Lince: um arcabouço para ofuscação de estilo de escrita em textos

Antônio M. R. Franco[1], **Ítalo F. S. Cunha**[1], **Leonardo B. Oliveira**[1]

[1]Universidade Federal de Minas Gerais

{franco,cunha,leob}@dcc.ufmg.br

Abstract. *Anonymity is paramount for the safety and protection of journalists and whistleblowers reporting frauds and corruption scandals. Most of the tips nowadays are submitted over the Internet, and currently, there are many ways to navigate anonymously. However, one can still identify anonymous users by their writing style. With the advances of neural networks and natural language processing, the accuracy of text classifiers to identify authorship attributes is increasing. On the other hand, new approaches have been proposed to fight those adversaries. In this work, we aim to come up with a framework for authorship obfuscation. We evaluated two approaches for authorship obfuscation and proposed changes to improve text generation quality and make it for non-tech users to use it. Such change improved the text quality by up to 20%, keeping the adversary performance lower than the chance level.*

Resumo. *O anonimato é essencial para a segurança física de jornalistas e denunciantes em geral que submetem denúncias através da Internet. Atualmente, existem abordagens para se obter anonimato online, no entanto, usuários anônimos ainda podem ser identificados pelo seu estilo de escrita. A chance de sucesso de um classificador identificar corretamente o autor de um texto tem crescido cada vez mais com o avanço das pesquisas em processamento de linguagem natural. Por outro lado, novas abordagens para geração automática de textos ofuscados também têm surgido para combater os adversários do anonimato na Internet. Neste trabalho, objetivamos conceber um arcabouço para ofuscação de autoria de textos. Para isso, avaliamos duas abordagens para ofuscação de autoria de textos e propomos melhorias para otimizar a qualidade do texto gerado pelos ofuscadores e para facilitar o uso para usuários não-técnicos. Tais melhorias resultaram em um aumento de até 20% na qualidade das sentenças geradas, enquanto mantiveram a taxa de sucesso do adversário abaixo do nível de chance.*

1. Introdução

Há vários cenários em que o anonimato é relevante na Internet. Um exemplo corriqueiro que requer anonimato são sistemas de avaliação onde colaboradores de uma empresa avaliam seus pares ou onde pesquisadores revisam artigos científicos. Em tais sistemas, os participantes se sentem mais propensos a submeterem avaliações honestas quando sabem que o sigilo será garantido. Outro exemplo são denunciantes que utilizam canais de denúncias em geral, como jornalistas que denunciam práticas de corrupção ou funcionários relatando fraudes para a ouvidoria de uma empresa. Em casos críticos (e.g.,

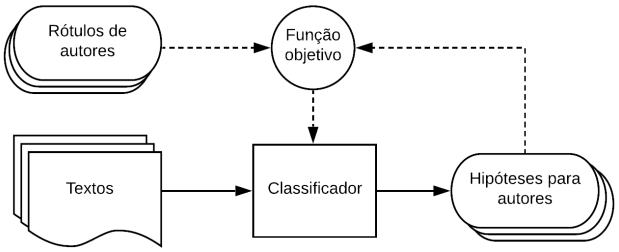

Figura 1. Exemplo de um classificador de textos para identificação de autoria.

exposição de casos de corrupção de governos[1]), inclusive, o anonimato pode ser crucial para preservar a integridade física do denunciante.

Atualmente existem ferramentas que provêm algum nível de anonimato na Internet. Uma lista de ferramentas e boas práticas de privacidade online pode ser encontrada na página do projeto *PrivacyTools*[2]. Dentre as técnicas e ferramentas disponíveis, é possível encontrar desde recomendações para evitar que um navegador seja rastreado pela Internet até abordagens para mascarar o endereço de origem de um acesso (e.g., a rede Tor[3]). Tais ferramentas, no entanto, assumem que os usuários não revelarão sua identidade no conteúdo publicado. Por exemplo, é razoável assumir que o autor de um texto não irá assiná-lo com o seu nome. No entanto, mesmo um texto apócrifo pode ter sua autoria revelada através de atributos implícitos.

Um destes atributos é o estilo de escrita do autor. Um adversário pode analisar o estilo com que o autor escreve com o objetivo de inferir a sua identidade. Especialistas em linguística costumam aplicar, manualmente, técnicas de estilometria para reconhecer atributos específicos que caracterizam o estilo de escrita de um autor. A área que estuda estas técnicas é chamada linguística forense, e investigadores já as aplicaram, por exemplo, no famoso caso do Unabomber [Fitzgerald 2004]. Indo além de abordagens manuais, um adversário pode treinar um classificador utilizando técnicas de aprendizagem supervisionada para inferir a identidade do autor de textos. Como é ilustrado na Figura 1, um adversário pode treinar um classificador de autor de textos com um conjunto de dados composto de textos rotulados com seus respectivos autores. Neste caso, o treinamento do classificador deve minimizar uma função objetivo que compara as hipóteses de autores retornadas pelo modelo com os rótulos reais.

Este trabalho objetiva conceber um arcabouço para aplicar transformações automáticas em textos de modo a minimizar a chance de acerto de um adversário que faz uso de classificadores de autor de texto. Tal arcabouço deve ser acessível para um público sem conhecimentos técnicos e, portanto, deve requerer o mínimo de entradas possíveis do usuário. Idealmente, os autores usuários deste arcabouço devem informar apenas um texto de entrada a ser transformado, sem que seja necessário criar regras pré-definidas de transformação ou selecionar autores específicos para copiar o estilo de escrita.

