Detectando Avaliações Falsas na Internet Usando Classificação de Texto

Daniel Atkinson Oliveira[114054054]

Universidade Federal do Rio de Janeiro, Rio de Janeiro RJ, Brasil
danatkhoo@gmail.com

Resumo Nos últimos anos o problema de notıcias falsas tem se tornado cada vez mais aparente. A abundância desse tipo de notıcia na internet apresenta um grande risco para a maioria dos usuários, especialmente para aqueles que só lêem manchetes. Um outro problema muito relacionado ao de notıcias falsas é o de avaliações falsas. Ambos os tipos de publicações têm como objetivo apresentar informações ou opiniões falaciosas e são dificeis de ser detectados por humanos. No caso das avaliações falsas tanto consumidores quanto prestadores de serviços são afetados. Ambas essas questões, inicialmente muito discutidas no ambiente popular e midiático, estão ganhando cada vez mais atenção no ambiente acadêmico pois esse tipo conteúdo continua a crescer, tornando-o um problema maior, porém ao mesmo tempo, aumentando as possibilidades de estudos. Neste artigo é feita a implementação de um modelo n-grama para detectar automaticamente avaliações falsas. O modelo implemen tado é baseado no modelo criado por Ahmed et al. [1] e pode ser usado também para detecção automática de notıcias falsas. No estudo original duas técnicas de extração de atributos e seis técnicas de classificação por aprendizado de máquina são comparados. Também são testados difer entes valores para os n-gramas e diferentes faixas de atributos. Neste artigo somente será implementado e analisado o caso que resultou na maior taxa de acerto no artigo original.

Keywords: avaliações falsas online · notıcias falsas · classificação de texto · segurança de redes sociais online.

1 Introdução

O experimento que será descrito foi realizado de acordo com a definição do segundo trabalho [4] da disciplina "Recuperação da Informação" [3], da Universidade Federal do Rio de Janeiro. O documento de definição do trabalho [4] diz que o aluno deve "Escolher e estudar um artigo atual da área de Recuperação da Informação. Desenvolver também algum tipo de implementação e experimento relacionado ao artigo selecionado. Escrever um relatório em formato de artigo cientıfico relatando o trabalho desenvolvido e apresentá-lo aos colegas." O artigo de Ahmed et al. [1] foi escolhido, pois, em sua implementação, diversos conceitos que foram abordados em sala de aula são postos em prática, além de ter apresentado bons resultados para a resolução de um problema muito relevante e atual.

O artigo de Ahmed et al. [1] começa com a apresentação do problema de avaliações falsas online, assim como uma explicação para o surgimento desse fenômeno. Em seguida, é feita uma categorização útil dos tipos de avaliações falsas online: aquelas que têm como propósito apresentar informação falsa sobre o produto ou serviço, seja essa informação positiva ou negativa (tipo 1), aquelas que são puramente direcionadas à marca ou empresa que oferece o serviço ou produto sem apresentar experiência com o produto (tipo 2) e aquelas que não são avaliações ou são textos de marketing, apresentando nenhuma relação ou relação indireta com o produto (tipo 3). É dito no artigo que a categoria mais difícil de identificar é a primeira, e é o foco do artigo, no que diz respeito à detecção de avaliações falsas.

Ahmed et al. [1] apresentam uma categorização de tipos de notícias falsas, argumentando que, pelo fato de, tanto notícias falsas quanto avaliações falsas compartilharem um mesmo atributo, que é o conteúdo falso, seria possível criar um modelo que classificaria tanto notícias falsas quanto avaliações falsas, que é a proposta deles. O modelo proposto por eles é uma combinação de atributos n-gramas de palavras, métricas de frequências de termos e classificação por aprendizado de máquina, sendo testadas duas diferentes técnicas de extração de atributos e seis diferentes técnicas de classificação por aprendizado de máquina. Três diferentes datasets foram usadas para avaliação dos modelos, incluindo notícias falsas e verdadeiras assim como avaliações falsas e verdadeiras. Os resultados obtidos para os datasets usados foram melhores do que as alternativas que já existiam para os mesmos datasets.

