USO DE DEEP LEARNING NA DETECÇÃO DE FAKE NEWS

Manoel Villas Bôas Júnior

Mestre em Computação Aplicada, ITLab/POLI da Universidade Federal do Rio de Janeiro (UFRJ), Rio de Janeiro, RJ, Brasil

mvbjunior@poli.ufrj.br

Sergio Luiz Machado

Master Business em Computação Aplicada, ITLab/POLI da Universidade Federal do Rio de Janeiro (UFRJ), Rio de Janeiro, RJ, Brasil

sergiolmachado@hotmail.com

RESUMO

Este documento apresenta uma aplicação para a detecção de notícias falsas através do uso da técnica de aprendizagem profunda. Foram utilizadas duas bases-de-dados: uma compilação de notícias falsas perpetradas na Internet, e outra de notícias genuínas, publicadas pelo jornal brasileiro Folha de São Paulo. O modelo proposto tem como objetivo aplicar diferentes métodos de extração de dados para a tarefa de classificação de textos supervisionados. Para que os dados relevantes ao experimento fossem adequadamente processados pelo algoritmo de aprendizagem de máquina, foram usados os modelos de extração saco de palavras, caso especial da técnica n-grama, onde n-grama=1, além do modelo bigrama, ambos combinados com o modelo de processamento de linguagem natural TF-IDF (Frequência do termo - Frequência documental inversa). O resultado deste trabalho permite identificar qual das técnicas de extração ofereceu os melhores resultados na detecção de notícias falsas.

Palavras-chave: Aprendizagem Profunda. Notícias falsas. Técnicas de extração de dados. TF-IDF.

THE USE OF DEEP LEARNING IN FAKE NEWS DETECTION

ABSTRACT

This paper presents an application for the detection of fake news using the deep learning technique. Two databases were used: a compilation of fake news perpetrated on the Internet, and another one of genuine news, published by the Brazilian newspaper Folha de São Paulo. The proposed model aims to apply different methods of data extraction for the task of classification of supervised texts. In order to make the data relevant to the experiment adequately processed by the machine learning algorithm, we used the bag-of-words extraction models, a special case of n-gram technique, where n-gram = 1, as well as the bigram model, both combined with the natural language processing model TF-IDF (Frequency of the term - Inverse document frequency). The result of this work allows to identify which of the extraction techniques offered the best results in the detection of fake news.

Keywords: Deep Learning. Fake News. Data extraction techniques. TF-IDF.

1 INTRODUÇÃO

Este trabalho trata sobre a importância da disseminação de notícias falsas na internet. Sob essa ótica o problema a ser resolvido é o de conhecer melhor uma das técnicas computacionais mais utilizadas atualmente para a identificação de notícias falsas na Internet: o aprendizado profundo de máquina, ou *deep learning*.

2 DESENVOLVIMENTO

Neste capítulo é proposto um modelo de detecção de notícias falsas (Sistema Protótipo de Identificação de *Fake News*) baseado na técnica de redes neurais artificiais. O modelo foi implementado utilizando a linguagem *Python*, a escolha se justifica por esta linguagem possuir ferramentas especializadas no processamento de linguagem natural, como o NLTK (NLTK, 2018) e o Gensim (ŘEHŮŘEK, 2018). Além disso, a linguagem *Python* dispõe da biblioteca scikit-learn (SCIKIT-LEARN, 2018), que contém a

implementação dos algoritmos de otimização de hiperparâmetros, bem como das principais técnicas de aprendizado de máquina, dentre elas, Redes Neurais Artificiais. Esta implementação se deu em um sistema protótipo que identifica se a notícia contida no texto é falsa ou verdadeira. A arquitetura deste modelo é baseada em quatro fases, descritas a seguir:

- Fase 1 – Preparação dos Dados
- Fase 2 – Indexação
- Fase 3 – Mineração
- Fase 4 – Pós-Processamento

Figura 1 - Diagrama do processo de mineração de textos.

Fonte: CARRILHO JÚNIOR, J.R., **Desenvolvimento de uma metodologia para mineração de textos**

2.1 Preparação dos dados

Nesta seção, ocorrem as etapas de leitura das bases-de-dados que se deseja converter, além dos processos de limpeza e normalização destes dados. Ao final deste processo, *datasets* estão prontos para serem consumidos pelos algoritmos de otimização e

de aprendizado de máquina.