Neste trabalho, nós avaliamos duas técnicas de ofuscação de textos baseadas pu-

[1] https://www.nytimes.com/2019/03/18/world/africa/south-africa-anc-magaqa-killing-arrests.html

[2] https://www.privacytools.io

[3] https://www.torproject.org

ramente em dados que não requerem regras pré-definidas. Em nossa avaliação, nós treinamos modelos para ofuscar o estilo de escrita de cinco edições da bíblia em inglês. Propomos também uma técnica de treinamento que mescla diferentes estilos de escrita para ofuscar os textos, viabilizando assim o uso prático de modelos de ofuscação que requerem que seja informado um estilo de escrita alvo. Tal técnica evita que um indivíduo em particular seja atribuído indevidamente como autor de um texto, evitando uma possível acusação de um inocente. Além disso, esta técnica faz com que o escritor que busca ofuscar o seu texto não tenha que escolher um estilo em particular para imitar. Propomos também duas estratégias para treinar modelos com baixo volume de dados: aplicar tokenização por sub-palavras e usar vetores de palavras pré-treinados. Ao aplicar tais técnicas, a perda de qualidade textual foi 20% menor, enquanto o desempenho do adversário foi mantido próximo ao nível de chance. Por fim, desenvolvemos uma aplicação web com uma API REST para facilitar a implementação de modelos de ofuscação de textos.

2. Conceitos fundamentais

Nesta seção nós introduzimos os conceitos e notações necessárias para compreender o problema de identificação de autoria e as suas contramedidas.

Processamento de Linguagem Natural. Analisar e transformar textos por meio de algoritmos requer técnicas de Processamento de Linguagem Natural (NLP) [Indurkhya and Damerau 2010]. Uma tarefa comum no contexto de NLP é a classificação de textos. Tal tarefa consiste em treinar um modelo com um conjunto de dados rotulado e utilizar o modelo para tentar predizer a classe de novos exemplos que não estavam nos dados de treino. Cada exemplo de entrada do modelo consiste, tipicamente, em uma sentença ou um documento. Já o rótulo pode ser utilizado para representar qualquer característica do texto. Por exemplo, se o objetivo da tarefa de classificação for uma análise de sentimentos, o rótulo informará se um texto é negativo ou positivo.

Redes Neurais Recorrentes. Rede Neural Recorrente (*Recurrent Neural Network* – RNN) [Rumelhart et al. 1986] é um tipo de arquitetura de rede neural adequada para lidar com séries temporais. Elas conseguem representar tais séries melhor do que uma rede neural tradicional (e.g., uma rede perceptron multicamadas) porque elas memorizam as características aprendidas a partir dos passos anteriores. Uma sentença pode ser analisada como uma série temporal [Graves 2013], já que a ordem das palavras é relevante para o seu significado. Por exemplo, uma rede perceptron multicamadas poderia ser implementada para identificar locais a partir de sentenças escritas. Neste caso, como o tamanho das sentenças pode ser variável, nós teríamos que definir um tamanho fixo de parâmetros (que seria o tamanho da maior sentença). Se a rede receber a sentença "Eu estive no Brasil em 2019."e "Em 2019, eu estive no Brasil.", é esperado que ela seja capaz de identificar "Brasil"em ambas as sentenças. Neste caso, como uma rede perceptron multicamadas possui parâmetros separados para cada posição da entrada, é difícil aprender que o mesmo local aparece em posições diferentes nas duas sentenças. Por outro lado, RNN resolve este problema implementando compartilhamento de parâmetros. Assim, ao processar uma dada palavra, a rede terá armazenado os parâmetros aprendidos nos passos anteriores. Matematicamente, a cada passo t, uma RNN recebe como entrada um vetor $x^{<t>}$ e um estado oculto $a^{<t-1>}$ e retorna um vetor de saída $\hat{y}^{<t>}$ e um novo estado oculto $a^{<t>}$. No primeiro passo, como não há um estado oculto anterior, $a^{<t-1>}$ começa com zeros.

Tokenização por sub-palavras. RNNs esperam números como entrada e, portanto, é necessário converter os textos em uma representação numérica. Uma abordagem comum é separar os *tokens* de um texto e construir um vocabulário onde cada *token* é convertido para um número distinto. O processo de tokenização pode ser feito por caracteres, por palavras ou por sub-palavras. Tokenização por caracteres resulta em um alto número de *tokens* por sentença em comparação a separação por palavras. Por outro lado, o tamanho do vocabulário é significativamente maior para o segundo caso. Um bom equilíbrio entre os dois métodos é a tokenização por sub-palavras, que pode ser implementada através de algoritmos como o Byte-pair Encoding (BPE)[Tacorda et al. 2017].

Modelos sequência-para-sequência. Uma outra tarefa de NLP consiste em mapear uma sequência em outra. Isso pode ser útil, por exemplo, para traduzir uma sentença de um idioma para outro. Outro exemplo é a sumarização de textos, onde um documento é dado como entrada e o objetivo do modelo é retornar o seu resumo. Em ambos os casos, o texto é representado como uma sequência de tokens (e.g., palavras e pontuação) e os modelos aplicados para resolver este problema são chamados *seq2seq* (do inglês *sequence-to-sequence* – sequência-para-sequência)[Sutskever et al. 2014]. Modelos *seq2seq* podem ser implementados por meio de abordagens diferentes. Uma abordagem bem conhecida é o uso de RNN em uma arquitetura codificador-decodificador. Neste caso, duas RNNs são empregadas para construir um codificador - que é responsável por representar uma sentença de entrada em um vetor chamado vetor de contexto - e um decodificador - que é responsável por produzir uma sentença de saída a partir do vetor de contexto.