Uma importante distinção a se fazer é a de modelos de detecção baseados em conteúdo versus modelos de detecção baseados em comportamento do avaliador, já que essas são as duas possíveis estratégias a serem tomadas para detecção de avaliações falsas, além da estratégia mista. Um dos principais modelos baseados em conteúdo foi desenvolvido por Ott et al. [2], onde foram usados atributos n-gramas de palavras para detectar avaliações falsas. O dataset, que também foi construídos por eles, consiste de 800 avaliações falsas de hotéis, coletadas de Amazon Mechanical Turk, e 800 avaliações verdadeiras coletadas de TripAdvisor. Além disso, metade das avaliações falsas tem cunho positivo e metade tem cunho negativo, o mesmo se aplica às avaliações verdadeiras. Esse dataset veio a ser usado para avaliação de diversos modelos, inclusive, o desenvolvido por Ahmed et al. [1]. A proposta deste artigo é reproduzir o modelo criado por Ahmed et al. [1] que obteve melhor desempenho com o dataset criado por Ott et al. [2].

2 Metodologia

2.1 Visão Geral

Um fluxograma, representando as etapas percorridas até se chegar na detecção de conteúdo falso, usado por Ahmed et al. [1] pode ser visto na Figura 1. A partir do dataset de Ott et al. [2] as avaliações passam por uma etapa de pré-processamento, os atributos n-grama de palavras são então adquiridos, uma ma-

triz de atributos por documento é gerada e então o o classificador é treinado e testado.

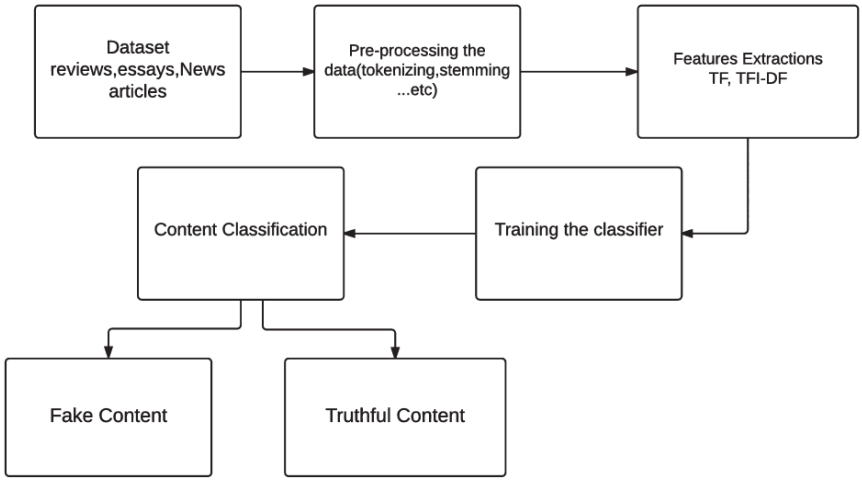

Figura 1. Fluxograma do processo de classificação. Fonte: Ahmed et al. [1]

2.2 Pré-processamento

Antes dos atributos serem extraıdos das avaliações é necessário fazer pré-processamento dos dados, uma série de etapas que diminui o volume de dados, retirando informação irrelevante do dataset. Um script em Bash foi criado para executar todas essas etapas de pré-processamento para todos os documentos do dataset. A ordem da maioria dessas etapas pode ser modificada para a realização de diferentes análises ao longo do processo. Essa ordem foi escolhida por conveniência.

2.3 Segmentação em Frases e Conversão para Caixa Baixa

Para iniciar o pré-processamento foi feita a segmentação das frases em cada um dos documentos. Essa etapa é essencial para manter a estrutura sintática dos dados quando for feita a tokenização em n-gramas de palavras. Segmentação de frases não é um problema tão trivial, já que po . Uma função foi escrita em C, simulando a seguinte expressão regular, usada para identificar o fim de uma frase e inicio de outra:

[.?!;]”?)?[ ](?[A-Z]

O que significa ”seleciona toda pontuação, seguida ou não de aspas duplas, seguida ou não de parêntesis da direita, seguida de um espaço, seguido ou não de

parêntesis da esquerda, seguido de caracter em caixa alta". Após a segmentação de frases uma função em C foi escrita para transformar todos os caracteres caixa alta em caixa baixa.

