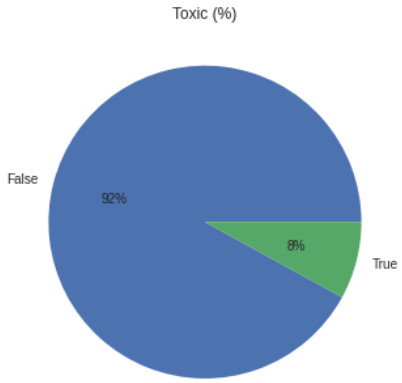

Fig. 4. Frequência da classe tóxico

V. Exploração das características dos textos

A fim de realizar uma análise dos textos em questão do comportamento mediante o contexto dos dados antes de apresentar a tarefa de classificação, utilizaremos duas abordagens de mineração de dados e processamento de língua natural para auxiliar na familiarização ao conjunto de dados e às relações computacionais realizadas.

A. Associações entre tokens

Regras de associação são usadas em mineração de dados para identificar quais elementos geram como resultado outros. Isso pode ser aplicado para identificar em exemplos extremamente reduzidos de textos, quais relações se destacam e possivelmente diferenciam um estilo de escrita de outro. Para identificar essas relações é comum utilizar o algoritmo *apriori*, cuja implementação em Python está disponível pela biblioteca *mlxtend*.

Para a aplicação do algoritmo em tempo viável neste problema de texto foi necessário pegar uma pequena amostra de textos não tóxicos (1% do total) e outra de textos tóxicos (5% do total, dado que há menos dados de textos tóxicos). Para cada texto foram removidas as *stopwords* (pois somente poluiriam a visualização), e foram representados em vetores usando a estratégia *Bag of Words*, que conta a quantidade de palavras que uma palavra aparece em uma frase, e cada dimensão então é uma palavra. A diferença nessa implementação de *Bag of Words* é que ele foi determinado de maneira binária, ou seja, se uma palavra aparece em um texto ela tem 1 nessa coluna e 0 caso contrário.

Dado a esparsidade dos textos a métrica de suporte para o agoritimo *apriori* foi de 0.02, representado a proporção de relações que teremos entre as palavras do corpus. Também por haverem poucos dados, a métrica de confiança em um resultado para gerar uma regra de associação foi de 0.2, mas é útil para gerar menos nós e se visualizar melhor um pequeno exemplo de associação.

A figura 5 apresenta a associação entre os 5 *tokens* que se destacaram nos textos não tóxicos. Pode-se notar que não há algo realmente informativo quando se observa somente esta imagem, além de que parece haver uma tendência à gerar implicações às palavras *people* e *would* (verbo no passado).

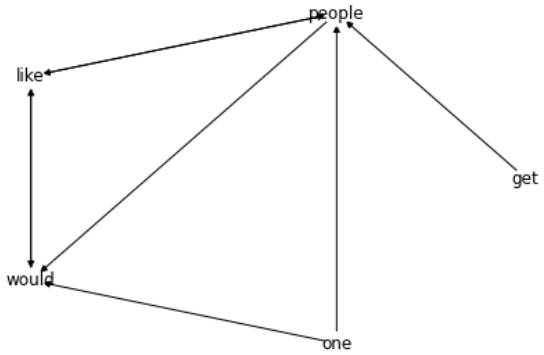

Fig. 5. Associação de textos não tóxicos

A análise das associações tóxicas da 6 já permite uma comparação à figura anterior e identificar novas relações. A adição do token *one* adiciona um individualismo que emerge, diferente de quando havia apenas *people* (que é uma relação de consequência nestes textos também). Em especial, há a adição dos tokens *president* e trump, o que deixa claro o viés da época e regiões onde o conjunto de textos foi extraído pelo seu destaque.

B. Representações utilizando embeddings

Parte essencial para atuar com tarefas de classificação de textos é ter uma boa representação numérica das sentenças textuais a fim de gerar classificadores que possam se aproveitar do contexto dos dados em que estão inseridos, e assim representar os textos com a menor perda de informação possível. De fato, estratégias como *Tf-idf* e *embeddings* realizam esses processamentos, de maneira muito eficiente para a maioria dos problemas.

