CENTRO PAULA SOUZA
FACULDADE DE TECNOLOGIA DE SANTOS – FATEC RUBENS LARA

SISTEMA DE BUSCA E ANÁLISE DE SIMILARIDADE ENTRE ARTIGOS CIENTÍFICOS COM TF-IDF E PCA

JOÃO PEDRO DIAS

Santos – SP
29 de outubro de 2025

Sumário

INTRODUÇÃO

O presente trabalho tem como objetivo desenvolver um sistema de busca e análise de similaridade entre artigos científicos utilizando técnicas de Processamento de Linguagem Natural *(PLN)*. Foram aplicados conceitos de vetorização de textos com *TF-IDF*, cálculo de similaridade de cosseno, redução de dimensionalidade por *PCA* e visualização tridimensional dos vetores. Como resultado, o sistema é capaz de identificar os artigos mais semelhantes a uma consulta textual (*query*) e exibir suas similaridades, ângulos e resumos, além de representá-los graficamente em um espaço 3D.

1 DESCRIÇÃO DO DATASET

O conjunto de dados utilizado foi obtido no *Kaggle*, no repositório *"Topic Modeling for Research Articles"*. O dataset contém milhares de artigos científicos, cada um com título, resumo e o tema relacionado. Neste projeto, foram utilizados os campos de título e resumo, sendo aplicadas técnicas de pré-processamento para limpeza textual e extração de termos relevantes para a análise de similaridade.

2 ESTRUTURA E FUNCIONAMENTO DO SISTEMA

O sistema foi implementado em Python, em um ambiente do Google Colab e estruturado em diferentes etapas. A seguir, cada parte do código é explicada com base nos principais blocos apresentados no *notebook*.

2.1 Importação das Bibliotecas

Nesta etapa, foram importadas as bibliotecas essenciais para o desenvolvimento do sistema, incluindo:

- Manipulação de dados: *pandas*, *numpy*;
- Processamento de texto: *nltk*, *re*;
- Cálculo de TF-IDF e similaridade de cosseno: *scikit-learn*;
- Visualização dos vetores: *matplotlib*.

Além disso, foi realizado a instalação dos recursos do *NLTK*, como o *tokenizador* e as *stopwords*.

```
[ ]
import pandas as pd
import numpy as np
import re
import nltk
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from matplotlib import pyplot as plt
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('punkt_tab')
```

Figura 1: Importação das bibliotecas e instalação dos recursos do NLTK.

2.2 Carregamento e Preparação dos Dados

O conjunto de dados foi carregado, contendo as colunas **TITLE** e **ABSTRACT**. Em seguida, foi criada uma nova coluna denominada **TEXT**, que une ambas as informações, ampliando o contexto disponível para o cálculo de similaridade.

```
[ ]
df1 = pd.read_csv('train.csv', engine='python', on_bad_lines='warn')
df2 = pd.read_csv('test.csv', engine='python', on_bad_lines='warn')
df = pd.concat([df1, df2])
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 11432 entries, 0 to 5726
Data columns (total 9 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   ID                  11432 non-null  int64
 1   TITLE               11432 non-null  object
 2   ABSTRACT            11432 non-null  object
 3   Computer Science    5705 non-null   float64
 4   Physics             5705 non-null   float64
 5   Mathematics         5705 non-null   float64
 6   Statistics          5705 non-null   float64
 7   Quantitative Biology  5705 non-null   float64
 8   Quantitative Finance  5705 non-null   float64
dtypes: float64(6), int64(1), object(2)
memory usage: 893.1+ KB
```

```
df.head()
```