Transferência de aprendizado. Em aprendizado de máquina, transferência de aprendizado consiste em transferir características aprendidas em um domínio para outro [Pan and Yang 2009]. Aplicando isso ao contexto de NLP, é possível treinar um modelo de linguagem em um corpus grande e de propósito geral (e.g., a Wikipédia) e depois utilizar os pesos deste modelo pré-treinado para iniciar o treinamento sobre um outro corpus específico (e.g., um conjunto de textos jurídicos). Ao adotar esta abordagem, nós podemos esperar que o modelo aproveite características gerais que foram aprendidas anteriormente (e.g., estrutura das sentenças, sintaxe e tempos verbais) e aprenda características mais específicas com o novo corpus [Howard and Ruder 2018]. Desta forma, é possível obter um bom desempenho mesmo em um conjunto de dados com poucos exemplos.

Redes Adversárias Generativas. Além de aprender as distribuições de probabilidade que representam como os dados são segmentados, há um outro tipo de modelo chamado modelos generativos que tem como objetivo aprender a distribuição dos dados em si. Este tipo especial de modelo pode ser utilizado, por exemplo, para gerar imagens, textos e áudios sintéticos. Um exemplo de modelos generativos são as Redes Adversárias Generativas (*Generative Adversarial Network* – GAN) [Goodfellow et al. 2014]. GAN é um arcabouço de treinamento utilizado para treinar modelos generativos adversariamente. Este arcabouço é composto de duas redes neurais: uma com o papel de gerador e outra com o papel de discriminador. O gerador tenta enganar o discriminador gerando exemplos que parecem reais e o discriminador tenta classificar estes exemplos como reais (que existem no conjunto de dados) ou falsos (produzidos pelo gerador).

Identificação de autoria. Identificação de autoria é a tarefa de inferir o autor de um texto apócrifo. Um dos métodos mais básicos para identificar o autor de um texto a partir do

seu estilo de escrita é a análise de frequência de palavras [Mosteller and Wallace 1963]. Outros métodos são baseados em modelos de aprendizado de máquina para classificação de textos e utilizam, por exemplo, classificadores como SVM e representação de *n-grams* contínuos [Varela et al. 2011, Sari et al. 2017, Narayanan et al. 2012]. Há também classificadores de autoria que utilizam modelos de aprendizagem profunda. Nesta linha, é possível aplicar arquiteturas como Rede Neural Convolucional (*Convolutional Neural Network* – CNN)[Ruder et al. 2016] e RNNs [Bagnall 2015].

3. Trabalhos relacionados

Nesta seção, discutimos os trabalhos relacionados à ofuscação de autoria. Tais trabalhos objetivam transformar um dado texto para esconder os atributos estilísticos que um adversário poderia analisar para descobrir a identidade do autor. Primeiramente, nós analisamos trabalhos que cobrem métodos de ofuscação que sugerem alterações no texto em vez de aplicá-las diretamente. Na sequência, analisamos trabalhos que abordam ofuscação de autoria através de regras de transformação. Por fim, analisamos trabalhos que aplicam técnicas de inteligência artificial para ofuscar textos automaticamente sem a necessidade de regras pré-definidas.

[McDonald et al. 2012] foi um dos primeiros trabalhos de relevância sobre ofuscação de autoria. Os autores propõem uma abordagem semi-automática baseada em um algoritmo de *k-means* que é capaz de analisar um conjunto de textos e compará-lo com um novo texto candidato a ofuscação. O método destaca as palavras que o autor deve alterar para reduzir as chances de um adversário predizer corretamente o autor do texto ofuscado utilizando um classificador de textos. Este método, no entanto, não aplica a ofuscação de autoria automaticamente, e portanto o autor é responsável por modificar as partes do texto que são apontadas e escolher novas palavras para substituí-las. Como os próprios autores apontam no artigo, a ferramenta Anonymouth é apenas o primeiro passo em direção a uma ferramenta de ofuscação de autoria, e portanto é necessária uma pesquisa futura.

[Karadzhov et al. 2017] propõem uma técnica baseada em regras como uma abordagem para ofuscação de autoria. A abordagem aplica substituição de palavras e símbolos comuns, que são escolhidos após uma análise estatística prévia. Além deste trabalho, [Potthast et al. 2016] avaliaram a eficiência de três métodos de ofuscação [Keswani et al. 2016, Mansoorizadeh et al. 2016, Mihaylova et al. 2016] submetidos para o PAN[4] 2016. Todos estes trabalhos são baseados em regras que foram escritas para inglês, e portanto não funcionam para textos escritos em outros idiomas.

[Shetty et al. 2018] propõem uma nova abordagem para ofuscação automática de textos. Ao contrário de outras abordagens que utilizam regras pré definidas para aplicar transformações nos textos, eles propuseram um modelo generativo treinado adversariamente para transformar textos com o objetivo de modificar o seu estilo de escrita e assim esconder a identidade do autor original. Além de fornecer um mecanismo para ofuscação automática, esta abordagem se baseia apenas em atributos aprendidos diretamente dos dados. Duas redes neurais são utilizadas para treinar o modelo proposto por eles: a primeira é um classificador construído para identificar os atributos de estilística e predizer o autor

[4]PAN é uma série de eventos científicos e tarefas compartilhadas no campo de forense de textos digitais e estilometria.

de um texto, e a segunda é uma rede codificador-decodificador construída para transformar sentenças de um estilo em outro, objetivando enganar o classificador. Elas são treinadas de maneira adversária utilizando o arcabouço GAN [Goodfellow et al. 2014].

[Emmery et al. 2018] propõem um método para ofuscação chamado ofuscação por invariância. Para conseguir ofuscação de autoria, os autores acoplam uma Camada de Reversão de Gradiente (*Gradient Reversal Layer*) (*Gradient Reversal Layer* – GRL) à saída de um codificador construído com uma RNN. A saída do codificador então passa por essa camada GRL antes de ser utilizada como entrada em uma camada totalmente conectada que é treinada para predizer o autor de uma sentença. A ideia geral é que a camada GRL faça com que o codificador aprenda a representar sentenças que são livres de atributos de autoria. A representação do codificador é então passada para um decodificador que é treinado fim-a-fim com o codificador em uma tarefa de auto-codificação, onde o objetivo do decodificador é gerar a mesma sentença que o codificador recebeu como entrada.