Na Fundamentação Teórica fora mencionada uma representação de *embeddings* utilizada pela biblioteca *Keras* para servir de entrada para modelos neurais como

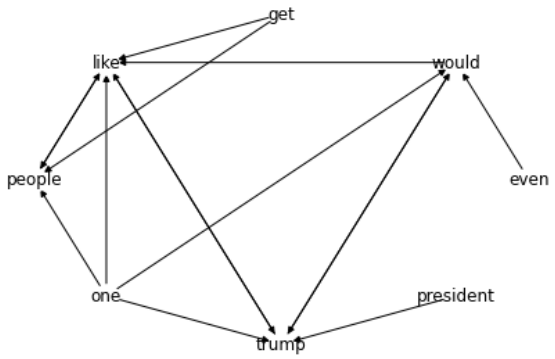

Fig. 6. Associação de textos tóxicos

CNNs, contudo não é a única implementação possível. Dois algoritmos para geração de embeddings são bem utilizados para representar textos como vetores de *n* dimensões de acordo com todo o campo de textos: *GloVe* e, em especial, *Word2Vec*. O método *Word2Vec* seleciona os *tokens* do texto e representá-os em uma quantidade pré-determinada de dimensões. O cálculo é realizado mediante a tentativa de identificar as palavras vizinhas de uma central de acordo com o corpus entregue (algoritmo *Skip-Gram*). Com isto todas as palavras podem ser representadas um vetor de, neste caso, 300 dimensões.

Tendo a representação vetorial das palavras, torna-se necessário mesclá-las a fim de gerar novamente cada sentença, mas agora representada por vetores de 300 dimensões. A maneira realizada para a visualização a ser apresentada consiste de utilizar os embeddings de cada texto e realizar a média de todas essas representações para cada dimensão ponderada pelo valor *Tf-idf*, ou seja, cada dimensão de um texto será a média da mesma dimensão para todas as palavras do corpus, onde cada vetor de palavra também é multiplicado por seu valor *Tf-idf*.

A figura 7 apresenta uma amostra de 1% dos textos de treinamento (caso balanceado) com a representação numérica baseada no *Word2Vec*.

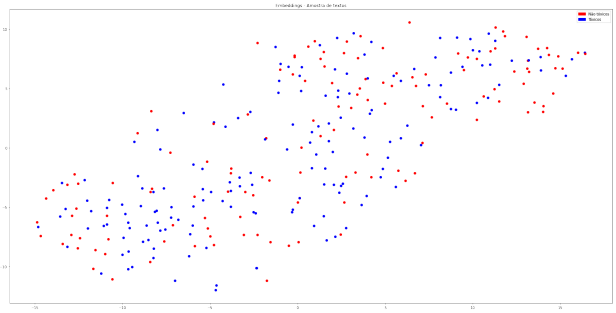

Fig. 7. Representações dos textos utilizando *Word2Vec* e redução de dimensionalidade

É importante mencionar que como haviam 300 dimensões para representar palavras, foi necessário reduzir para 2 e assim criar uma imagem bidimensional. O redutor de dimensionalidade foi o t-Distributed Stochastic Neighbor Embedding (t-SNE) dado sua viabilidade para ser usado em visualizações. Os parâmetros do algoritmo foram de 2500 iterações e *perplexity* de 40, usando 40 vizinhos para geração mais distinta de possíveis clusters. Também, a implementação do *word2Vec* foi feita usando a biblioteca *gensim*.

Ao fim, pode se notar pela figura que não há nenhum *cluster* claro, pelo contrário, há muitas intersecções entre textos tóxicos (em azul) e não tóxicos (em vermelho). Isso também é proveniente de estarmos tratando textos, e consequentemente características subjetivas das pessoas que não torna simples diferenciar um estilo de escrita como totalmente tóxico ou não. Contudo, também é possível notar que é raro apresentar pontos isolados de textos tóxicos ou não tóxicos, sendo na maioria acompanhados por pelo menos dois vizinhos do mesmo estilo de escrita.