2.4 Retirada de Pontuação e Palavras Vazias

Palavras vazias são as palavras mais comuns de uma lıngua, que acabam gerando ruıdos quando são usadas como atributos para classificação de texto. Alguns exemplos de palavras vazias da lıngua portuguesa são "um", "uma", "a", "o", "os", "do", "da", etc. Antes das palavras vazias serem retiradas as pontuações são retiradas. Todas as pontuações exceto aspas simples foram retiradas nesta etapa, pois algumas palavras vazias em inglês, que é a lıngua do dataset utilizado, possuem aspas simples (ex.: *won't*) e retirá-las nos documentos significaria que elas não seriam identificadas como palavras vazias. Uma outra solução seria tirar todas as aspas simples do documento que contém a lista de palavras vazias e dos documentos contendo as avaliações.

Um problema encontrado na hora da retirada de pontuação se deu devido a erros de digitação. Por exemplo, uma das avaliações contém o seguinte trecho: "I am very easy going about amenities, cleanliness, and the like...however (...)" Se escrito corretamente, entre as reticências e a próxima palavra, haveria um espaço, porém esse não foi o caso. Uma função em C foi escrita para retirar as pontuações, porém, uma adição necessária é o tratamento desses casos, atualmente as palavras ficam coladas.

Após as pontuações serem retiradas, é feita a retirada de palavras vazias. Uma função em C também foi escrita realizar essa tarefa.

2.5 Segmentação em n-gramas

Um n-grama é um conjunto de n itens que aparecem contiguamente nos dados. O modelo n-grama é uma estratégia amplamente usada na área de modelagem e processamento de linguagem, principalmente n-gramas de palavras e caracteres. Por exemplo, a segmentação em digramas (2-gramas) da frase "My name is Daniel"seria "My name", "name is", "is Daniel". Nesta etapa foi feita a tokenização em n-gramas de palavras por frases, onde cada n-grama será um atributo para futura análise e classificação dos documentos. Seja $F(n, k)$ a quantidade de n-gramas numa frase com k palavras. Então,

$$F(n,k) = \begin{cases} k-(n-1), & k \geq n \\ 0, & k < n \end{cases}$$

Para executar esta etapa foi escrita uma função em C que divide as palavras de cada frase em n-gramas, para um valor inteiro de n qualquer, respeitando $F(n, k)$. Uma adição interessante ao algoritmo seria uma funcionalidade que permitisse a visualização dos n-gramas que mais apareceram nas avaliações.

2.6 Stemização

Após os dados serem segmentados em n-gramas só uma etapa falta para atingir o formato ideal para extração de atributos: stemização. A técnica de stemização (ou *stemming*) consiste na transformação de palavras para um formato de base, tal que palavras relacionadas sejam mapeadas para a mesma base. Por exemplo, o algoritmo ideal de *stemming* para a lıngua inglesa transformaria as palavras "began", "begun"e "beginning" em "begin". Essa etapa é importante pois diminui a quantidade de classes nos dados enquanto mantém boa parte do aspcto semântico, o que é importante para etapas futuras de processamento. Para executar a stemização, o Porter stemmer [6] foi usado, assim como em Ahmed et al. [1].

2.7 Análise das classes de palavras

Uma etapa importante para a compreensão das características linguísticas de avaliações falsas versus avaliações verdadeiras é a análise das diferentes classes de palavras presentes nas avaliações. Para a realização dessa análise foram montadas listas das seguintes classes de palavras: verbos, advérbios, adjetivos, palavras que podem ser tanto verbos quanto substantivos (existem diversas dessas na língua inglesa) e palavras funcionais.

Duas classes são especialmente úteis para essa análise: palavras funcionais e palavras de conteúdos.

Palavras funcionais são palavras com pouco sentido léxico, cuja função é expressar relação gramatical entre palavras em uma frase ou especificar o sentimento do orador. Não é facil adicionar novas palavras funcionais a uma língua, logo se trata de uma classe fechada. Exemplos de subclasses de palavras funcionais são:

- Preposições (ex.: de, em, sem, entre);
- Pronomes (ex.: ele, elas);
- Determinadores (ex.: o, a, um, meu);
- Conjunções (ex.: e, quando, enquanto, ou);
- Verbos auxiliares (ex.: ser, estar, ter);
- Partículas (ex.: se, como, não);

Palavras de conteúdo são todas as palavras que não são palavras funcionais e é uma classe fechada. Exemplos de subclasses de palavras de conteúdo são:

- Substantivos (ex.: Daniel, sala, tarefa);
- Adjetivos (ex.: atento, feliz, alto);
- Advérbios (ex.: muito, realmente, completamente);
- Verbos não-auxiliares (ex.: segurar, implementar, traduzir);

Um algoritmo em C foi escrito para receber como entrada os nomes dos arquivos contendo as listas das classes de palavras e contar, para cada avaliação falsa e verdadeira, quantas palavras eram de cada classe. Algumas adições que seriam interessantes para o algoritmo seria o cálculo da média e variância da porcentagem de cada classe de palavra por algoritmo falso e verdadeiro, assim como uma funcionalidade para possibilitar a visualização desses valores.