	ID	TITLE	ABSTRACT	Computer Science	Physics	Mathematics	Statistics	Quantitative Biology	Quantitative Finance
0	1	Reconstructing Subject-Specific Effect Maps	Predictive models allow subject-specific inf...	1.0	0.0	0.0	0.0	0.0	0.0
1	2	Rotation Invariance Neural Network	Rotation invariance and translation invarian...	1.0	0.0	0.0	0.0	0.0	0.0
2	3	Spherical polyharmonics and Poisson kernels fo...	We introduce and develop the notion of spher...	0.0	0.0	1.0	0.0	0.0	0.0
3	4	A finite element approximation for the stochas...	The stochastic Landau--Lifshitz--Gilbert (LL...	0.0	0.0	1.0	0.0	0.0	0.0
4	5	Comparative study of Discrete Wavelet Transfor...	Fourier-transform infra-red (FTIR) spectra o...	1.0	0.0	0.0	1.0	0.0	0.0

```
[ ]
df = df[['TITLE', 'ABSTRACT']]
df['TEXT'] = df['TITLE'] + ' ' + df['ABSTRACT']
```

Figura 2: Carregamento do dataset e criação de nova coluna `TEXT`.

2.3 Pré-processamento dos Textos

Foram aplicadas transformações para limpar e normalizar os textos, facilitando o cálculo de similaridade. As etapas incluem: conversão para minúsculas, remoção de pontuação (mantendo hífen), eliminação de espaços extras, *tokenização*(divide o texto em palavras), remoção de *stopwords*(elimina palavras comuns) e *stemming*(reduz palavras à sua forma base).

```
[ ]
def preprocess_text(text):
    # lowercase
    text = text.lower()

    # remover pontuação(exceto hífen) e remover espaços extras
    text = re.sub(r'[^\w\s-]', '', text)
    text = text.strip()

    # tokenização
    from nltk.tokenize import word_tokenize
    tokens = word_tokenize(text)

    # remover stopwords
    from nltk.corpus import stopwords
    stop_words = set(stopwords.words('english'))
    tokens = [word for word in tokens if word not in stop_words]

    # stemming
    from nltk.stem import PorterStemmer
    ps = PorterStemmer()
    tokens = [ps.stem(word) for word in tokens]

    # juntar tudo de novo
    return ' '.join(tokens)
```

```
[ ]
df['CLEAN_TEXT'] = df['TEXT'].apply(preprocess_text)
```

```
[ ]
df['CLEAN_TEXT'].head()
```

	CLEAN_TEXT
0	reconstruct subject-specif effect map predict ...
1	rotat invari neural network rotat invari trans...
2	spheric polyharmon poisson kernel polyharmon f...
3	finit element approxim stochast maxwel – land...
4	compar studi discret wavelet transform wavelet...

dtype: object

Figura 3: Aplicamento da função de pré-processamento dos textos.

2.4 Aplicação do TF-IDF

Após o pré-processamento, aplicou-se o método TF-IDF (Term Frequency–Inverse Document Frequency), que transforma cada documento em um vetor numérico, permitindo mensurar a relevância dos termos.

```
[ ]
    vectorizer = TfidfVectorizer()
    tfidf_matrix = vectorizer.fit_transform(df['CLEAN_TEXT'])
```

Figura 4: Vetorização dos textos utilizando TF-IDF.

