

Análise de sentimentos em tweets de avaliação de produtos/serviços

Antonio Miguel de Macedo Lopes, Gabriel André Melo de Oliveira Silva,
Paulo Jin Sol Kim, Yuri Vinicius de Lima Chen

*Escola de Artes, Ciências e Humanidades
Universidade de São Paulo
São Paulo, Brasil*

antoniomigueldemacedo@usp.br, oliveira.8102@usp.br, paulojskim@usp.br, yuri.chen.si@usp.br

Abstract—Nesse trabalho de processamento de texto buscamos desenvolver um algoritmo capaz de classificar o sentimento presente em tweets de avaliação/opinião de produtos/serviços como positivo, negativo, neutro ou indefinido. Para isso, foi usado o algoritmo NMF com o uso do método de humano no loop.

Index Terms—Análise, sentimento, machine learning, clusterização

I. INTRODUÇÃO

Na atual era do Big Data, uma das aplicações de processamento de linguagem natural que vem ganhando destaque é a análise de sentimentos. A análise de sentimentos é essencial para as empresas entenderem melhor o comportamento dos consumidores, para assim auxiliar na tomada de decisão. A análise pode ser empregada de diversas formas: em aplicações de voz do cliente, em respostas de um questionário e em comentários de avaliações, por exemplo. Em nosso trabalho, buscamos classificar o sentimento, sendo este positivo, negativo ou neutro (além da possibilidade de não haver como indicá-lo), de postagens da rede social Twitter, os tweets, que emitem opinião sobre produtos da Apple ou Google. Para isso então, usamos o algoritmo de fatoração matricial não negativa (Non-negative matrix factorization - NMF) com o método de humano no loop para remover as stopwords.

II. DEFINIÇÃO DO PROBLEMA

Nosso trabalho buscou desenvolver um algoritmo com o método de humano no loop que consiga identificar o sentimento presente em tweets que emitem opinião sobre algum produto/serviço. Para isso, utilizamos métodos de mineração de textos para conseguir analisar cada tweet de usuários. Por exemplo, utilizamos a tokenização das palavras de cada tweet, utilizamos o vetorizador TfIdf e stop words. Para testarmos o algoritmo que resolva o problema em questão, trabalhamos com um dataset que contém tweets que foram avaliados se emitem opinião de sentimento positivo ou negativo sobre produtos Apple ou Google. Há 9093 linhas no dataset, cada linha possui os atributos:

- **tweet_text**: texto do tweet em questão.
- **emotion_in_tweet_is_directed_at**: qual produto o tweet se refere.

- **is_there_an_emotion_directed_at_a_brand_or_product**: qual emoção o tweet tem, que pode ser positiva, negativa, nenhum dos dois ou indefinido.

III. ARQUITETURA DA SOLUÇÃO

A. Ferramentas

Para o trabalho usamos a linguagem Python juntamente com as seguintes bibliotecas para a visualização e manipulação dos dados:

- Pandas: biblioteca de Python para manipulação e análise de dados. Em particular, oferece estruturas e operações para manipular tabelas, chamado de Dataframes;
- Matplotlib.pyplot: coleção de funções que fazem o matplotlib, biblioteca para criar gráficos em Python;
- Seaborn: biblioteca de visualização de dados Python baseada em matplotlib.

Além disso, utilizamos o Google Colab como a nossa 'IDE' para desenvolvimento do código, pois nela é possível compartilhar o código em real time com os colegas e desenvolver em conjunto, além de ser uma plataforma em nuvem, em que se pôde utilizar os recursos computacionais disponibilizados pela Google.

B. Importando o Dataset

Primeiramente importamos as bibliotecas citadas e o dataframe:

Algoritmo 1

```
0: import zipfile
0: import pandas as pd
0: import seaborn as sbn
0: import matplotlib.pyplot as plt
0: import re
0: import string
0: from pandas.core.frame import DataFrame
0: twitter_path = 'https://github.com/hype-usp/Grupos-de-estudos/blob/main/products.zip?raw=true'
0: twitter_data = pd.read_csv(twitter_path, compression='zip')
0: df = twitter_data = 0
```

```
imports_l.png
```

Fig. 1. Imports e carregando dataset.

