

Trabalho de Álgebra Linear

Similaridade por cosseno aplicada para recomendação de vinhos

Matheus Moura Justino

20 de maio de 2025

1. Introdução

Este relatório descreve o funcionamento de um algoritmo em python desenvolvido para recomendação de vinhos, utilizando técnicas de Processamento de Linguagem Natural (PLN) e Álgebra Linear.

Com o objetivo de utilizar uma base de dados sólida e confiável, foi escolhido o *dataset Wine Tasting*, disponibilizado pela *Maven Analytics*. Ele contém 130 mil avaliações de vinhos publicadas no site *Wine Enthusiast*, oferecendo uma ampla quantidade de texto, requisito essencial para o funcionamento do algoritmo.

2. Instalação e Importação de Bibliotecas

Primeiramente, serão instaladas as bibliotecas necessárias: Pandas, Natural Language Toolkit (NLTK), Scikit-Learn e Numpy. Para uma melhor organização, foi criado um arquivo chamado `requirements.txt` contendo todas essas dependências.

```
%pip install -r "../requirements.txt"
```

Em seguida, serão importadas as bibliotecas, módulos e funções utilizadas e também feito o download do modelo de tokenização pré-treinado chamado "punkt", além do pacote contendo as *stopwords*, ambos disponibilizados pelo NLTK:

```
import pandas as pd

import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

import numpy as np

nltk.download('punkt')
nltk.download('stopwords')
```

3. Leitura e Limpeza dos Dados

Os dados são carregados de um arquivo CSV. Após a leitura, são selecionadas apenas as colunas relevantes para a aplicação do algoritmo: *Designation* (nome do vinho) e *Description* (avaliação escrita do vinho).

Em seguida, os dados são tratados, removendo suas linhas duplicadas e valores nulos:

```
df_raw = pd.read_csv("../data/winemag-data-130k-v2.csv")

df = df_raw[["designation", "description"]]

df = df.drop_duplicates().dropna().reset_index(drop=True)
```

4. Remoção de *Stopwords*

Stopwords são palavras comuns que não agregam valor semântico ao texto. Por isso, sua remoção foi realizada a fim de aumentar a eficiência do algoritmo.

Primeiramente, é criada uma variável contendo todas as *stopwords* fornecidas pelo pacote NLTK. Em seguida, uma função que remove as *stopwords* e caracteres não alfabéticos de uma coluna da base de dados, neste caso, a coluna *description*. Por fim, é criada uma nova coluna chamada *clean_description*, contendo as descrições sem *stopwords*, mantendo assim a coluna com as descrições originais intacta:

```
stop_words = set(stopwords.words("english"))

def remove_stopwords(column: pd.Series) -> pd.Series:
    return column.apply(
        lambda s: " ".join(
            [word for word in word_tokenize(s)
             if word.lower() not in stop_words and word.isalpha()]
        )
    )

df["clean_description"] = remove_stopwords(df["description"])
```

5. Vetorização com TF-IDF

O código a seguir transforma as descrições limpas em vetores numéricos utilizando a técnica TF-IDF (Term Frequency - Inverse Document Frequency), armazenando o resultado em uma nova variável, uma matriz onde cada linha representa uma descrição e cada coluna, uma palavra única, considerando todas as descrições da base de dados.

Diferentemente do *CountVectorizer*, que apenas conta a frequência das palavras, o *TfidfVectorizer* também considera a frequência, atribuindo pesos a cada palavra com base em sua aparição.

A função *fit transform* transforma cada texto (descrição de vinho) em um vetor numérico com os valores de TF-IDF, esses valores são armazenados em uma variável.

```
vectorizer = TfidfVectorizer()

df_matrix = vectorizer.fit_transform(df["clean_description"])
```

6. Entrada de Dados do Usuário

Duas variáveis são criadas: a primeira armazena uma lista com os nomes dos vinhos disponíveis (em letras minúsculas), e a segunda contém três vinhos aleatórios, que servirão como sugestões ao usuário:

```
available_wines = [w.lower() for w in df["designation"].values]

random_wines = df.sample(3)["designation"].values.tolist()
```

Em seguida, inicia-se um *loop* que exibe as três sugestões e solicita ao usuário que digite o nome de um vinho de sua preferência (ou "000" para selecionar um aleatório). Após a escolha, o nome do vinho selecionado é exibido:

```

while True:
    print("\nSugestoes de vinhos:")
    for wine in random_wines:
        print(f"- {wine}")

    chosen_wine = input("\nDigite o nome de um vinho que voce goste (ou
        digite 000 para aleatorio): ").strip().lower()

    if chosen_wine == "000":
        chosen_wine = df.sample()["designation"].str.lower().values[0]
        print(f"\nVinho aleatorio selecionado: {chosen_wine}")
        break
    elif chosen_wine in available_wines:
        print(f"\nVinho selecionado: {chosen_wine}")
        break
    else:
        print("Vinho nao disponivel. Tente novamente.")
        random_wines = df.sample(3)["designation"].values.tolist()