A primeira base contém 1448 artigos classificados como boato (CHAVES, 2018), onde cada registro contém texto e metadados. A segunda base é composta de 167083 artigos genuínos, extraídos do jornal Folha de São Paulo (SANTANA, 2018). Deste universo de artigos genuínos, pouco menos de 5800 foram selecionados em uma amostra uniforme. Os artigos a serem processados pelos algoritmos passaram previamente por um processo de padronização inicial, que consiste na substituição de letras maiúsculas por minúsculas, bem como a remoção de caracteres especiais e outras sequências de caracteres como "\n", "\t", "\r", dentre outros. Os vetores contendo notícias falsas e genuínas receberam um valor (1 para notícias falsas e 0 para notícias genuínas), foram unificados e misturados aleatoriamente. Em seguida, foram divididos em três novos conjuntos de dados: treinamento, teste e validação, na seguinte proporção:

- 1448/20%: Definição de parâmetros e otimização de hiperparâmetros.
- 4344/60%: Treinamento do classificador.
- 1448/20%: Avaliação dos resultados.

Cada conjunto foi armazenado em um arquivo, onde cada registro contém duas colunas: a primeira é a notícia; a segunda informa o valor 0 ou 1, indicando se a notícia é verdadeira ou falsa, respectivamente. A Figura 1 ilustra um exemplo com 6 registros, contendo o artigo e o respectivo valor, indicando se a notícia é falsa ou genuína:

Figura 1 - Registros contendo artigos e respectivos valores.

[["Notícia/boato A",0], ["Notícia/boato B",1], ["Notícia/boato C",1], ["Notícia/boato E",0], ["Notícia/boato F",1], ["Notícia/boato K",1]]

Os artigos estão em formato textual e possuem tamanho variável, porém, sequência de palavras ou símbolos não pode ser processada diretamente pelos algoritmos de aprendizado de máquina. Portanto, o próximo passo na preparação do texto consiste na representação destes textos como vetores de tamanho fixo. Neste ponto, foi feita a tokenização dos textos.

Entrada: "Arquivo informando notícia de prisão do Governador é um vírus. Informe a todos os seus contatos para não abrir o arquivo chamado decretada a prisão do

Governador."

Através da biblioteca *natural language toolkit* (NLTK, 2018), é feita a tokenização e o resultado pode ser visto abaixo:

Saída: ['arquivo', 'informando', 'notícia', 'de', 'prisão', 'do', 'governador', 'é', 'um', 'vírus', 'informe', 'a', 'todos', 'os', 'seus', 'contatos', 'para', 'nao', 'abrir', 'o', 'arquivo', 'chamado', 'decretada', 'a', 'prisao', 'do', 'governador']

Neste ponto, a saída foi tokenizada e todos os tokens estão em minúsculas. Utilizando a saída acima como entrada para a remoção das *stopwords* e a aplicação do s*temming*, temos como saída:

Saída: ['arquivo', 'informando', 'notícia', 'prisão', 'governador', 'é', 'vírus', 'informe', 'todos', 'contatos', 'nao', 'abrir', 'arquivo', 'chamado', 'decretada', 'prisao', 'governador']

2.2 Indexação

Como resultado da etapa de limpeza, palavras sem relevância foram excluídas do vocabulário. Em seguida, usamos o modelo "Saco de palavras" (CARRILHO JÚNIOR, 2018) para fazer a contagem de palavras o resultado do processamento é ilustrado na Figura 2.

Figura 2: Contagem de palavras.

arquivo	2
informando	1
noticia	1
prisao	1
governador	2
e	1
virus	1
informe	1
todos	1
contatos	1

não	1
abrir	1
chamado	1
decretada	1

"Arquivo informando notícia de prisão do Governador é um vírus. Informe a todos os seus contatos para não abrir o arquivo chamado decretada a prisão do Governador."

Ao final desta etapa, serão gerados vetores de tamanho *n*, onde n é o tamanho de cada texto, e cada entrada do vetor corresponde à quantidade de vezes que um termo na posição *i* do vocabulário apareceu no texto: [2,1,1,1,2,1,1,1,1,1,1,1,1,1]. A partir do modelo "Saco de palavras", é possível efetuar algumas variações que podem gerar melhores resultados, por isso, além de dividir uma sentença ou frase em palavras individuais, também foi adotada a divisão de sentenças em duas palavras (bigramas). Outro modelo de processamento de linguagem natural adotado por este experimento para a determinação das palavras mais importantes em cada documento do *corpus* é o modelo TF-IDF (DROIDHEAD, 2018). Este modelo já está implementado na biblioteca Gensim (ŘEHŮŘEK, 2018), e comprovados na prática por uma comunidade de pesquisadores. Assim, o experimento foi baseado nesta estrutura.