VI. Arquitetura da solução

Cinco arquiteturas de classificadores foram utilizadas para resolver o problema de detecção de toxicidade: usando representação textual por TF-IDF, três arquiteturas usando redes neurais e uma usando *fine-tuning* do BERT.

1) Arquitetura dos Classificadores TF-IDF: Para os classificadores treinados usando a representação textual de TF-IDF, várias etapas de pré-processamento foram utilizadas antes de aplicar os modelos propriamente ditos. Primeiramente os textos, em língua inglesa, tiveram suas contrações expandidas por meio da biblioteca *contractions* para python. Por exemplo, "I'm" foi expandido para "I am", dentre outras contrações.

Após isso, os textos foram convertidos para letras minúsculas e as foram removidas as *stop words* ("palavras vazias", comuns da língua e consideradas sem valor semântico para o processamento de linguagem natural). A lista de *stop words* utilizada foi a disponilibilizada pela biblioteca *NLTK - Natural Language Toolkit*. Por fim, foram removidos acentos das palavras.

Uma vez processados, foi utilizado o *TweetTokenizer* para separar o texto em tokens, a partir dos quais foram gerados n-gramas. Cada n-grama foi considerado como um termo distinto para o cálculo do TD-IDF. O conjunto dos TF-IDF de cada n-grama compõe as características de entrada para os modelos classificadores.

Os modelos treinados foram: Naive Bayes Multinomias, Árvore de Decisão e Regressão Logística, todos previamente detalhados na seção "Fundamentação Teórica". A implementação utilizada desses modelos foi a disponibilizadas pela biblioteca *scikit learn*. O diagrama da figura 8 ilustra as etapas desse processamento dos textos até chegar ao classificador. É importante ressaltar que sempre que um texto novo for apresentado aos classificadores ele deve passar previamente por todo esse pré-processamento, valendo inclusive para extrair predições de um ambiente online real.

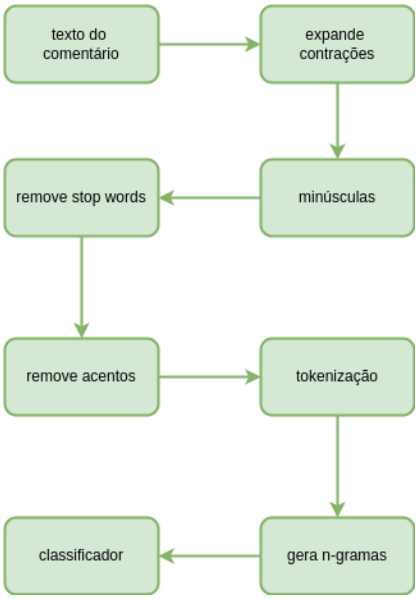

Fig. 8. A arquitetura para modelos TD-IDF

2) Arquiteturas utilizando CNN e LSTM: Foram desenvolvidos três modelos utilizando as redes neurais típicas para tarefas de classificação de texto. Todas contém a camada de embedding proporcionada pela *Keras*, que converte as sequências de tokens para ser compreendida pelo resto da rede. Os textos passam por uma conversão de tokens para índices, e cada sentença é reduzida para um vetor de 300 dimensões (*embeddings*) composta pelos tokens que as constituem ou junção de tokens realizadas pela propría biblioteca quando necessário englobar mais de 300 dimensões.

A primeira arquitetura de rede neural consiste em uma CNN que tem como entrada a camada de embeddings, uma taxa de dropout (desativação de entradas ou neurônios a fim de fazer o modelo se ajustar a casos inesperados e reduzir *overfitting*, uma camada de convolução 1D, uma camada de maxpooling global, outra desativação de neurônios por dropout, e uma camada totalmente conectada densa a fim de trazer o resultado da classificação como output.

A segunda arquitetura consiste de uma LSTM com entrada pela camada de embeddings, seguida da camada da própria LSTM com 100 células / unidades, uma taxa de dropout e uma camada densa.

A última arquietura consiste de uma mescla entre as predições da CNN sendo passadas para uma rede recorrente do tipo LSTM. Tal como as outras, a entrada é uma camada de embedings, seguida de uma taxa de dropout, convolução 1D, maxpooling global, a passagem das saídas da CNN para uma LSTM de 100 células, e uma camada densa.