2.8 Extração de atributos

Ahmed et al. [1] investigaram dois diferentes tipos de atributos: Term frequency e Term frequency-inverted document frequency.

Term frequency (TF) é uma forma de estimar a similaridade entre documentos a partir da contagem de termos que aparecem em cada um. O vocabulário inteiro é calculado a priori e, para cada documento, um vetor com tamanho equivalente ao número de termos no vocabulário é inicializado. Para cada documento é feita uma contagem de quantas vezes aparecem cada termo do vocabulário.

Seja $f_{i,j}$ a frequência do termo i no documento j e seja V o conjunto de todos os termos de indexação e D o conjunto de todos os documentos. Após ser calculado $f_{i,j}$, $\forall i \in V$, $\forall j \in D$, é feita a normalização do vetor de frequência de termos, para cada documento, tal que a soma dos termos de cada um é igual a um. Seja $n_{i,j}$ a quantidade de vezes que o termo i aparece no documento j. O tamanho do documento j é $|j| = \sum_{i \in j} n_{i,j}$.

Seja $TF_{i,j}$ igual a $f_{i,j}$ após a normalização, temos que

$$TF_{i,j} = \frac{n_{i,j}}{|j|}$$

Term frequency-inverted document frequency (TF-IDF) é uma métrica usada para calcular a importância de um termo em um documento pertencente a um dataset. Quanto mais um termo aparece num documento, maior sua importância para esse documento, porém, se esse termo também aparece muito em outros documentos do dataset, menos importante ele é.

O *inverse document frequency*, ou frequência inversa do documento (IDF) de um termo i referente ao conjunto de todos os documentos D (notação: $IDF_{i,D}$) é calculado da seguinte forma:

$$IDF_{i,D} = 1 + \ln \frac{|D|}{|i \in d|}, \forall d \in D$$

TF-IDF do termo i referente ao documento $d \in D$ é calculado da seguinte forma:

$$TF - IDF_{i,d,D} = TF_{i,j} \times IDF_{i,D}$$

Esta seção não chegou a ser implementada completamente. Um algoritmo em Python foi escrito, usando a biblioteca *scikit-learn*, para definir o vocabulário de

termos e fazer a contagem dos termos em cada documento, porém, os atributos TF e TF-IDF não foram extraídos.

2.9 Classificação

Ahmed et al. [1] realizaram o treino de um classificador para inferir se um documento é verdadeiro ou falso, a partir dos atributos TF e TF-IDF serem extraídos na etapa anterior.

O dataset foi dividido em conjuntos de teste e de treino, tal que 80% foi usado para treino e 20% para teste, já que foi feita validação cruzada 5-fold, que consiste num processo iterativo de 5 etapas, onde, a cada etapa, 20% do dataset é usado para teste e 80% é usado para treino, tal que nenhum documento é usado para teste em mais de uma etapa. O processo pode ser visualizado na Figura 2.