2.5 Função de Busca e Visualização 3D

A função implementada realiza a busca de artigos com base em uma *query*, utilizando TF-IDF e similaridade de cosseno, além de gerar uma visualização tridimensional interativa dos vetores.

```
[ ]
def search_articles(query, top_n=5):
    query_processed = preprocess_text(query)
    query_vec = vectorizer.transform([query_processed])

    # Similaridade do cosseno
    similarities = cosine_similarity(query_vec, tfidf_matrix).flatten()

    # Índices dos mais semelhantes
    top_indices = similarities.argsort()[::-1][:top_n]

    # Cálculo dos ângulos (em graus)
    angles = np.degrees(np.arccos(np.clip(similarities[top_indices], -1.0, 1.0)))

    # Montar DataFrame com resultados
    results = pd.DataFrame({
        'TITLE': df.iloc[top_indices]['TITLE'].values,
        'SIMILARITY': similarities[top_indices],
        'ANGLE (°)': angles,
        'ABSTRACT': df.iloc[top_indices]['ABSTRACT'].values
    })



    print("\n🔍 Resultados mais relevantes:\n")
    display(results[['TITLE', 'SIMILARITY', 'ANGLE (°)']])

      # 🔷 Mostrar resumo do artigo mais relevante
    print("\n🧠 Resumo do artigo mais relevante:\n")
    print(f"📘 {results.iloc[0]['TITLE']}\n")
    print(results.iloc[0]['ABSTRACT'])
    print("\n" + "-" * 90)
```

```
    import plotly.graph_objects as go
    from sklearn.decomposition import PCA

    vectors_to_plot = np.vstack([query_vec.toarray(), tfidf_matrix[top_indices].toarray()])

    # Redução PCA
    pca = PCA(n_components=3)
    reduced_vecs = pca.fit_transform(vectors_to_plot)

    # 🔷 Visualização 3D
    fig = plt.figure(figsize=(8, 6))
    ax = fig.add_subplot(111, projection='3d')

    ax.scatter(reduced_vecs[0, 0], reduced_vecs[0, 1], reduced_vecs[0, 2],
               color='red', s=100, label='Query')

    ax.scatter(reduced_vecs[1:, 0], reduced_vecs[1:, 1], reduced_vecs[1:, 2],
               color='royalblue', s=60, label='Top Artigos')

    for i, title in enumerate(results['TITLE']):
        ax.plot([reduced_vecs[0, 0], reduced_vecs[i + 1, 0]],
                [reduced_vecs[0, 1], reduced_vecs[i + 1, 1]],
                [reduced_vecs[0, 2], reduced_vecs[i + 1, 2]],
                color='gray', linestyle='--', alpha=0.5)
        ax.text(reduced_vecs[i + 1, 0], reduced_vecs[i + 1, 1],
                reduced_vecs[i + 1, 2], f'{i + 1}', fontsize=9)

    ax.set_title("Visualização 3D: Query e Artigos Similares", pad=20)
    ax.legend()
    ax.view_init(elev=20, azim=120)
    plt.tight_layout()
    plt.show()

    return results
```

Figura 5: Função principal de busca e geração de visualização 3D.

2.6 Execução de Consultas Interativas

Por fim, foi implementado um *loop* que permite ao usuário realizar consultas diretamente no terminal. O programa é encerrado quando o usuário digita "sair", "exit"ou "quit".

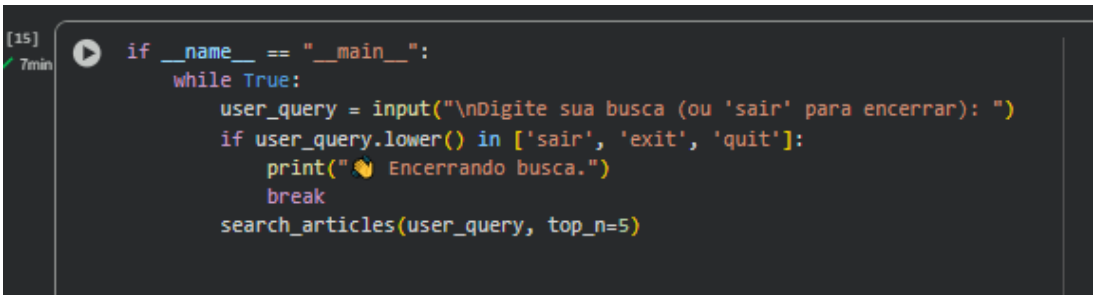
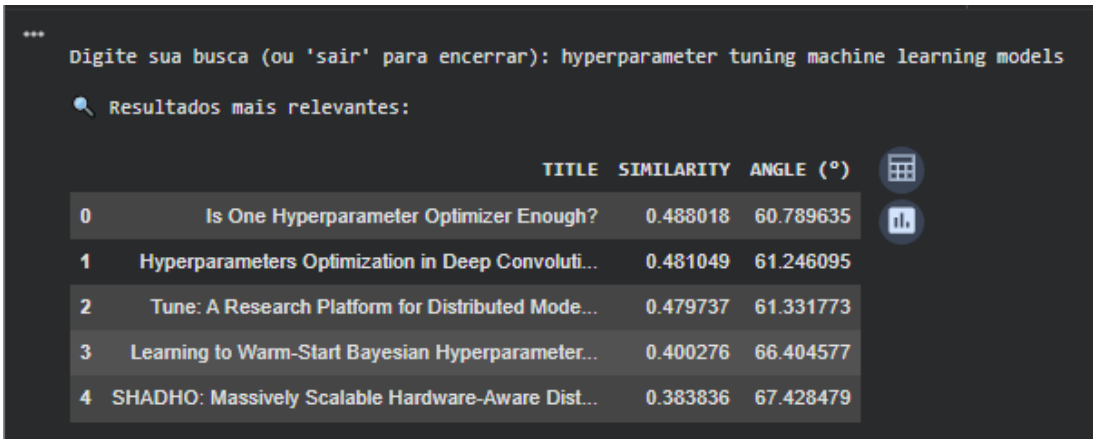