2.3 Mineração (Otimização e Execução)

O algoritmo de otimização de hiperparâmetros escolhido para o experimento foi a busca aleatória (CLASSE RANDOMIZEDSEARCHCV, 2018), por ser um dos algoritmos de otimização com melhor desempenho. Esta abordagem utiliza um intervalo de valores para cada um dos parâmetros do algoritmo e pesquisa esse espaço de valores para o melhor resultado possível. Portanto, uma medida de desempenho deve ser selecionada, para isso, a pontuação-F1 (*F1-score*) foi a escolhida.

A pontuação-F1 é amplamente utilizada em pesquisas de processamento de lin-

guagem natural, pode ser usada para avaliar classificadores binários e, por meio dos valores de precisão e revocação, oferece informações adicionais sobre o desempenho. O desempenho de cada conjunto de parâmetros é avaliado usando uma validação cruzada de cinco vezes e conduzido para o algoritmo com todos os quatro conjuntos de dados gerados a partir do conjunto de validação.

O conjunto de testes foi vetorizado usando os mesmos vetores que foram usados para a vetorização do conjunto de treinamento para que ambos os conjuntos tenham o mesmo comprimento de vetor de entrada no experimento final, além disso, a saída da otimização de hiperparâmetros deve ser considerada neste momento, já que os parâmetros do modelo que alcançaram a classificação mais alta (melhor pontuação-F1) nessa etapa são colocados em uma configuração diretamente no código do experimento.

O experimento de avaliação foi executado para cada combinação de algoritmo de aprendizado de máquina e configuração, os dados de treinamento e teste foram carregados, o modelo foi equipado com os dados de treinamento e avaliados com os dados do teste. Como na hiperparametrização, a medida do desempenho do modelo é a pontuação-F1 (*F1-score*). Além disso, os resultados de revocação e precisão são calculados para obter mais *insights*.

Para efeitos de comparação, o experimento foi executado quatro vezes, mantendo-se, geralmente, os parâmetros (CLASSE SKLEARN.NEURAL_NETWORK.MLPCLASSIFIER, 2018) em seu valor padrão. O número de camadas ocultas para o experimento é 100 (valor padrão).

Tabela 1 - Resultado do experimento para unigramas e unigramas com TF-IDF.

N-grama = 1 (B*ag-of-Words*)				
Parâmetros	Execução 1	Execução 2	Execução 3	Execução 4

Função de Ativação: ReLU Otimizador: Adam TF-IDF: **Desativado**	F1-score: 0.998 VN: 1143 FP: 1 FN: 0 VP: 304 ACC: 0.999 Precisão: 0.996 Recall: 1.0	F1-score: 0.996 VN: 1142 FP: 2 FN: 0 VP: 304 ACC: 0.998 Precisão: 0.993 Recall: 1.0	F1-score: 0.996 VN: 1142 FP: 2 FN: 0 VP: 304 ACC: 0.998 Precisão: 0.993 Recall: 1.0	F1-score: 0.996 VN: 1142 FP: 2 FN: 0 VP: 304 ACC: 0.998 Precisão: 0.993 Recall: 1.0
Função de Ativação: ReLU Otimizador: Adam TF-IDF: **Ativado**	F1-score: 0.978 VN: 1144 FP: 0 FN: 13 VP: 291 ACC: 0.991 Precisão: 1.0 Recall: 0.957	F1-score: 0.978 VN: 1144 FP: 0 FN: 13 VP: 291 ACC: 0.991 Precisão: 1.0 Recall: 0.957	F1-score: 0.965 VN: 1144 FP: 0 FN: 20 VP: 284 ACC: 0.986 Precisão: 1.0 Recall: 0.934	F1-score: 0.946 VN: 1144 FP: 0 FN: 31 VP: 273 ACC: 0.978 Precisão: 1.0 Recall: 0.898

Tabela 2 - Resultado do experimento para bigramas e bigramas com TF-IDF.

