Lince: Uma Ferramenta para Ofuscação Automática de Textos em Português

Antônio M. R. Franco[1], **Leonardo B. Oliveira**[1]

[1]Universidade Federal de Minas Gerais

{franco,leob}@dcc.ufmg.br

Abstract. *Currently, there are several approaches to provide anonymity on the Internet. However, one can still identify anonymous users through their writing style. With the advances in neural network and natural language processing research, the success of a classifier when accurately identify the author of a text is growing. On the other hand, new approaches that use neural networks for automatic generation of obfuscated texts have also arisen to fight anonymity adversaries. In this work, we present Lince, a tool that implements two machine learning state-of-art text obfuscation approaches.*

Resumo. *Atualmente existem diversas abordagens que fornecem anonimato na Internet. No entanto, usuários anônimos ainda podem ser identificados pelo seu estilo de escrita. Com o avanço das pesquisas em redes neurais e processamento de linguagem natural, a chance de sucesso de um classificador identificar corretamente o autor de um texto tem crescido cada vez mais. Por outro lado, novas abordagens que utilizam redes neurais para geração automática de textos ofuscados também têm surgido para combater os adversários do anonimato na Internet. Neste trabalho, nós apresentamos Lince, uma ferramenta que implementa duas abordagens do estado da arte baseadas em aprendizado de máquina para ofuscação de textos.*

1. Introdução

O anonimato na Internet é uma condição que pode ser requerida em diversas situações. Um denunciante relatando um ato fraudulento através de um canal de denúncias pode querer submeter seu relato de forma anônima, assim como um cliente que submete uma avaliação negativa sobre um produto ou serviço pode requerer seu anonimato. Em casos mais graves, o anonimato pode ser requerido inclusive como forma de preservar a vida de um denunciante[1].

Um usuário pode aplicar diversas técnicas para ocultar seus atributos de identidade. Um exemplo é a rede Tor[2], que pode ser utilizada para ocultar os endereços de rede dos seus usuários. No entanto, mesmo utilizando estas técnicas uma pessoa ainda poderia ser identificada pelo seu estilo de escrita [Narayanan et al. 2012] utilizando Processamento de Linguagem Natural (PLN).

Para proteger o seu anonimato, um usuário também pode utilizar PLN para ofuscar os atributos estilísticos do seu texto e reduzir as chances de sucesso do adversário, como no exemplo da Figura 1. Duas técnicas que utilizam redes neurais para

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

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

geração automática de textos ofuscados foram propostas recentemente por Emmery et al. [Emmery et al. 2018] e Shetty et al. [Shetty et al. 2018]. O grande desafio destas abordagens – além de mascarar o estilo de escrita do autor – é preservar a semântica do texto gerado.

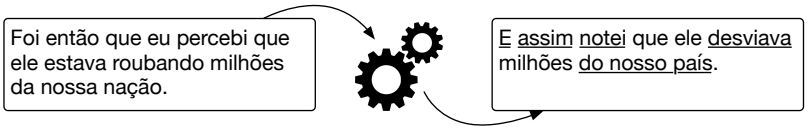


Figura 1. Exemplo de texto gerado por um ofuscador de estilo.

O nosso objetivo neste trabalho é propor uma ferramenta para ofuscar textos em português utilizando duas técnicas do estado da arte [Shetty et al. 2018, Emmery et al. 2018]. Além de aplicar as técnicas de ofuscação implementadas, nossa ferramenta também calcula a chance de um adversário descobrir o real autor do texto e a qualidade semântica do texto gerado.

Este artigo está organizado da seguinte forma. A Seção 2 discute os trabalhos relacionados. A Seção 3 apresenta os fundamentos necessários para compreender as abordagens de ofuscação de texto implementadas. A Seção 4 descreve a forma como implementamos o protótipo, como realizamos os treinamentos das redes neurais e quais foram os conjuntos de dados utilizados. A Seção 5 apresenta as métricas disponíveis no protótipo para avaliar a performance do adversário e a qualidade semântica do texto. Por fim, a Seção 6 explica como será a demonstração e a Seção 7 conclui o artigo.

2. Trabalhos Relacionados

O PAN[3] publica relatórios anualmente com os resultados das avaliações das ferramentas que são submetidas para a força tarefa de ofuscação de textos [Mihaylova et al. 2016, Stamatatos et al. 2018]. As ferramentas submetidas para o PAN até então são regras pré-definidas que são aplicadas para transformar os textos, e não houve nenhuma submissão de ferramentas baseadas em redes neurais. Além disso, o idioma Português não faz parte do escopo de avaliação do PAN.