Figura 6: Loop interativo para consultas.

3 RESULTADOS

A seguir, são apresentados exemplos de consultas realizadas no sistema e suas respectivas saídas.



	TITLE	SIMILARITY	ANGLE (°)
0	Is One Hyperparameter Optimizer Enough?	0.488018	60.789635
1	Hyperparameters Optimization in Deep Convoluti...	0.481049	61.246095
2	Tune: A Research Platform for Distributed Mode...	0.479737	61.331773
3	Learning to Warm-Start Bayesian Hyperparameter...	0.400276	66.404577
4	SHADHO: Massively Scalable Hardware-Aware Dist...	0.383836	67.428479

Figura 7: Resultados da busca 1 (consulta: *hyperparameter tuning machine learning models*).

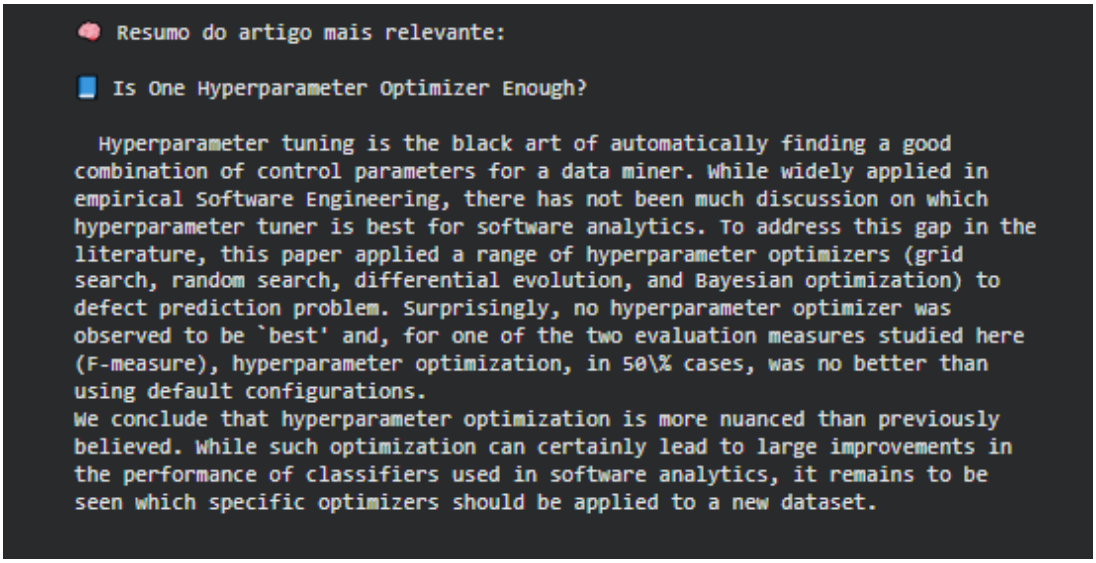


Figura 8: Resumo do artigo mais relevante.