N-grama = 2				
Parâmetros	Execução 1	Execução 2	Execução 3	Execução 4

Função de Ativação: Sigmóide Otimizador: Adam TF-IDF: **Desativado**	F1-score: 0.988 VN: 1140 FP: 4 FN: 3 VP: 301 ACC: 0.995 Precisão: 0.986 Recall: 0.990	F1-score: 0.988 VN: 1140 FP: 4 FN: 3 VP: 301 ACC: 0.995 Precisão: 0.986 Recall: 0.990	F1-score: 0.985 VN: 1138 FP: 6 FN: 3 VP: 301 ACC: 0.993 Precisão: 0.980 Recall: 0.990	F1-score: 0.986 VN: 1139 FP: 5 FN: 3 VP: 301 ACC: 0.994 Precisão: 0.983 Recall: 0.990
Função de Ativação: Tangente Hiperbólica Otimizador: LBFGS TF-IDF: **Ativado**	F1-score: 0.849 VN: 1142 FP: 2 FN: 78 VP: 226 ACC: 0.954 Precisão: 0.991 Recall: 0.743	F1-score: 0.847 VN: 1142 FP: 2 FN: 79 VP: 225 ACC: 0.944 Precisão: 0.991 Recall: 0.740	F1-score: 0.843 VN: 1141 FP: 3 FN: 80 VP: 224 ACC: 0.942 Precisão: 0.986 Recall: 0.736	F1-score: 0.843 VN: 1141 FP: 3 FN: 80 VP: 224 ACC: 0.942 Precisão: 0.986 Recall: 0.736

De acordo com os dados constantes das tabelas 1 e 2, o melhor desempenho possível foi alcançado na configuração de unigramas sem a medida TF-IDF, caso em que a revocação alcançou 100%, ou seja, de 100 artigos de notícias falsas, todos foram detectados. Para a combinação de bigramas com o modelo de redes neurais concluiu-se que o desempenho também foi elevado e com valores próximos em relação aos unigramas, mas para o bigrama inversamente transformado, a precisão cai consideravelmente.

Oportunamente, destacamos outra medida de desempenho, a acurácia, que indica a porcentagem de documentos corretamente classificados, cuja fórmula é apresentada a seguir:

Carrilho Júnior (CARRILHO JÚNIOR, 2018), adota em seu trabalho a acurácia como medida de desempenho para a análise da categorização de textos utilizando outros dois algoritmos de categorização textual: o *Naive Bayes* e o SVM – *Support Vector Machine*. Naquele trabalho, os resultados obtidos com o SVM e o *Naive Bayes* foram considerados bastante aproximados, tendo como acurácia média 0,8891 e 0,8912, respectivamente. Ao utilizar o algoritmo de redes neurais, obtemos uma acurácia mais elevada, com valores entre 0,986 e 0,995.

3 CONSIDERAÇÕES FINAIS

A detecção de notícias falsas com base apenas no conteúdo dos artigos foi comprovada na implementação do projeto e no experimento que o acompanha, onde foi mostrado que a combinação de unigramas e o modelo de redes neurais obteve o melhor desempenho e pode detectar quase que a totalidade de artigos de notícias falsas, sendo considerada uma abordagem adequada para tarefas de classificação de textos.

Em suma, as redes neurais foram capazes de produzir excelentes resultados em termos de pontuação e revocação com unigramas e bigramas, porém adicionar a frequência inversa do documento tanto para unigramas quanto para bigramas melhorou o tempo de computação, mas gerou perda significativa no desempenho dos modelos.

REFERÊNCIAS

CARRILHO JÚNIOR, J.R., **Desenvolvimento de uma metodologia para mineração de textos**. Pontifícia Universidade Católica do Rio de Janeiro, 2007. Acesso em: 26 set. 2019.

CHAVES, R. **Boatos do WhatsApp e outros do Boatos.Org - 1900 boatos desmentidos por boatos.org** Disponível em: https://www.kaggle.com/rogeriochaves/boatos-de-whatsapp-boatosorg. Acesso em: 26 set. 2019.

CLASSE RANDOMIZEDSEARCHCV. Disponível em:

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html

Acesso em: 26 set. 2019.

CLASSE SKLEARN.NEURAL_NETWORK.MLPCLASSIFIER. Disponível em:

https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

Acesso em: 26 set. 2019.

DROIDHEAD, **TF-IDF (Term Frequency-Inverse Data Frequency)**. Disponível em:

https://medium.freecodecamp.org/how-to-process-textual-data-using-tf-idf-in-python-cd2bbc0a94a3

Acesso em: 26 set. 2019.

NLTK (Natural Language Toolkit). Disponível em: https://www.nltk.org/

Acesso em: 26 set. 2019.

ŘEHŮŘEK, R. **Topic Modelling for Humans** Disponível em: https://radimrehurek.com/gensim/index.html. Acesso em: 26 set. 2019.

SANTANA, M., **News of the Brazilian Newspaper - 167.053 news of the site Folha de São Paulo (Brazilian Newspaper)**. Disponível em: https://www.kaggle.com/marlesson/news-of-the-site-folhauol. Acesso em: 26 set. 2019.

SCIKIT-LEARN, **Machine Learning in Python**

Disponível em: https://scikit-learn.org/stable. Acesso em: 26 set. 2019.