3) Arquitetura do Classificador BERT: Todo o processo de tokenização e produção de representação numérica dos textos (*embbedings*) já está configurado por padrão na arquitetura do BERT. Além disso, o BERT é treinado para entender textos em língua natural, sem remoção de *stop words* ou outro tipo de pré-processamento. Foi optado por utilizar a versão *uncased* do BERT, que ignora diferenças de maiúsculas e minúsculas, sendo que o próprio tokenizador do BERT faz a conversão do texto para letras minúsculas.

Um detalhe de arquitetura do BERT é que os textos de entrada podem possuir um tamanho máximo de 512 tokens após a tokenização (o que corresponde a aproximadamente 510 palavras descontando os tokens especiais) e textos acima desse tamanho são truncados. Isso pode representar uma dificuldade ao lidar com textos grandes, no entanto essa limitação não impactou nos resultados deste trabalho devido ao tamanho limitado de textos produzidos em comentários na internet. É possível ver pelo gráfico 10 que a maioria dos comentário foi limitado a 1000 caracteres, com o tamanho máximo de aproximadamente 2000 caracteres, de modo que nenhum texto ultrapassou os 512 tokens.

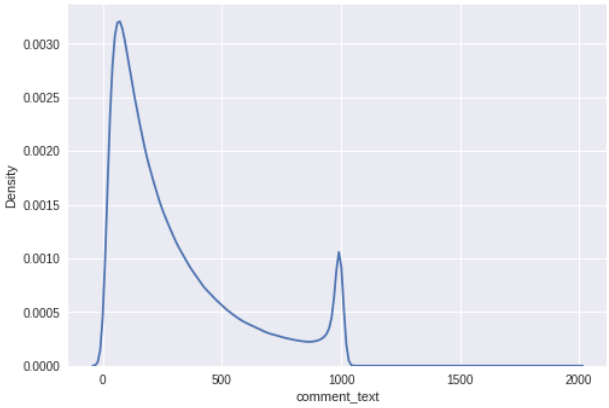

Fig. 9. Comprimento dos textos de comentários

VII. Configuração de experimentos

Em todas arquiteturas utilizadas foi utilizada a mesma divisão de dados para treino, teste e validação. O conjunto de dados original já possui uma divisão entre dados de treino e de teste, e essa divisão foi respeitada neste trabalho. No entanto, nem todos os dados foram utilizados. Para contornar o problema do desbalanceamento e o consequente viés de classificação seguindo a classe majoritária, nos dados de treino foi feito uma subamostragem, ou seja, dados da classe negativa foram removidos aleatoriamente até que ambas classes estivessem balanceadas. Isso foi feito por meio do *RandomUnderSampler* da biblioteca *imblearn*, usando $random\_state = 0$ para fins de reprodutibilidade. Desses dados subamostrados, 90% foram utilizados no treino dos modelos - correspondente a 283.813 amostras, e 10% foram utilizados para validação - correspondente 31.535 amostras.

O conjunto de dados de teste foi utilizado apenas para comparar os modelos que tiveram melhor desempenho na validação. Esse conjunto foi utilizado desbalanceado, de forma a simular um cenário real da utilização dos classificadores em que a classe positiva corresponde a apenas uma fração minoritária do total de dados. Foram utilizados todos os dados de teste para extrair esses resultados, num total de 191.787 amostras.

Os classificadores para a representação textual de TF-IDF utilizaram as configurações padrão do *scikit-learn*, sem grandes ajustes de hiperparâmetros. Apenas o classificador de regressão logística necessitou de um ajuste no número máximo de iterações. Esses classificadores foram treinados com 1-gramas (TF-IDF de palavras) e com n-gramas de tamanhos variando de 1 a 4 (1-4-gramas). Para os 1-gramas, também foram efetuados testes removendo caracteres não alfabéticos. Devido ao alto número de características produzidas ao gerar os 1-4-gramas, foram selecionadas apenas as 1000 características com maior valor f na análise de variância (seletor *KBest* do *scikit-learn*).