Além de importar o dataset, nós precisamos importar todas as bibliotecas utilizadas, como por exemplo o scikitlearn, nltk, re, string e etc.

```
imports.png
```

Fig. 2. Imports das bibliotecas.

E também, foi necessário fazer download manualmente de alguns pacotes. O porque disso, não sabemos porém tivemos que fazer para que o programa funcionasse.

```
download.png
```

Fig. 3. Download dos pacotes extras.

C. Estatísticas dos atributos

Depois de importar o dataset, verificamos quantos tweets foram avaliados como positivos, negativos, neutros e indefinidos:

Algoritmo 2

```
0: value_count = df['is_there_an_emotion_directed_at_a_brand_or_product'].value_counts()
0: value_count = 0
```

TABLE I
CONTAGEM DOS SENTIMENTOS POSITIVOS, NEGATIVOS, NEUTROS E INDEFINIDOS

Atributos	Valores
No emotion toward brand or product	5389
Positive emotion	2978
Negative emotion	570
I can't tell	156

Há no dataset, 2978 tweets com sentimento positivo, 570 negativos, 5389 neutros (No emotion toward brand or product) e 156 indefinidos (I can't tell).

Também verificamos quais produtos aparecem nos tweets:

Algoritmo 2

```
0: df['emotion_in_tweet_is_directed_at'].unique() = 0
```

Então descobriu-se que há 10 resultados distintos: 'iPhone', 'iPad or iPhone App', 'iPad', 'Google', 'Android', 'Apple', 'Android App', 'Other Google product or service' (outro produto ou serviço Google), 'Other Apple product or service' (outro produto ou serviço Apple) ou 'nan' (nenhum produto).

```
unique_emotion.png
```

Fig. 4. Quais produtos estão sendo analisados.

D. Non-negative matrix factorization (NMF)

O NMF foi introduzido pela primeira vez por Paatero e Tapper em 1994, e popularizado em um artigo de Lee e Seung em 1999. Desde então, o número de publicações referenciando a técnica cresceu rapidamente.

O NMF é um algoritmo de fatoração de matrizes, ou seja, decompõe uma matriz como uma aproximação da multiplicação de duas outras matrizes. Esta é a definição formal: dada uma matriz não negativa V , encontrar fatores de matriz não negativos W e H tais que:

$$V = W * H \quad (1)$$

Sendo que W é a matriz que contém os tópicos, agrupamentos de atributos, e H é a matriz que possui os pesos desses tópicos.

NMF é útil quando há muitos atributos e os atributos são ambíguos ou têm baixa previsibilidade. Ao combinar atributos, o NMF pode produzir padrões, tópicos ou temas significativos. O NMF tem aplicações no processamento de imagens faciais por exemplo:

```
nmf-facial-processing.png
```

Fig. 5. NMF em processamento de imagens faciais

Neste caso, a matriz W é composta por características como olhos, narizes, bigodes e lábios, enquanto as colunas de H indicam quanto da característica está presente na imagem. E claro, o NMF tem aplicação para o processamento de linguagem, que é o tipo de problema que nosso trabalho busca

resolver. No processamento de texto, cada linha da matriz V corresponde a uma palavra e cada coluna a um documento (texto).

O NMF produzirá duas matrizes W e H . As colunas de W são grupos de palavras, ou seja, os tópicos, que são conjuntos de palavras encontradas simultaneamente em diferentes documentos. Então a matriz H nos diz como somar contribuições de diferentes tópicos para reconstruir a mistura de palavras de um determinado documento de texto original. Resumindo:

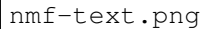


Fig. 6. NMF em processamento de texto

o NMF traduz um conjunto de documentos (texto, imagem) indiferenciados, sem padrão, em documentos compostos por tópicos, que são agrupamentos de termos (palavras, pixels). Portanto, o NMF é útil para diminuir a dimensionalidade e identificar características.

Ademais, o NMF possui em si uma propriedade de poder ser utilizado para clustering, pois realiza esse agrupamento nas colunas de V , além do fato de W poder indicar os centroides do grupo.

$$\mathbf{V} = (v_1, \dots, v_n)$$

E. Implementando o NMF

Para implementar o NMF precisamos importar o objeto fornecido pela biblioteca `scikitlearn`. Após isso, precisamos instanciar um objeto do NMF com um parâmetro no construtor que é o número de clusters que voce deseja.