[Potthast et al. 2016] propuseram um experimento para comparar a performance de diversos ofuscadores de texto. Assim como nos nossos experimentos, eles também utilizaram métricas para comparar a eficiência em esconder os atributos estilísticos e preservar a semântica dos textos gerados. Dentre os ofuscadores que foram avaliados, no entanto, não havia nenhum que fosse baseado em aprendizado profundo e que fosse treinado para ofuscar textos em português.

[Davis 2019] desenvolveu uma ferramenta chamada Nondescript[4] para ofuscar atributos de estilo de escrita que podem relevar o autor de um texto. Assim como as demais ferramentas já publicadas, Nondescript também é dependente do idioma e não funciona para textos escritos em português.

[3]PAN é uma série de eventos científicos e tarefas compartilhadas para resolver problemas de estilometria e forense em textos. Mais detalhes podem ser consultados em `https://pan.webis.de`.

[4]`https://github.com/robincamille/nondescript2`

3. Fundamentos

O problema de ofuscação de textos é similar a um problema de tradução de automática de linguagens, onde o objetivo é traduzir um texto de um idioma para o outro [Shetty et al. 2018]. A diferença está no alvo da tradução: em vez de traduzir um texto para outro idioma, o objetivo de um ofuscador é transformá-lo em outro texto escrito no mesmo idioma, porém com um estilo de escrita diferente. Muitos problemas de tradução automática atualmente são resolvidos com técnicas de aprendizagem profunda (*deep learning*) aplicadas à PLN, e nesta seção serão apresentados os conceitos fundamentais para compreender as abordagens avaliadas.

Redes Neurais Recorrentes. No campo de aprendizado de máquina, existem arquiteturas de redes neurais que são mais eficientes para resolver problemas de PLN. As redes neurais profundas tradicionais não memorizam informações relacionadas entre as entradas, e isso é um problema em tarefas de PLN [Goodfellow et al. 2016]. Esse problema pode ser ilustrado com o seguinte exemplo. Suponha que você quer traduzir a frase "Minha fruta favorita é manga."para inglês. Se a palavra manga for analisada fora de contexto, um tradutor não saberá se a tradução correta é *mango* ou *sleeve*. Redes neurais recorrentes resolvem esse problema ao memorizar as informações que são aprendidas anteriormente. Um exemplo de arquitetura de uma rede neural recorrente é exibido na Figura 2.

Identificação de Autoria. Um adversário pode utilizar uma rede neural recorrente para analisar um conjunto de textos e extrair características intrínsecas dos estilos de escrita dos seus autores. Estas características, por sua vez, podem ser utilizadas para treinar um modelo de aprendizagem profunda com o objetivo de classificar um texto de entrada como sendo ou não escrito por um autor específico. Dependendo do volume de textos no conjunto de dados utilizados para teste, um modelo de aprendizado de máquina pode obter performance superior à análise manual de um perito experiente em estilometria [Varela et al. 2011].

Ofuscação de Textos. O problema de ofuscação de textos, portanto, consiste em modificar um texto de entrada It de modo que o texto modificado Ot não possua os atributos estilísticos que podem ser utilizados para identificar o seu autor. Este problema pode ser visto como a tradução de um texto de um estilo para o outro, porém dentro da mesma linguagem. O principal desafio em tarefas de ofuscação texto é a preservação da semântica do texto original [Emmery et al. 2018]. Neste trabalho, nós implementamos um protótipo com duas abordagens de ofuscação de textos baseadas em redes neurais recorrentes: ofuscação por invariância [Emmery et al. 2018] e ofuscação por transferência de estilo [Shetty et al. 2018].

4. Desenvolvimento do Protótipo

Esta seção apresenta a arquitetura do protótipo desenvolvido, as arquiteturas das redes neurais utilizadas para ofuscar os textos e os conjuntos de dados que utilizamos para treinar e testar modelos. O código fonte do protótipo está disponível em um repositório do GitLab[5].