[Gröndahl and Asokan 2019] combinaram a ideia de ofuscação baseada em regras com uma abordagem baseada em inteligência artificial e conceberam novo método para ofuscação de autoria. O método consiste em utilizar regras de transformação para gerar sentenças candidatas e em seguida submetê-las a um classificador de textos treinado para identificar autoria. A sentença que fizer com que o classificador de textos tenha a menor acurácia é então escolhida para compor o texto final. Como esta abordagem faz uso de regras pré-definidas, é necessário um esforço considerável para aplicá-la em outros idiomas além do inglês.

[Bo et al. 2019] propõem uma abordagem para ofuscação que não requer rótulos de autores no conjunto de dados. Os autores também fazem uso das ideias de privacidade diferencial propostas em [Fernandes et al. 2019] para estabelecer um nível mínimo de anonimato. O método é composto por um modelo auto-codificador com um mecanismo exponencial. A principal diferença com relação aos métodos propostos anteriormente é que, durante o passo de geração das hipóteses candidatas para uma sentença, a amostragem é feita utilizando privacidade diferencial em vez de simplesmente aplicar a função `argmax`. Além disso, o auto-codificador é treinado com uma função de recompensa em uma tarefa de aprendizado por reforço que tem como objetivo maximizar a preservação semântica e sintática da sentença gerada.

[Mahmood et al. 2019] conceberam um método para ofuscação de estilo de escrita chamado Mutant-X. Este método é baseado em um algoritmo genético que introduz alterações em um texto usando técnicas de mutação com o objetivo de reduzir a acurácia de um classificador de textos e maximizar uma função que mede a similaridade semântica. Os autores utilizaram uma rede neural recorrente como classificador de autoria e a função METEOR para medir a similaridade semântica.

[Bevendorff et al. 2020] propõe uma nova abordagem para ofuscação de texto. Nesta abordagem, o ofuscador explora um espaço de variantes de um texto até encontrar uma paráfrase que maximize a distância de Jensen-Shannon com o menor custo possível em perda de qualidade.

Comparado aos trabalhos anteriores, o nosso trabalho propõe um arcabouço com diferentes abordagens de ofuscação de textos. Tal arcabouço facilita o acesso aos usuários

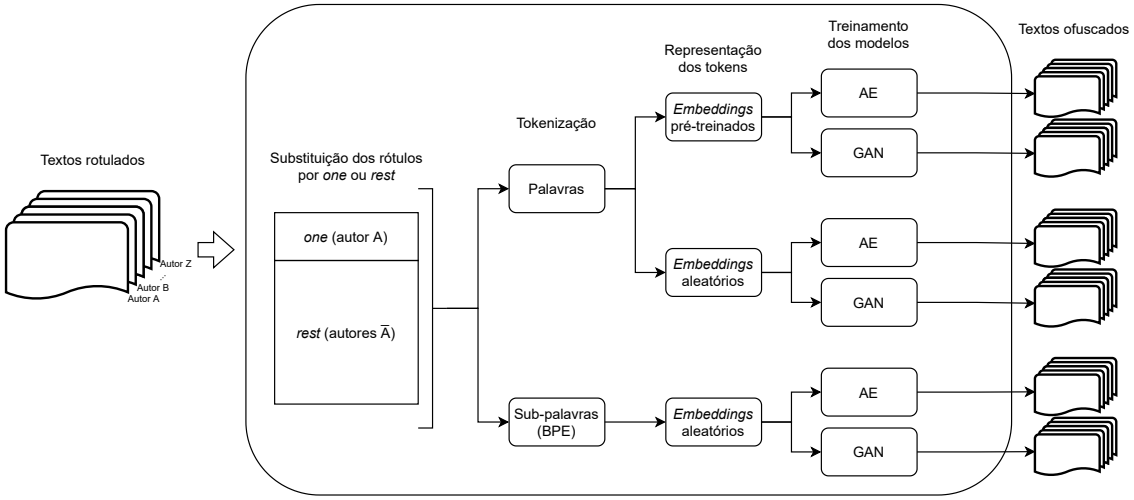

Figura 2. Treinamento dos modelos para ofuscação de texto.

não técnicos pois não requer a indicação de estilos de escrita específicos para transformar o texto e nem a criação de regras estáticas de transformação. Além disso, nosso trabalho também contempla uma aplicação *web* que visa disponibilizar uma interface intuitiva para os usuários.

4. Desenvolvimento

Nesta seção nós descrevemos o conjunto de dados que utilizamos nos nossos experimentos, os detalhes de implementação de cada modelo e a metodologia que utilizamos para avaliar os métodos de ofuscação.

4.1. Visão geral

O treinamento dos modelos foi realizado conforme ilustrado na Figura 2. Dado um conjunto de textos rotulados com seus respectivos autores, o primeiro passo consiste em substituir os rótulos originais aplicando uma estratégia *one-vs-rest*. Nesta substituição, os textos de um autor A receberiam o rótulo *one* enquanto os textos dos demais autores seriam rotulados como *rest*. Na sequência vem a etapa de tokenização, onde aplicamos tokenizadores baseados em palavras e baseados em BPE. Uma vez separados em *tokens*, é preciso converter cada *token* em uma representação numérica. Nesta etapa, cada *token* é representado como um vetor (também conhecido como *embedding*) e os valores de cada dimensão do vetor são iniciados aleatoriamente ou extraídos de uma lista de *embeddings* pré-treinados. Por fim, treinamos os dois modelos avaliados para cada conjunto de pares de tokenização e representação de *tokens*. Após o treinamento, usamos cada um dos modelos para gerar os textos ofuscados e avaliar o desempenho de cada modelo.