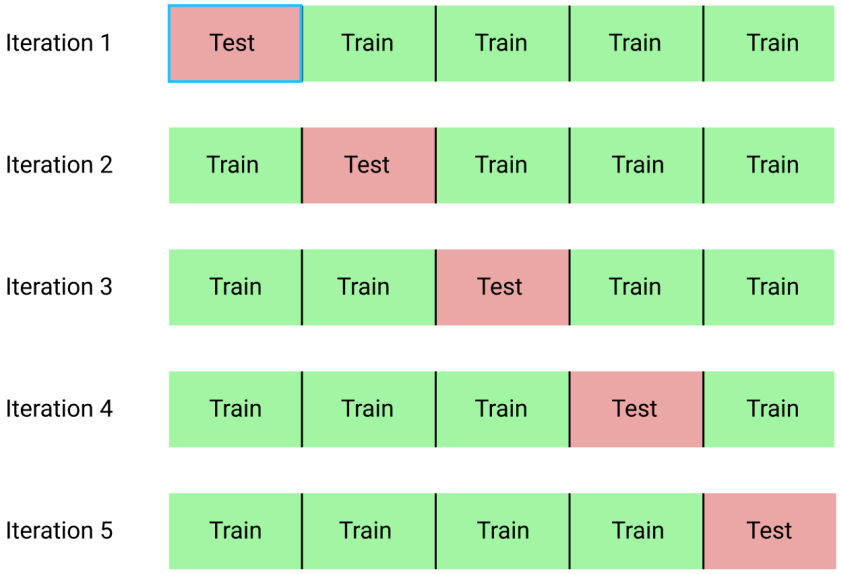

Figura 2. Processo de valicação cruzada 5-fold

Após a extração de todos os atributos, seja por TF ou TF-IDF, de todos os termos j referentes a cada um dos m documentos i no conjunto de treino e a seleção dos p termos com valores mais altos, a matriz de atributos $X = [x_{ij}]_{1 \leq i \leq m, 1 \leq j \leq p}$ é construída, onde

$$\begin{cases} x_{ij} = atributo_j & j \in i \\ x_{ij} = 0, & c.c. \end{cases}$$

Essa seção não foi implementada. Ahmed et al. [1] implementaram usando a biblioteca *scikit-learn* do Python. Eles também usaram 6 diferentes implementações de classificadores do *natural language toolkit* (NLTK) do Python, para inferir as classes de documentos:

- Stochastic Gradient Descent (SGD);
- Support Vector Machine (SVM);
- Linear Support Vector Machine (LSVM);
- Linear Regression (LR);
- K-nearest neighbours (KNN);
- Linear Regression (LR);
- Decision Tree (DT).

3 Dificuldades

O algoritmo criado para executar as etapas de pré-processamento em todas as avaliações do dataset está atualmente apresentando um problema na leitura dos arquivos de avaliação quando é executado fora do diretório que contém as avaliações. Apesar do algoritmo de pré-processamento, de análise e, parcialmente, o de extração de atributos, estarem funcionais não foi possível executar o script escrito em Bash para executar as etapas, recebendo como entrada todas as avaliações. Uma solução a curto prazo seria copiar os arquivos executaveis para cada diretório que contém as avaliações e executá-los em cada um desses diretórios, entretanto isso não foi feito.

4 Resultados e conclusão

O dataset criado por Ott et al. [2] é constituído de 800 avaliações verdadeiras e 800 avaliações falsas, todas escritas por humanos. Foi contestado que as avaliações falsas, por terem sido fabricadas com a intenção de serem analisadas, não representam uma amostra fiel de avaliações falsas encontradas organicamente. Ainda assim, os resultados obtidos por Ahmed et al. [1] são impresisonantes. Segmentando os termos em digramas e treinando o modelo de LSVM com os 10000 atributos com valores mais altos, eles conseguiram uma taxa de acerto de 90% nas classificações, um pouco melhor do que os 89,3% alcançados por Ott et al [2] com o mesmo dataset e bem melhor do que o melhor humano, que classificou corretamente 65% das vezes.

Os resultados obtidos por Ahmed et al. [1] não foram reproduzidos pois as etapas necessárias não foram completas, logo, a proposta apresentada neste artigo não foi concluída.

Referências

1. Ahmed, H.: Detecting opinion spams and fake news using text classification. Em: Obaidat, M. (ed.) Security and Privacy, Volume 1, Issue 1, e9 (2017) https://doi.org/10.1002/spy2.9

2. Ott, M.: Finding deceptive opinion spam by any stretch of the imagination. HLT '11 Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1, pp. 309–319.
   Portland, Oregon (2011)
3. Página da ementa "Recuperação da Informação"
   http://dcc.ufrj.br/~giseli/2018-1/ri
   Rabelo, G.
4. Definição do Trabalho 2
   http://dcc.ufrj.br/~giseli/2018-1/ri/Definicao_Trabalho2.pdf
   Rabelo, G.
5. Github com código
   https://github.com/danielatk/fake-detector
   Atkinson, D.
6. Porter Stemmer
   https://tartarus.org/martin/PorterStemmer
   Porter, M.