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

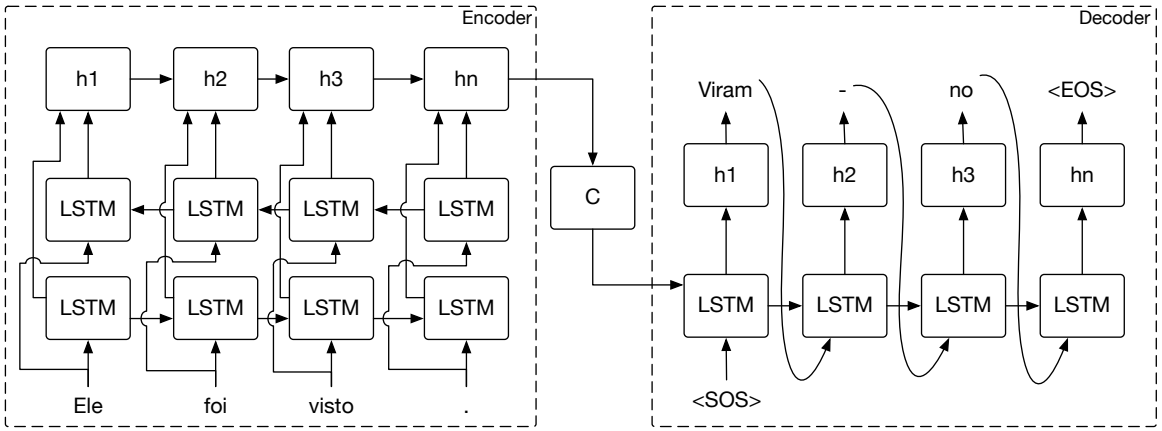


Figura 2. Arquitetura de uma rede neural recorrente com um *encoder* e um *decoder* [Cho et al. 2014].

4.1. Interface Web

O módulo web de Lince possui dois componentes principais: um *backend* e um *frontend*. O backend foi desenvolvido utilizando a linguagem de programação Python 3 e o *micro-framework* Flask. O *frontend* foi implementado com os *frameworks* Vue.js e Vuetify.js. Uma representação visual dos componentes da aplicação é exibida na Figura 3.

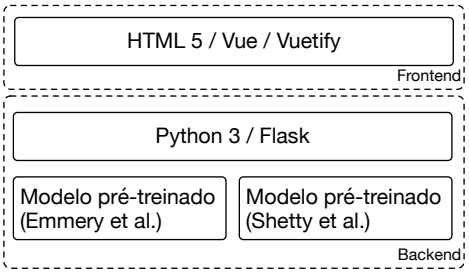


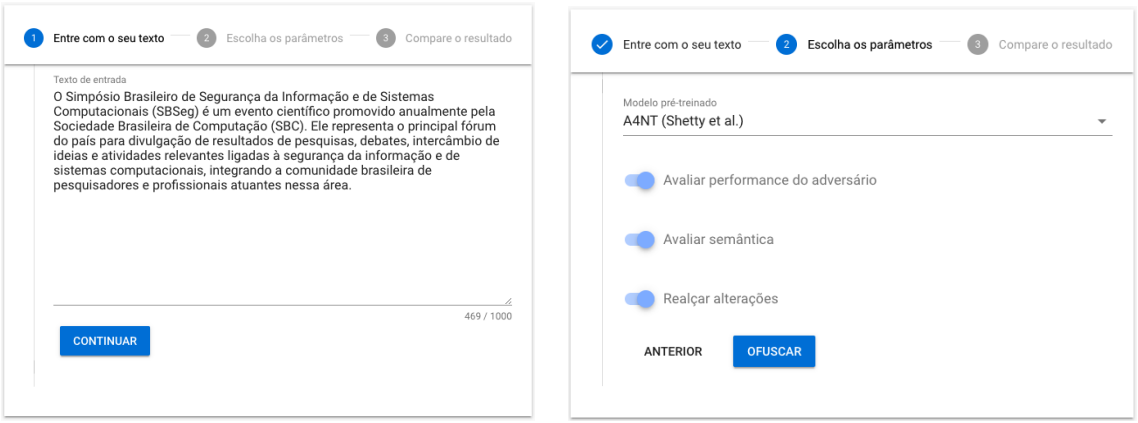
Figura 3. Arquitetura da aplicação web.

O principal componente da interface web que implementamos é um formulário de múltiplos passos. No primeiro passo deste formulário, o usuário é convidado a inserir o texto de entrada que será ofuscado. No segundo passo, é possível customizar parâmetros como o modelo pré-treinado que será utilizado e as ferramentas de avaliação que serão executadas. Por fim, no terceiro passo o usuário consegue visualizar o texto ofuscado e as métricas geradas pelas ferramentas de avaliação escolhidas. A Figura 4 mostra um exemplo do formulário onde o texto é inserido.

4.2. Redes Neurais

Uma vez que o usuário insere o texto na interface web e seleciona os parâmetros da ofuscação, a nossa aplicação carrega o modelo de linguagem selecionado e executa o passo de ofuscação. Inicialmente, nós implementamos dois modelos do estado da arte, que serão explicados a seguir.