A rede neural convolucional usou como taxas de dropout 0.2, para a camada convolucional 1D foi usado kernel de tamanho 3, 64 filtros e função de ativação ReLU, a camada densa é composta por dois neurônios de saída (dado que os outputs foram convertidos para o formato [0,1] e [1,0]) e função de ativação sigmoid. A função de perda foi de cross-entropia binária, o otimizador usado foi o Adam com taxa de aprendizado 0.01 e também foi adicionada a técnica de early stopping quando não se aumentasse a acurácia em até 3 épocas, dado que o otimizador Adam ficou muito mais sujeito a overfitting do que o Gradiente estocástico descendente, levando cerca de 2-3 épocas para já ser notável o overfiting, mas ainda assim com resultados melhores.

Para a LSTM muitas partes são semelhantes à CNN: taxas de dropout de 0.2, otimizadores e função de perda e configuração de camada densa no mesmo formato que a CNN, com a diferença de que há a camada LSTM, com 100 células e função de ativação tanh (hyperbolic tangent).

A junção de LSTM e CNN segue as mesmas configurações das redes anteriores: A configuração da rede é a mesma que a CNN até gerar as saídas pelo maxpoolig global, agora passando elas como features para a LSTM de 100 unidades. Na sequência os resultados são passados para a camada densa com as mesmas características mencionadas anteriormente.

Para o classificador BERT, foi utilizado um batch size de 3 e taxa de aprendizado e 2×10^{-5}. O ajuste fino foi treinado por até 4 épocas, salvando os modelos intermediários.

VIII. Resultados obtidos

Todos os classificadores, em todas suas variações foram treinados e testados com os conjunto de dados balanceados de treino e validação. O resultado dessas execuções para a classe tóxico é apresentado na tabela I, para a classe não tóxico na tabela II e o resultado médio na tabela III.

Para o conjunto de dados desbalanceado de testes, foi elegido apena o classificador que apresentou melhor desempenho dentre os de TF-IDF - a regressão logística, além do classificador BERT a fim de traçar uma comparação entre as diferentes técnicas. A acurácia e as métricas para a classe "tóxico" são apresentados na tabela IV.

IX. Análise dos resultados

Todos algoritmos testados apresentaram bons resultados quando avaliados utilizando o dataset validação. Os três algoritmos que se saíram melhores nessa tarefa foram a regressão logística com representação textual de TF-IDF de 1-gramas (palavras), LSTM e BERT. Como esperado, o BERT foi o que obteve os maiores resultados, com acurácia, precisão, revocação e f1-score médios de aproximadamente $0,9$. No entanto, outros algoritmos mais simples obtiveram resultados bastante próximos, em particular a regressão logística obteve acurácia, precisão, revocação e f1-score médios de aproximadamente $0,87$.

Nos modelos de representação textual por TF-IDF, é possível notar que etapas adicionais de pré-processamento

TABLE I
Resultados para a classe tóxico - Dataset de Validação

Classificador	Precisão	Revocação	F1 score
naive bayes 1-gramas	0,762	0,848	0,803
naive bayes 1-grama apenas alfa	0,743	0,782	0,762
naive bayes 1-4-gramas	0,832	0,780	0,805
árvore de decisão 1-gramas	0,811	0,810	0,811
árvore de decisão 1-gramas apenas alfa	0,714	0,689	0,702
árvore de decisão 1-4-gramas	0,730	0,770	0,750
regressão logística 1-gramas	0,894	0,836	0,864
regressão logística 1-gramas apenas alfa	0,833	0,730	0,778
regressão logística 1-4-gramas	0,825	0,846	0,835
LSTM	0,845	0,832	0,839
CNN	0,863	0,772	0,815
CNN+LSTM	0,828	0,833	0,831
BERT	0,876	0,926	0,901