Com isso em mãos, utilizamos os dados pré-processados para que o NMF consiga criar os clusters. Para isso, utiliza-se as funções `.fit` e `.fit_transform` injetando os dados como parâmetro.

Algoritmo 3

```
0: nmf_model = NMF(4)
0: topic = nmf_model.fit(data_dtm_noun)
0: doc_topic = nmf_model.fit_transform(data_dtm_noun) = 0
```

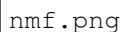


Fig. 7. Criação do modelo do NMF.

Além disso, é necessário realizar todo um pré-processamento nos dataset para que o algoritmo tenha uma melhor precisão, por exemplo retirando as colunas com elementos vazios e tratando cada palavra do texto transformando-os em substantivos, verbos ou adjetivos.

Primeiramente, nós tivemos que retirar os caracteres especiais, pontuações e símbolos para que os tweets sejam mais legíveis para o algoritmo. Para isso, nós criamos uma função chamada `clean_text` mostrado a seguir.

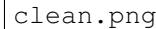


Fig. 8. Função que limpa os tweets.

Segundo, nós retiramos todas as palavras que se referenciavam a marca, como 'iPhone', 'iPad', 'Apple', 'Google' e etc. Dessa forma, essas palavras não terão influencia no nosso modelo.

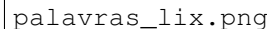


Fig. 9. Limpando as palavras que referenciam marca.

Por último, nós transformamos todas as palavras em substantivos para que não haja conflito entre as palavras que estão no passado, presente ou futuro. Assim fica mais fácil para o algoritmo encontrar os clusters certos. Além disso, foi necessário tokenizar cada uma das palavras utilizando o `word_tokenize` da biblioteca `nlTK`.

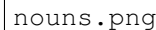


Fig. 10. Função que transforma as palavras em substantivo.

Terminado o pré-processamento, criamos o modelo utilizando o algoritmo NMF. Após isso, plotamos o gráfico dos clusters utilizando a biblioteca `matplotlib`.

Algoritmo 4

```
0: A = np.concatenate((np.array( [ [1,3,10] +
np.random.rand(3)-.5 for i in range(25) ] ), np.array(
[ [3,7,2] + np.random.rand(3)-.5 for i in range(25) ]
), np.array( [ [5,1,10] + np.random.rand(3)-.5 for i in
range(25) ] ), np.array( [ [7,4,5] + np.random.rand(3)-.5
for i in range(25) ])), axis = 0)
0: H = topic.components_
0: h = np.argmax(H,0)
```

```

0: fig = plt.figure()
0: ax = Axes3D(fig)
0: fig.add_axes(ax)
0: ax.scatter(A[h==0][:,0],A[h==0][:,1],A[h==0][:,2],c='r', label='Topic 0');
0: ax.scatter(A[h==1][:,0],A[h==1][:,1],A[h==1][:,2],c='b', label='Topic 1');
0: ax.scatter(A[h==2][:,0],A[h==2][:,1],A[h==2][:,2],c='y', label='Topic 2');
0: ax.scatter(A[h==3][:,0],A[h==3][:,1],A[h==3][:,2],c='g', label='Topic 3');
0: ax.legend() =0

```

plot.png

Fig. 11. Algoritmo para plotar o gráfico 3D.

F. Humano no loop

Para inserir o humano no loop, criamos stop words customizados para a nossa aplicação. Assim, retira-se todas as palavras que podem confundir nosso algoritmo, resultando assim em uma precisão baixa.

humano.png

Fig. 12. Trecho de código com humano no loop.

Além disso, após a inserção do humano no loop o nosso modelo ficou com 20 dimensões. Por outro lado, no modelo sem o humano no loop ficou com 104 dimensões.

No caso quando não estamos utilizando o humano no loop, o mesmo trecho de código ficaria sem a variável `stop_noun` e utilizando apenas os stop words padrão. O código ficaria como na figura abaixo.

G. Observação de cada cluster

Após criar o modelo, nós fizemos uma função chamada `display_topics` na qual ela mostra as principais palavras que influenciaram na criação daquele tópico. Os parâmetros

stop_words_sem_humano.png

Fig. 13. Trecho de código sem humano no loop.

dessa função são: `model`, `feature_names`, `num_top_words` e `topic_names`.

display.png

Fig. 14. Função de mostrar os tópicos.