Uma das abordagens que implementamos foi proposta em [Shetty et al. 2018]. A arquitetura desta abordagem é composta por um gerador de textos, que funciona de maneira análoga a um tradutor; e um classificador de textos, que é utilizado para identificar



(a) **(b)**

Figura 4. Telas do (a) texto submetido e dos (b) parâmetros do ofuscador.

as classes dos textos gerados pelo gerador. O gerador é uma rede neural recorrente com um *encoder* que recebe a sentença de entrada e gera um vetor de contexto; e um *decoder* que recebe o vetor de contexto e gera a sentença de saída. O classificador é uma rede recorrente convencional treinada para identificar atributos estilísticos dos textos. O gerador e o classificador são conectados um ao outro em uma arquitetura de *Generative Adversarial Network* (GAN) [Goodfellow et al. 2014]. A implementação deste modelo foi baseada no código disponibilizado pelos autores[6].

A outra abordagem que implementamos foi proposta em [Emmery et al. 2018]. A arquitetura da rede é composta por um *encoder*, que é responsável por processar o texto de entrada com células *Long short-term memory* (LSTM) [Hochreiter and Schmidhuber 1997] e gerar um vetor de contexto que armazena as propriedades das sentenças; e por um *decoder*, que recebe como entrada o vetor de contexto e gera a sentença alvo no estilo de escrita desejado. Além disso, esta rede neural possui uma camada de *Gradient Reversal Layer* (GRL) [Ganin and Lempitsky 2014] que é utilizada para aprender os atributos invariantes que não devem ser modificados ao gerar o texto ofuscado. Nossa implementação foi baseada no código disponibilizado pelos autores no GitHub[7].

As duas abordagens utilizam uma arquitetura de *encoder-decoder* para ofuscar os textos. A diferença entre as duas está em como as redes neurais são treinadas. A abordagem de Shetty et al. utiliza GAN para treinar os modelos, enquanto a abordagem de Emmery et al. utiliza um *autoencoder* com GRL e *decoder* condicional.

4.3. Conjuntos de dados

Seguindo a abordagem proposta em [Emmery et al. 2018], nós utilizamos um conjunto de dados extraído de diferentes traduções em português da bíblia para treinar os modelos. A bíblia foi escolhida porque os versículos de duas traduções diferentes podem ser combinados para formarem pares de sentenças que possuem o mesmo sentido mas que foram escritas em estilos diferentes. As traduções foram extraídas do portal

[6] https://github.com/rakshithShetty/A4NT-author-masking
[7] https://github.com/cmry/style-obfuscation

bibliaonline.com.br utilizando um *script* automático para *scrapping* de conteúdo web. As versões utilizadas foram a Nova Versão Internacional, Almeida Corrigida Fiel e Almeida Revista e Atualizada. No total, o nosso conjunto de dados ficou composto por 31102 versículos de cada versão. Nós permutamos os versículos extraídos para construir pares entre traduções distintas e, no total, obtivemos 186612 pares.

Nós também treinamos os modelos com um outro conjunto de dados formado por uma base disponibilizada em [Varela et al. 2011], que é composta por 3000 textos em português escritos por 100 colunistas de portais de notícias diversos. Esta base foi escolhida porque os textos são segmentados por autores, e esse é um requerimento para treinar o classificador de textos, pois são necessárias anotações para distinguir entre os diferentes estilos de escrita que podem ser identificados.

4.4. Treinamento dos Modelos

Para treinar o modelo de Shetty et al., nós executamos os seguintes passos. (i) Treinamos o classificador de estilos de cada ofuscador utilizando os conjunto de dados da Subseção 4.3. (ii) Treinamos os *autoencoders* de cada estilo do conjunto de dados para gerar os textos iniciais. (iii) Treinamos o gerador de textos final utilizando o *framework* GAN com o classificador e com os *autoencoders* treinados previamente. O processo de treinamento do gerador de textos é ilustrado na Figura 5.

Para treinar o modelo de Emmery et al., nós executamos os *scripts* de treinamento disponibilizados pelos autores no repositório do GitHub[8] utilizando os mesmos conjntos de dados da Subseção 4.3.

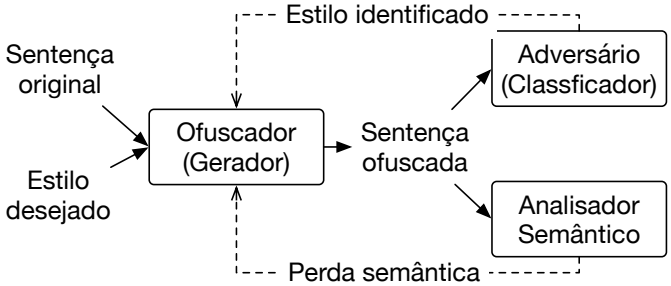


Figura 5. Treinamento dos modelos para ofuscação de textos.