TABLE II
Resultados para a classe não-tóxico
Dataset de Validação

Classificador	Precisão	Revocação	F1 score
naive bayes 1-gramas	0,827	0,732	0,777
naive bayes 1-grama apenas alfa	0,767	0,727	0,747
naive bayes 1-4-gramas	0,791	0,840	0,815
árvore de decisão 1-gramas	0,809	0,810	0,809
árvore de decisão 1-gramas apenas alfa	0,697	0,722	0,709
árvore de decisão 1-4-gramas	0,755	0,713	0,733
regressão logística 1-gramas	0,845	0,900	0,872
regressão logística 1-gramas apenas alfa	0,758	0,852	0,802
regressão logística 1-4-gramas	0,841	0,841	0,830
LSTM	0,833	0,846	0,839
CNN	0,792	0,877	0,832
CNN+LSTM	0,831	0,825	0,828
BERT	0,921	0,868	0,894

TABLE III
RESULTADOS MÉDIOS - DATASET DE VALIDAÇÃO

Classificador	Acurácia	Precisão	Revocação	F1 score
naive bayes 1-gramas	0,791	0,794	0,790	0,790
naive bayes 1-grama apenas alfa	0,755	0,755	0,754	0,754
naive bayes 1-4-gramas	0,810	0,811	0,810	0,810
árvore de decisão 1-gramas	0,810	0,810	0,810	0,810
árvore de decisão 1-gramas apenas alfa	0,705	0,706	0,706	0,705
árvore de decisão 1-4-gramas	0,742	0,743	0,742	0,742
regressão logística 1-gramas	0,868	0,870	0,868	0,868
regressão logística 1-gramas apenas alfa	0,791	0,795	0,791	0,790
regressão logística 1-4-gramas	0,833	0,833	0,833	0,833
LSTM	0,839	0,845	0,832	0,838
CNN	0,824	0,863	0,772	0,815
CNN+LSTM	0,829	0,828	0,833	0,831
BERT	0,897	0,899	0,897	0,897

TABLE IV
RESULTADOS - DATASET DE TESTES

Classificador	Acurácia	Precisão	Revocação	F1 score
regressão logística	0,899	0,432	0,858	0,575
BERT	0,875	0,384	0,946	0,546

degradaram o desempenho dos classificadores. Em todos, as pontuação para 1-gramas foram similares ou superiores à pontuação para n-gramas de tamanho superior. Isso pode ser explicado pela quantidade exponencial características produzidas ao gerar os n-gramas. Mesmo com o uso de seletor de características, não foi possível encontrar características que produzissem um resultado melhor que as características dos 1-gramas. O algoritmo que mais sofreu com isso foi o de árvores de decisão, os outros dois algoritmos apresentaram resultados similares com a inclusão de característica adicionais.

Já a remoção de caracteres não-alfabéticos degradou o desempenho de todos os classificadores. Uma possível explicação é que informação valiosa foi perdida ao descartar esses caracteres, dado que a comunicação pela internet se utiliza de artefatos como emojis que ultrapassam os limites da linguagem tradicional.

No dataset balanceado não houve grande distinção nos resultados entre as classes "tóxico" e "não-tóxico", reflexo do treinamento ter sido feito com um dataset balanceado. O dataset desbalanceado de testes apresentou resultados bem diferentes. As métricas para a classe "não-tóxico" em geral foram bastante altas, devido à presença majoritária dessa classe de 92% dos exemplos. Assim, todas métricas para essa classe obtiveram valores acima de $0,9$. No entanto, para a classe "tóxico" a precisão foi bem mais baixa.

Ao treinar com o dataset balanceado, os modelos adquiriram alta sensibilidade à toxicidade, o que se reflete nas revocações comparativamente altas de $0,86$ para a regressão logística e $0,95$ para o BERT. Isso é particularmente verdadeiro para o BERT, que foi capaz de detectar aproximadamente 95% dos textos tóxicos. No entanto, essa sensibilidade exagerada diminui consideravelmente a precisão e aumenta muito o número de falsos positivos. Para o BERT, apenas 38% dos documentos identificados como tóxicos o eram de fato, e 62% foram falsos positivos. Para a regressão logística, foram 43% identificados corretamente e 57% de falsos positivos.