4.2. Conjunto de dados

Nós utilizamos um corpus composto por cinco edições da bíblia em inglês. As edições contempladas foram a *American Standard Version* (ASV), *Bible in Basic English* (BBE), *Darby Bible* (DBY), *World English Bible* (WEB) e *Young's Literal Translation* (YLT). Cada versículo foi considerado como uma sentença diferente. Cada edição possui ~ 31000 sentenças, totalizando ~ 155000 sentenças para o corpus inteiro. A sentença mais

(a) por palavras. **(b) por BPE.**

Figura 3. Similaridade de tokens entre os autores.

curta tem 5 *tokens*, enquanto a mais longa tem 100. Nós apresentamos as estatísticas do conjunto de dados na Tabela 1. Além disso, nós mostramos o percentual de *tokens* compartilhados entre os diferentes edições na Figura 3 (com e sem a utilização de BPE).

Autor	# sentenças	# tokens	# tokens únicos	tam. méd.	tam. mín.	tam. máx.
ASV	31025	924338	13618	29	4	100
BBE	31025	961868	6729	31	4	100
DBY	30952	919945	14787	29	4	100
WEB	31024	910618	14064	29	4	100
YLT	31021	961762	13803	31	4	100

Tabela 1. Descrição do conjunto de dados.

4.3. Modelagem de ataque

Nós partimos das premissas a seguir para modelar os ataques possíveis contra as abordagens de ofuscação de texto. É assumido que o autor do texto não irá inserir nenhuma informação que possa identificá-lo explicitamente, como o seu nome, sua localização, ou outro atributo que ajude a revelar a sua identidade. Além disso, é assumido que o adversário tem acesso ao texto ofuscado e, eventualmente, a outros textos não-ofuscados que o autor tenha tornado públicos e que possuem seus atributos de identidade. O ofuscador de textos também deve ser resiliente contra ataques de de-ofuscação. Assim, nós assumimos como premissa que tanto o ofuscador quanto os textos ofuscados são públicos, mas o texto não ofuscado é mantido em segredo. Desta forma, a tarefa do adversário será inferir o atributo de identidade I correspondente ao autor A tendo acesso somente ao texto ofuscado O e a um conjunto de textos T que pode incluir textos escritos pelo autor em questão.

4.4. Modelos para ofuscação de textos

Nós implementamos diferentes modelos para ofuscação de textos baseados em duas abordagens: ofuscação por transferência de estilo e ofuscação por invariância. Os modelos de ofuscação por transferência de estilo transformam uma sentença de um autor A para fazer com que o seu estilo de escrita pareça com o de um autor B. Por outro lado, a abordagem por invariância faz com que o codificador aprenda a representar sentenças suprimindo os atributos que apontam para o estilo de escrita, e desta forma cabe ao decodificador apenas produzir sentenças a partir de um vetor de contexto neutro.

A abordagem de ofuscação por transferência de estilo foi implementada utilizando o arcabouço GAN seguindo a abordagem de [Shetty et al. 2018]. O gerador é uma rede

codificador-decodificador que possui múltiplos decodificadores – cada um treinado para gerar sentenças em um estilo diferente. Precisamente, a rede possui N decodificadores, onde N é o número de autores. No nosso caso, como adotamos a abordagem *one-vs-rest*, temos 2 decodificadores. O discriminador, por sua vez, é um classificador de textos treinado com N rótulos. Ambos gerador e discriminador são treinados como adversários: o primeiro tenta produzir sentenças para enganar o discriminador, e o segundo tenta melhorar sua predição quando é enganado pelo gerador. Ao final do treinamento, espera-se que o gerador seja capaz de enganar o discriminador e, portanto, transformar sentenças de um domínio A para um domínio B fazendo com que elas sejam classificadas como pertencentes a outro autor.

Em seguida, nós implementamos os modelos de ofuscação por invariância baseando-se na abordagem de [Emmery et al. 2018]. A invariância neste caso é aprendida por um componente chamado GRL [Ganin and Lempitsky 2015]. Assim como na abordagem de ofuscação por transferência de estilo, é utilizada uma arquitetura codificador-decodificador para transformar a sentença de origem em uma sentença alvo. A diferença, no entanto, está no treinamento do modelo. Neste caso, em vez de utilizar o arcabouço GAN, a saída do codificador é passada para o componente GRL, e a saída da camada GRL é passada para rede perceptron multicamadas. Durante a propagação (*forward pass*), a camada de GRL funciona como uma função identidade: ela apenas retorna a mesma entrada que é passada como argumento. No entanto, durante a fase de retro-propagação (*backpropagation*), ela inverte o sinal dos gradientes computados pela função de perda, direcionando o otimizador na direção oposta. Isso faz com que o codificador seja penalizado toda vez que o estilo de escrita for predito corretamente pelo classificador, e portanto o encoraja a aprender uma representação neutra em termos de estilo de escrita.

Nós também implementamos melhorias em ambos os modelos. O primeiro ponto que nós mudamos foi o passo de tokenização. Tokenização é o processo de transformar uma sentença em tokens, que por sua vez podem ser convertidos em números para serem utilizados como entrada para uma modelo de aprendizagem profunda. As propostas de ofuscação originais aplicaram uma tokenização a nível de palavra, onde cada palavra (ou símbolo de pontuação) é convertido em um token. O problema desta abordagem é que o modelo é incapaz de aprender palavras que não estão presentes no conjunto de dados e acaba substituindo-as por um token especial para representar palavras desconhecidas. Isso afeta consideravelmente o desempenho do modelo, sobretudo quando o vocabulário do conjunto de dados é restrito. Uma alternativa é a tokenização por caracteres. Neste caso, cada símbolo é convertido para um token e uma sentença é representada pela combinação de tokens para cada um de seus caracteres. Esta abordagem resolve o problema de palavras desconhecidas, mas é computacionalmente cara. Um meio termo é a tokenização por sub palavras, e nós a implementamos utilizando o algoritmo de codificação de pares de bytes (BPE) com um limite de 10000 tokens.