```

7. Aplicação do Algoritmo de Similaridade

Nesta etapa, as informações do vinho escolhido (nome e descrição) são armazenadas em uma variável, o que facilita a recuperação posterior dessas informações.

```

chosen_wine_line = df.loc[df["designation"].str.lower() == chosen_wine].
    iloc[0]

```

Em seguida, remove-se a linha correspondente ao vinho escolhido do *dataset* inicial, garantindo que ele não apareça entre os recomendados:

```

df = df[df["designation"].str.lower() != chosen_wine]

```

Agora será criada uma lista com todas as descrições limpas, onde o último elemento será a descrição do vinho escolhido, essa lista será transformada em uma grande matriz:

```

to_compare_list = df["clean_description"].tolist() + [chosen_wine_line["
    clean_description"]]

to_compare_matrix = vectorizer.transform(to_compare_list)

```

Por fim, a função *cosine_similarity* calcula o produto escalar normalizado entre o vetor de referência (descrição do vinho escolhido) e todos os outros (descrição de todos os vinhos):

$$(X, Y) = \frac{X \cdot Y}{\|X\| \cdot \|Y\|}$$

```

cosine_sim = cosine_similarity(to_compare_matrix[-1], to_compare_matrix
   [:-1])

```

8. Retorno dos Resultados

Após calcular a similaridade por cosseno entre vetores, foi utilizada a função `np.argmax()` para identificar qual vetor é mais semelhante ao vetor de referência.

A função `argmax`, fornecida pelo Numpy, retorna o índice do maior valor dentro da matriz. Como a ordem dos valores na matriz não foi alterada, o índice é o mesmo do *dataset* original.

```
most_sim = np.argmax(cosine_sim)
```

Sabendo disso, foi feito um *output* simples para o resultado do algoritmo. Primeiro é exibido novamente o nome do vinho digitado, seguido de sua descrição.

Abaixo, é mostrado o nome do vinho semelhante, segundo o algoritmo, também com sua descrição.

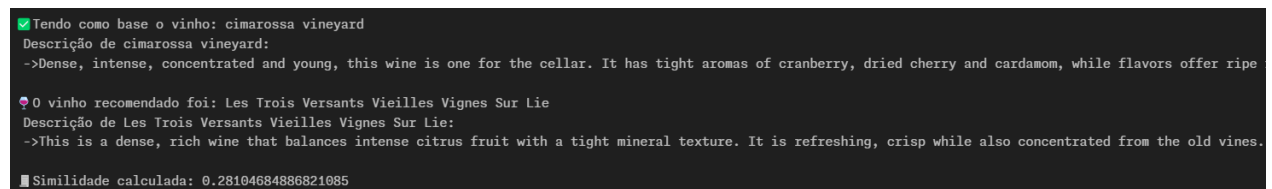
Por fim, é exibido o valor do cálculo da similaridade. Lembrando que, quanto mais próximo de 1, mais semelhantes são os vetores (vinhos):

```
print(f"Tendo como base o vinho: {chosen_wine}")
print(f"Descrição de {chosen_wine}: -> {chosen_wine_line['description']}")

print(f"O vinho recomendado foi: {df['designation'].iloc[most_sim]}")
print(f"Descrição de {df['designation'].iloc[most_sim]}: -> {df['description'].iloc[most_sim]}")

print(f"Similaridade calculada: {cosine_sim[0, most_sim]}")
```

Abaixo, uma imagem da execução do algoritmo, com a seleção de um vinho aleatório como ponto de partida:



```
✓ Tendo como base o vinho: cimarossa vineyard
Descrição de cimarossa vineyard:
->Dense, intense, concentrated and young, this wine is one for the cellar. It has tight aromas of cranberry, dried cherry and cardamom, while flavors offer ripe

🍷 O vinho recomendado foi: Les Trois Versants Vieilles Vignes Sur Lie
Descrição de Les Trois Versants Vieilles Vignes Sur Lie:
->This is a dense, rich wine that balances intense citrus fruit with a tight mineral texture. It is refreshing, crisp while also concentrated from the old vines.

■ Similaridade calculada: 0.28104684886821085
```

Figura 1: Resultado final do algoritmo

Referências

YT: Text Representation Using Bag Of Words (BOW): NLP Tutorial For Beginners - S2 E3

YT: Stop Words: NLP Tutorial For Beginners - S2 E4