X. EXPLICABILIDADE

Dado que a regressão logística apresentou bons resultados, desenvolvemos um pequeno sistema para facilitar o acesso do público gerar e entendimento de como a solução é gerada, disponível em um aplicativo web [3]. Essa solução utiliza o *Local interpretable model-agnostic explanations* - LIME [8], que essencialmente utiliza perturbações no conjunto de dados (e neste caso, textos) para identificar proximidades entre as palavras relevantes para a classificação e assim poder realizar ponderações para os resultados finais.

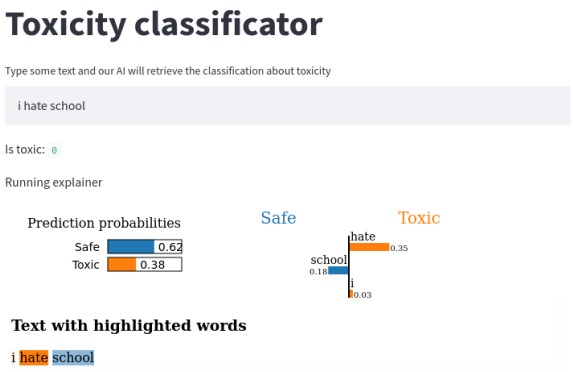

Fig. 10. Visualização do sistema para explicabilidade das classificações

Alguns testes podem facilitar a compreensão humana de como este modelo relativamente simples, porém poderoso para este problema, pode gerar suas conclusões. A presença do token *hate*, por exemplo, influencia bastante uma classificação para ser tóxica, mas dependendo das palavras no contexto, pode ter menor peso para o resultado final e gerarem uma justificativa melhor para a escolha de uma classificação pelo modelo. Fica também mais simples de se identificar vieses provenientes do comportamento dos dados de treino utilizando um explicador como este.

XI. ANÁLISE DE VIÉS

A fim de analisar a presença de viés nos classificadores, estes foram novamente avaliados utilizando os subconjuntos do dataset de testes que possuem anotações sobre os grupos mencionados no texto. Assim como a toxicidade, essa informação foi anotada por humanos e é representada pela fração dos avaliadores que consideraram haver menção ao grupo e valores

[3]Sistema disponível em https://toxic-texts.herokuapp.com/

maiores ou iguais a $0,5$ foram considerados positivos. Nesta análise, os classificadores foram avaliados usando os dados que possuem menção a cada um dos grupos com mais de 500 exemplos.

Para tentar identificar se há injustiça ao classificar textos que mencionam determinados grupos, medidas que levam em consideração o número de falsos positivos foram as que se mostraram mais relevantes. Considerando que a classe positiva é a "tóxica", os falsos positivos são os comentários não-tóxicos classificados incorretamente, representando assim uma injustiça por parte do modelo. A medida elegida para análise, portanto, foi a taxa de falsos positivos, que representa, neste caso, a fração de textos não-tóxicos classificados como tóxicos, e é dada pela equação 6, onde FP são os falsos positivos e TN são os verdadeiros negativos. Os valores da taxa de falsos positivos para os melhores classificadores em cada um dos grupos avaliados é apresentada na tabela V.

$$FPR = \frac{FP}{(FP + TN)} \tag{6}$$

Pela tabela V é possível ver que todos classificadores possuem viés similar, absorvido, portanto, do conjunto de dados. É possível observar que a taxa de falsos positivos para as classes "male" (masculino) e "female" (feminino) são parecidas, não havendo assim viés que favoreça um grupo em detrimento do outro. Infelizmente o conjunto de dados não possui amostras o suficiente para fazer a mesma análise para outras identidades de gênero. Quanto à sexualidade, o único grupo com dados para ser avaliado é o "homosexual_gay_or_lesbian", que apesar de não poder ser comparado com outros grupos similares, apresentou taxas de falso positivo alarmantemente altas acima de $0,5$, ou seja, mais da metade dos comentários não tóxicos que fazem menção a este grupo foram incorretamente classificados como tóxicos. Isso reflete o fato de muitos dos comentários utilizarem homossexualidade como ofensa, viés absorvido pelos modelos.

Dentro dos grupos que fazem menção a religiões, o grupo mais injustiçado foi o de muçulmanos - "muslim", apresentando também taxas acima de 0.5. O grupo "jewish" (judeus) apresentou taxas menores, na faixa de 0.35 e o grupo "christian" (cristão) apresentou a menor taxa de falsos positivos avaliada, inferior a $0,2$.