Outra modificação que propomos é a aplicação de transferência de aprendizado para aproveitar os pesos de vetores de palavras pré-treinados. Nos modelos com esta melhoria, nós definimos os vetores de palavras com os valores disponibilizados por [Hartmann et al. 2017] em vez de iniciá-los aleatoriamente. O objetivo desta modificação é utilizar transferência de aprendizado para aproveitar as características aprendidas durante o treinamento prévio dos vetores.

No total, nós treinamos trinta modelos diferentes de ofuscação distribuídos conforme descrevemos a seguir. Para cada abordagem de ofuscação nós treinamos três modelos: aplicando tokenização por palavras com e sem *embeddings* pré-treinados; e aplicando BPE. Nota-se que *embeddings* pré-treinados foram utilizados apenas nos modelos com tokenização por palavras, pois eles são compatíveis apenas com palavras inteiras. Este processo se repetiu para cada autor do conjunto de dados já que adotamos a estratégia *one-vs-rest*. Assim, como há três modelos para cada uma das duas abordagens e este processo se repete para cada uma das cinco classes de autores, o resultado é um total de trinta modelos distintos.

Além disso, nós introduzimos ruídos controlados na saída do codificador para incentivar a substituição de palavras. Os ruídos testados foram extraídos de distribuições normais com seis valores diferentes para o desvio padrão: 0,00 (sem ruído), 0,01, 0,05, 0,10, 0,15 e 0,20. A motivação para o uso de ruído, conforme discutido em [Emmery et al. 2018], é fazer com que o decodificador aumente a variação de palavras nas sentenças geradas.

Por fim, nós implementamos uma aplicação web para disponibilizar o acesso aos modelos de ofuscação de textos sem o uso de interfaces de linha de comando. Tanto o código da aplicação web quanto dos modelos foram escritos em Python. Usamos o microframework Flask como base para a aplicação web e o *framework* PyTorch como base para os modelos. Todos os códigos estão disponíveis em um repositório público[5].

5. Avaliação

Nós utilizamos duas métricas para avaliar os modelos de ofuscação. A primeira é o F1-score do adversário, que usamos para medir o desempenho do adversário após aplicar um ofuscador em um texto. Para obter tal métrica, implementamos um classificador baseado na abordagem de [Sari et al. 2017] para atuar como adversário. Treinamos o modelo de classificação de autor de textos com o nosso conjunto de dados e obtivemos F1-scores de 0,81, 0,82, 0,78, 0,80 e 0,83 para textos das edições ASV, BBE, DBY, WEB e YLT respectivamente. Estes valores são superiores à chance de uma inferência aleatória, que é de 0,20 para cada classe. Aqui é possível estabelecer uma linha de base: um bom ofuscador deve reduzir o F1-score do adversário próximo ao nível de uma inferência. Outra métrica avaliada foi o METEOR score[Banerjee and Lavie 2005]. Tal métrica é usada para medir a similaridade entre duas sentenças e nós a selecionamos por conta de sua correlação com o julgamento humano. Neste caso, o objetivo é mensurar o quão similar a sentença ofuscada é da sentença original. A implementação que utilizamos para medir o METEOR está disponível na biblioteca NLTK [Bird et al. 2009]. Para cada sentença ofuscada, aplicamos o classificador adversário para obter a sua predição de autor e calculamos o valor de METEOR. Ao final, nós calculamos o F1-score com base nas predições do adversário e a média do METEOR de todas as sentenças.

Antes de comparar as abordagens de ofuscação entre si, avaliamos se a estratégia *one-vs-rest* gera resultados diferentes quando comparada com *one-vs-one*. Esta comparação é apresentada nas Figura 4. Nota-se que o desempenho do adversário é $\sim 35\%$ melhor quando se usa *one-vs-rest* sem ruído. No entanto, o desempenho cai para próximo do nível de inferência quando é aplicado um ruído de 0,20 em ambas as

[5] `https://gitlab.com/antoniomrfranco/lince`

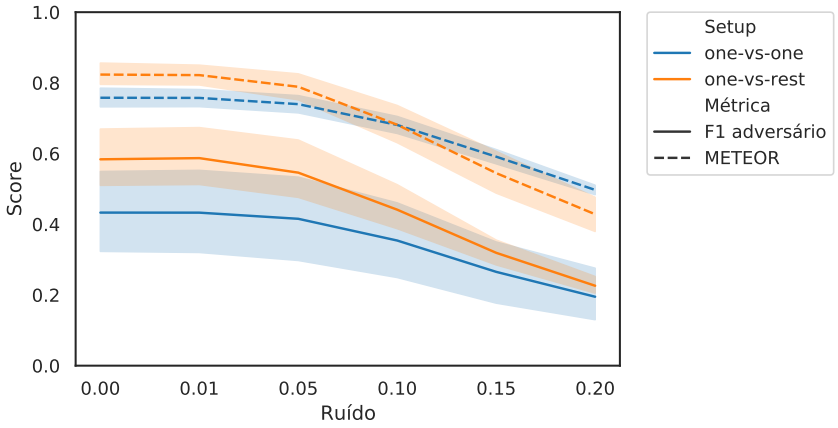

Figura 4. Comparação entre one-vs-one e one-vs-rest.