- **model:** é o nosso modelo criado, neste caso o modelo gerado pelo algoritmo NMF.
- **feature_names:** é o nome de cada feature do modelo.
- **topic_names:** é o nome de cada tópico, caso voce queira dar um nome específico.

Com isso, foi possível mostrar cada tópico no 'console' do Google Colab.

topics.png

Fig. 15. Tópicos mostrados após o print.

IV. CONFIGURAÇÃO DE TESTES

A configuração dos nossos testes foi feita a cada bloco de código no Google Colab que foi compartilhado junto com este relatório. Então, temos cada bloco de código que foi dividido por configuração de testes. Para executar o teste, basta executar cada bloco de código sequencialmente.

Cada bloco de código foi estruturado conforme a figura abaixo. E caso seja necessário rodar o teste, deve-se abrir cada uma das seções e rodar os blocos de códigos inseridos dentro dele. Mas lembrando que precisamos rodar sempre a seção dos "Passos Iniciais" antes de qualquer teste.



```
steps.png
```

Fig. 16. Blocos de código separados por teste.

V. RESULTADOS OBTIDOS

Em primeiro momento, foi realizado um teste, e, que se foi realizado o plot o do modelo utilizando o algoritmo NMF separados em cada cluster. Para melhorar o modelo e utilizar o *humano no loop*, manipulamos os stop words, acrescentando as palavras que consideramos desnecessárias para fins de agrupamento. Após isso, realizamos um novo agrupamento e fizemos um plot do novo modelo para observar o resultado.



```
sem_humano.png
```

Fig. 17. Sem humano no loop.

VI. ANÁLISE DOS RESULTADOS

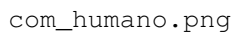
Não foi possível analisar a melhoria em relação à inserção do humano no loop para a aplicação, pois não conseguimos terminar todo o processo e não conseguimos realizar a comparação entre os dois modelos.

Só conseguimos plotar os dois clusteres, um sem o humano no loop e o outro com o humano no loop, porém sem nenhuma comparação paupável e direta para os dois modelos criados.

Para isso, nós tínhamos criado uma seção para a comparação de cada modelo conforme a figura a seguir. Porém, como mencionado, não foi possível realizar essa comparação.

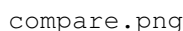
VII. CONCLUSÃO

A utilização de humano no loop no problema em questão -escolher manualmente as palavras a serem consideradas como stop words- foi de fundamental importância para uma mudança significativa nos resultados, demonstrando a força



```
com_humano.png
```

Fig. 18. Com humano no loop.



```
compare.png
```

Fig. 19. Seção de comparação.

que a utilização da técnica que humano no loop possui, aumentando ainda mais a necessidade de expandir cada vez mais o uso dessa técnica para estudos comparativos, a fim de melhorar explicabilidades e também métricas estatísticas como a precisão. É notório também o fato de que a utilização de humano no loop funciona bem em conjunto com o modelo de Non-Negative Matrix Factorization (NMF). A utilização dessa técnica não deve limitar-se apenas a esse modelo mas também a modelos como o k-means e também modelos de classificação, pois humano no loop não se restringe ao uso junto de técnicas de agrupamento.

REFERENCES

- [1] GARLA, Anupama. NMF — A visual explainer and Python Implementation. Disponível em: <https://towardsdatascience.com/nmf-a-visual-explainer-and-python-implementation-7ecdd73491f8>.
- [2] Non-negative/ matrix/ factorization. https://en.wikipedia.org/wiki/Non-negative_matrix_factorization
- [3] Prakharr0y. Non-Negative Matrix Factorization. <https://www.geeksforgeeks.org/non-negative-matrix-factorization/>
- [4] Visualization of NMF (Nonnegative Matrix Factorization) Learning. <https://linuxtut.com/en/d03b03a05dac415ee89c/>
- [5] GAUJOUX, Renaud. Generating heatmaps for Nonnegative Matrix Factorization. Disponível em: <https://cran.r-project.org/web/packages/NMF/vignettes/heatmaps.pdf>