Para os grupos que fazem menção a cor de pele "black" e "white", ambos apresentaram taxas extremamente elevadas acima de 0.6, mas equiparáveis. Assim, os modelos absorveram um viés de tratar qualquer menção a cor de pele como toxicidade, mas sem favorecer ou prejudicar em particular pretos ou brancos. O classificador BERT foi o que mais adquiriu esse viés, com taxas até $0,9$ superiores. Finalmente, para o grupo de doenças psiquiátricas ou mentais não é possível traças comparações com outros grupos, mas o viés foi particularmente mais acentuado para o classificador de regressão logística, com uma taxa de 0.4.

XII. Conclusão

Este trabalho teve por objetivo treinar e avaliar classificadores de toxicidade em textos de comentários na internet. Os classificadores desenvolvidos tiveram um bom desempenho quando avaliados em um dataset balanceado. Como esperado, o classificador BERT em geral teve os melhores resultados ao fazer uso de um pré-treinamento intensivo. No entanto, o classificador de regressão logística com representação textual em TF-IDF obteve desempenho bastante similar, sendo consideravelmente mais simples de ser treinado e modificado. Para fins práticos a regressão logística pode se mostrar uma opção melhor e mais flexível.

Ao avaliar os modelos em um dataset desbalanceado, cenário mais próximo do ambiente real, os modelos apresentaram sensibilidade muito alta, gerando muitos falsos positivos. Assim, os modelos aqui treinados podem ser um bom indicador de casos suspeitos de toxicidade, conseguindo encontrar a grande maioria dos comentários tóxicos, mas não podem ser utilizados como único critério para penalizar tais comentários, sendo necessário o uso combinado de outros classificadores ou avaliação humana.

Essa reavaliação torna-se particularmente necessária devido ao problema do viés. Termos relacionados aos homossexuais, muçulmanos ou termos associados à cor de pele em particular produzem uma grande quantidade de falsos positivos e são um ponto de alerta.

Em trabalhos futuros, espera-se poder treinar estes modelos utilizando o dataset de treino completo, sem a utilização de subamostragem para balanceamento, esperando obter modelos mais precisos em identificar toxicidade, mas potencialmente menos sensíveis. Já a técnica de subamostragem pode ser utilizada para balancear a quantidade de comentários tóxicos e não tóxicos dentro de cada grupo de identidades, possivelmente reduzindo o viés e produzindo classificadores mais justos.

References

[1] S. V. Georgakopoulos, S. K. Tasoulis, A. G. Vrahatis, and V. P. Plagianakos, "Convolutional neural networks for toxic comment classification," in *Proceedings of the 10th Hellenic Conference on Artificial Intelligence*, ser. SETN '18. New York, NY, USA: Association for Computing Machinery, 2018. [Online]. Available: https://doi.org/10.1145/3200947.3208069

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017.

[3] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018. [Online]. Available: http://arxiv.org/abs/1810.04805

[4] V. S. S. Settipalli and N. M. K. Dasireddy, "Reducing unintended bias in text classification using multitask learning." 2021.

[5] I. Gunasekara and I. Nejadgholi, "A review of standard text classification practices for multi-label toxicity identification of online content," in *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 21–25. [Online]. Available: https://aclanthology.org/W18-5103

[6] N. Chakrabarty, "A machine learning approach to comment toxicity classification," 2019.

[7] T. Pranckevicius and V. Marcinkevičius, "Application of logistic regression with part-of-the-speech tagging for multi-class text classification," 11 2016, pp. 1–5.

TABLE V
TAXA DE FALSOS POSITIVOS POR GRUPO

Classificador	**male**	**female**	**homosexual**	**christian**	**jewish**	**muslim**	**black**	**white**	**psychiatric**
reg. logística	0,29	0,25	0,53	0,16	0,35	0,50	0,63	0,60	0,40
BERT	0,30	0,28	0,65	0,20	0,37	0,55	0,72	0,68	0,32

[8] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should I trust you?": Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, 2016, pp. 1135–1144.