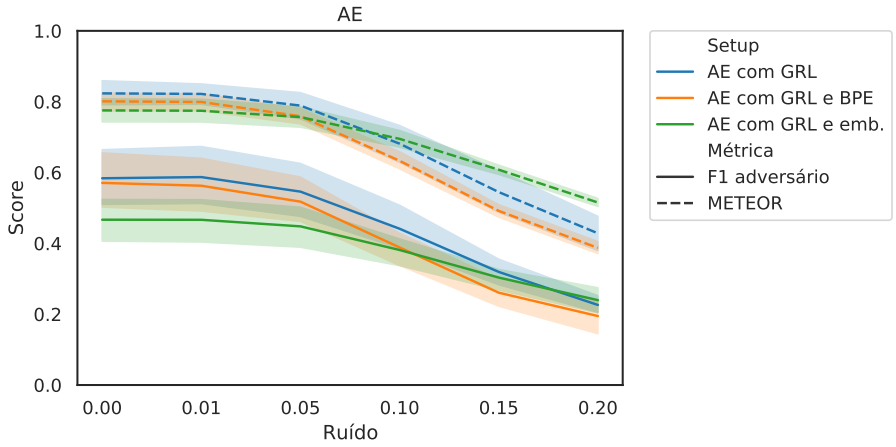

Figura 5. Comparação entre os modelos treinados com AE e GRL.

estratégias. O uso de *one-vs-rest* em auto-codificadores com GRL, portanto, só é viável com ruído. Por outro lado, o uso de *one-vs-rest* não prejudica o desempenho de modelos treinados com GAN a ponto de fazer com que o adversário tenha um desempenho maior que uma inferência aleatória.

A próxima comparação que fizemos foi entre as diferentes abordagens de ofuscação. Apresentamos os resultados destes experimentos nas Figuras 5 (auto-codificadores treinados com o método de ofuscação por invariância) e 6 (modelos treinados com o método de transferência de estilo utilizando GAN). Para cada método de ofuscação, são apresentados três modelos distintos: um para tokenização por palavras e sem *embeddings* pré-treinados, outro para tokenização por palavras com *embeeddings* pré-treinados e outro para tokenização por BPE. O eixo horizontal contém o desvio padrão aplicado durante a amostragem do vetor de ruído. O eixo vertical contém o *score* que pode representar tanto o F1-score do adversário (linha contínua) quanto o METEOR score (linha pontilhada).

O uso de vetores de palavras pré-treinados nos modelos treinados com auto-codificador e GRL gerou um METEOR score maior para os textos ofuscados com maior nível de ruído. Por exemplo, a média do METEOR score dos textos com ruído em 0,20 é de 0,42 para os modelos sem *embeddings* pré-treinados e 0,58 para os modelos com *embeddings* pré-treinados. O modelo treinado com GAN foi melhor em termos de ofuscação,

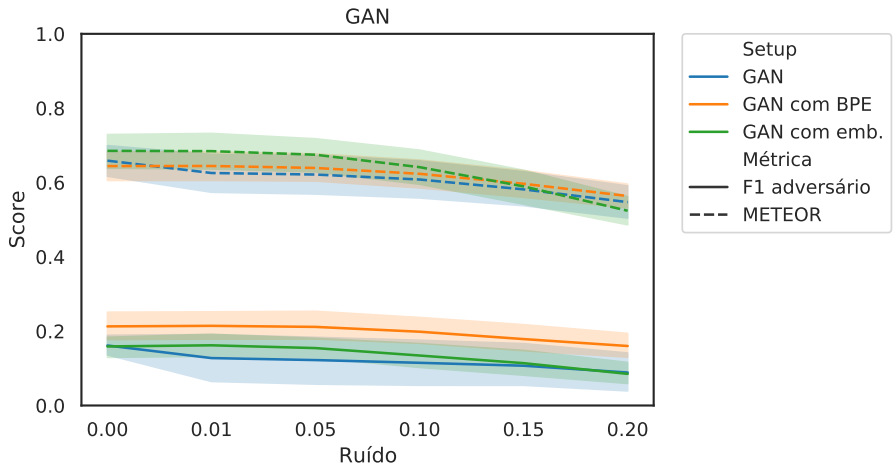

Figura 6. Comparação entre os modelos treinados com GAN.

mas em contrapartida, a similaridade semântica obteve um resultado inferior a $0,5$, o que indica uma alteração de significado entre as sentenças originais e as ofuscadas. O impacto dos vetores de palavras pré-treinados foi positivo na preservação semântica, mas os modelos treinados com esta abordagem foram menos eficientes ao enganar o adversário, visto que o F1-score ficou acima do nível de inferência. O uso de BPE não trouxe resultados positivos para os modelos treinados com auto-codificadores e GRL. No entanto, os modelos treinados com GAN apresentaram uma melhora na qualidade das sentenças geradas usando BPE.

6. Conclusão e trabalhos futuros

Nós avaliamos duas abordagens para ofuscação de atributos de autoria de textos. Para cada abordagem, nós definimos experimentos que combinaram diferentes técnicas de pré-processamento de texto, níveis de ruído e técnicas de transferência de aprendizado. Cada experimento foi avaliado sob uma métrica que mensura o ganho de ofuscação e outra que mensura a similaridade semântica entre o texto original e o texto ofuscado.

O experimento que obteve o melhor resultado em termos de ofuscação foi o que aplicamos um codificador-decodificador treinado com GAN, utilizando tokenização por BPE e com um ruído retirado de uma distribuição normal com desvio padrão de 0,15. Com esta configuração, o desempenho do classificador do adversário ficou próximo a uma inferência aleatória, e a similaridade semântica ficou 20% maior do que a abordagem que não utiliza BPE.

A estratégia de treino *one-vs-rest* impacta negativamente o desempenho dos modelos de ofuscação por invariância que utilizam auto-codificadores com GRL. No entanto, tal abordagem não gera uma diferença significativa nos modelos de ofuscação por transferência de estilos treinados com GAN.