5. Métricas para Avaliação

Esta seção descreve as métricas disponíveis no nosso protótipo que podem ser utilizadas para comparar o texto de entrada com o texto ofuscado.

Ofuscação. Para mensurar a performance do ofuscador ao esconder os atributos estilísticos do texto, o nosso protótipo inclui o *caravel* [Bagnall 2015], que foi a ferramenta que obteve a melhor nota na tarefa de identificação de autoria do PAN. Esta é uma ferramenta adversária, ou seja, tenta revelar se um texto foi escrito por um autor ou não. Para isso, ela precisa ser previamente treinada em um conjunto de textos de diferentes autores para que, ao receber um novo texto, ela retorne a probabilidade de cada autor ter escrito

[8] https://github.com/cmry/style-obfuscation/blob/master/src/train_sae.py

o texto em questão. Para computar a nota final do ofuscador de textos, *caravel* utiliza o F1-score computado sobre o conjunto de testes.

Semântica. Para mensurar a consistência semântica entre as sentenças originais e as sentenças geradas pelos ofuscadores, o nosso protótipo disponibiliza as métricas METEOR [Denkowski and Lavie 2014] e BLEU [Papineni et al. 2002]. Nós escolhemos estas métricas porque elas são utilizadas para medir a qualidade das sentenças geradas por trabalhos de tradução automática de linguagens. Nós utilizamos o *nlg-eval*[9] para implementar estas métricas no protótipo.

6. Demonstração

O nosso protótipo está hospedado em um servidor em nuvem e está disponível para ser acessado através da Internet. Para demonstrá-lo no Salão de Ferramentas, nós precisaremos de acesso à Internet para um laptop que levaremos e de um ponto de energia elétrica.

A demonstração que apresentaremos segue os seguintes passos:

- Escrever (ou colar) o texto que será ofuscado em um formulário da nossa aplicação web.
- Selecionar o modelo pré-treinado que será utilizado para transformar o texto e o estilo de escrita de destino.
- Selecionar as opções de avaliação (performance do adversário, pontuação da semântica e diferença entre a entrada e a saída).
- Receber o texto ofuscado pelos modelos implementados no nosso protótipo.

7. Conclusão e Trabalhos Futuros

Nós implementamos um protótipo que oferece duas abordagens baseadas em aprendizado de máquina para ofuscação automática de textos. Além de ofuscar o estilo de escrita de um texto, o nosso protótipo também calcula a chance de sucesso do adversário e a qualidade semântica do texto gerado pelo ofuscador.

Como trabalhos futuros, nós investigaremos o impacto de *word embeddings* pré-treinados na semântica dos textos gerados. Nós também investigaremos se redes neurais pré-treinadas (como BERT e XLNet) podem ser aplicadas para melhorar a qualidade semântica dos textos ofuscados. Por fim, nós investigaremos se criptossistemas como SMC (*Secure Multi-party Computation*) e criptografia homomórfica podem ser empregados para executar os algoritmos de ofuscação de textos em um ambiente de computação em nuvem sem enviar o texto não-ofuscado para os servidores.

Referências

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

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

[9] `https://github.com/Maluuba/nlg-eval`

Davis, R. C. (2019). Obfuscating authorship: Results of a user study on nondescript, a digital privacy tool. *CUNY Academic Works*.

Denkowski, M. and Lavie, A. (2014). Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the ninth workshop on statistical machine translation*, pages 376–380.

Emmery, C., Manjavacas, E., and Chrupała, G. (2018). Style Obfuscation by Invariance. In *COLING 2018*, pages 984–996.

Ganin, Y. and Lempitsky, V. (2014). Unsupervised domain adaptation by backpropagation. *arXiv preprint arXiv:1409.7495*.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

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.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

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.

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.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318.

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.

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.

Stamatatos, E., Rangel-Pardo, F. M., Tschuggnall, M., Stein, B., Kestemont, M., Rosso, P., and Potthast, M. (2018). Overview of PAN 2018. Author identification, author profiling, and author obfuscation. *Lecture Notes in Computer Science*, 11018:267–285.

Varela, P., Justino, E., and Oliveira, L. S. (2011). Selecting syntactic attributes for authorship attribution. In *IJCNN*, pages 167–172. IEEE.