Embora os resultados experimentais apontem o sucesso em enganar adversários que utilizam classificadores de texto, uma prova formal se faz necessária. Neste sentido, nós pretendemos futuramente aplicar métodos baseados em privacidade diferencial que sejam capazes de garantir um nível mínimo de privacidade. Outro ponto que endereçaremos futuramente é a transformação de documentos inteiros em vez de sentença por sentença.

Referências

Bagnall, D. (2015). Author identification using multi-headed recurrent neural networks. *arXiv preprint arXiv:1506.04891*.

Banerjee, S. and Lavie, A. (2005). METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *ACL workshop on intrinsic and extrinsic evaluation measures for machine translation*, pages 65–72.

Bevendorff, J., Wenzel, T., Potthast, M., Hagen, M., and Stein, B. (2020). On divergence-based author obfuscation: An attack on the state of the art in statistical authorship verification. *it-Information Technology*, 62(2):99–115.

Bird, S., Klein, E., and Loper, E. (2009). *Natural Language Processing with Python*. O'Reilly Media.

Bo, H., Ding, S. H., Fung, B., and Iqbal, F. (2019). ER-AE: Differentially-private Text Generation for Authorship Anonymization. *arXiv preprint arXiv:1907.08736*.

Emmery, C., Manjavacas Arevalo, E., and Chrupała, G. (2018). Style Obfuscation by Invariance. In *Proceedings of the 27th COLING*, pages 984–996, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Fernandes, N., Dras, M., and McIver, A. (2019). Generalised differential privacy for text document processing. In *International Conference on Principles of Security and Trust*, pages 123–148. Springer.

Fitzgerald, J. R. (2004). Using a forensic linguistic approach to track the unabomber. *Profilers. New York: Prometheus Books*, pages 193–221.

Ganin, Y. and Lempitsky, V. (2015). Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*, pages 1180–1189.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Graves, A. (2013). Generating Sequences With Recurrent Neural Networks. *arXiv preprint arXiv:1308.0850*.

Gröndahl, T. and Asokan, N. (2019). Effective writing style imitation via combinatorial paraphrasing. *arXiv preprint arXiv:1905.13464*.

Hartmann, N., Fonseca, E., Shulby, C., Treviso, M., Silva, J., and Aluísio, S. (2017). Portuguese word embeddings: Evaluating on word analogies and natural language tasks. In *Proceedings of the 11th Brazilian STIL*, pages 122–131.

Howard, J. and Ruder, S. (2018). Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.

Indurkhya, N. and Damerau, F. J. (2010). *Handbook of natural language processing*, volume 2. CRC Press.

Karadzhov, G., Mihaylova, T., Kiprov, Y., Georgiev, G., Koychev, I., and Nakov, P. (2017). The case for being average: A mediocrity approach to style masking and author obfuscation. In *CLEF for European Languages*, pages 173–185. Springer.

Keswani, Y., Trivedi, H., Mehta, P., and Majumder, P. (2016). Author Masking through Translation. In *CLEF (Working Notes)*, pages 890–894.

Mahmood, A., Ahmad, F., Shafiq, Z., Srinivasan, P., and Zaffar, F. (2019). A girl has no name: Automated authorship obfuscation using mutant-x. *Proceedings on Privacy Enhancing Technologies*, 2019(4):54–71.

Mansoorizadeh, M., Rahgooy, T., Aminiyan, M., and Eskandari, M. (2016). Author obfuscation using wordnet and language models—notebook for pan at clef 2016. In *CLEF 2016 Evaluation Labs and Workshop–Working Notes Papers*, pages 5–8.

McDonald, A. W., Afroz, S., Caliskan, A., Stolerman, A., and Greenstadt, R. (2012). Use fewer instances of the letter "i": Toward writing style anonymization. In *International Symposium on Privacy Enhancing Technologies Symposium*, pages 299–318. Springer.

Mihaylova, T., Karadjov, G., Kiprov, Y., Georgiev, G., Koychev, I., and Nakov, P. (2016). Su@ pan'2016: Author obfuscation. In *CLEF (Working Notes)*, pages 956–969.

Mosteller, F. and Wallace, D. L. (1963). Inference in an authorship problem: A comparative study of discrimination methods applied to the authorship of the disputed federalist papers. *Journal of the American Statistical Association*, 58(302):275–309.

Narayanan, A., Paskov, H., Gong, N. Z., Bethencourt, J., Stefanov, E., Shin, E. C. R., and Song, D. (2012). On the feasibility of internet-scale author identification. In *2012 IEEE Symposium on Security and Privacy*, pages 300–314. IEEE.

Pan, S. J. and Yang, Q. (2009). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

Potthast, M., Hagen, M., and Stein, B. (2016). Author Obfuscation: Attacking the State of the Art in Authorship Verification. In *CLEF (Working Notes)*, pages 716–749.

Ruder, S., Ghaffari, P., and Breslin, J. G. (2016). Character-level and multi-channel convolutional neural networks for large-scale authorship attribution. *arXiv preprint arXiv:1609.06686*.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

Sari, Y., Vlachos, A., and Stevenson, M. (2017). Continuous n-gram representations for authorship attribution. In *Proceedings of the 15th EACL: Volume 2, Short Papers*, pages 267–273.

Shetty, R., Schiele, B., and Fritz, M. (2018). A4NT: Author Attribute Anonymity by Adversarial Training of Neural Machine Translation. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 1633–1650, Baltimore, MD. USENIX Association.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Tacorda, A. J., Ignacio, M. J., Oco, N., and Roxas, R. E. (2017). Controlling byte pair encoding for neural machine translation. In *IALP'17*, pages 168–171. IEEE.

Varela, P., Justino, E., and Oliveira, L. S. (2011). Selecting syntactic attributes for authorship attribution. In *The 2011 International Joint Conference on Neural Networks*, pages 167–172. IEEE.