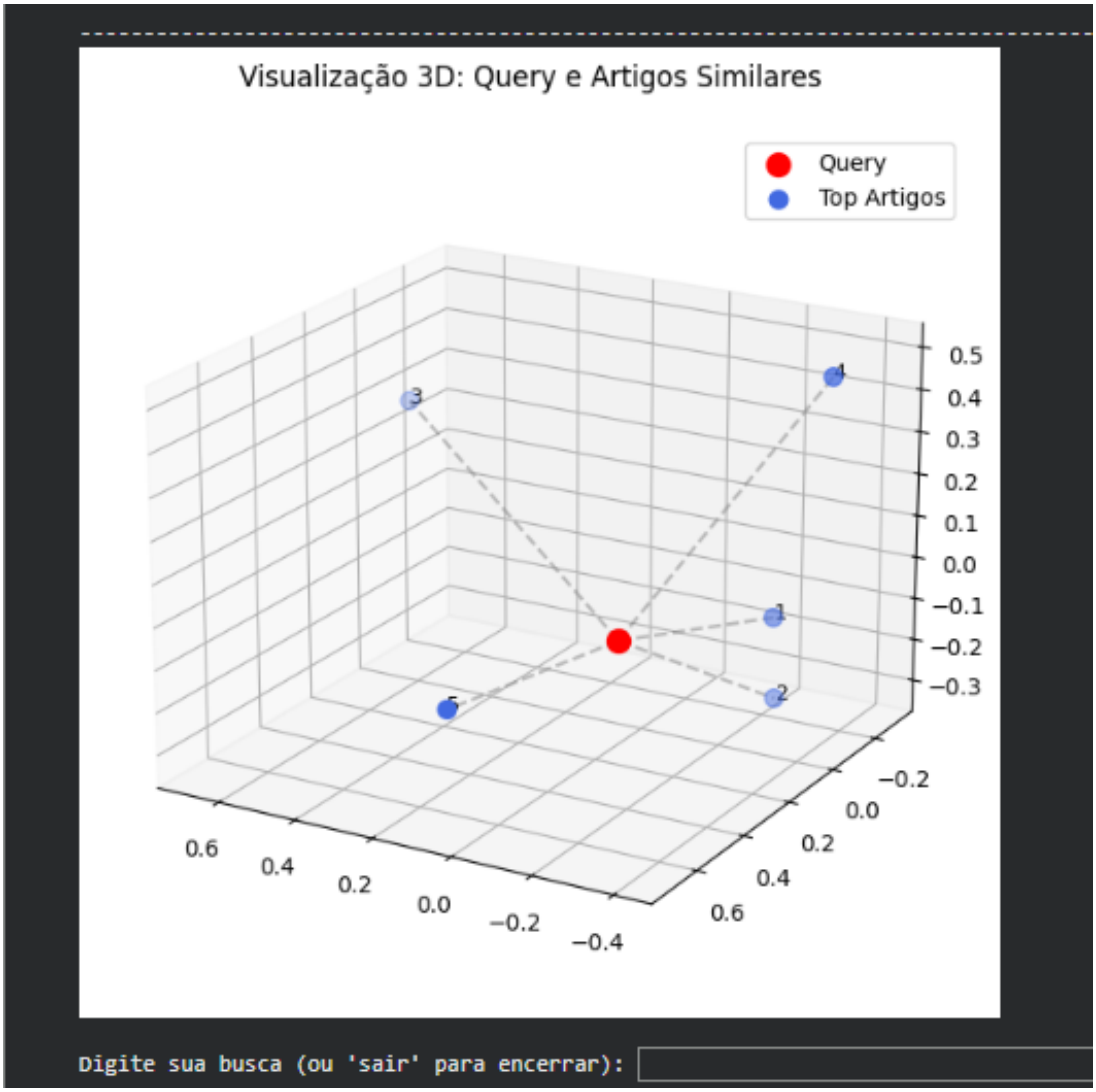


Figura 9: Visualização 3D dos vetores da consulta e artigos mais similares.

Observa-se que os resultados retornaram artigos relacionados à otimização de hiperparâmetros, apresentando valores de similaridade entre 0.38 e 0.48. O resumo exibido demonstra coerência temática com a consulta, reforçando a eficácia do modelo em identificar artigos semanticamente relevantes.

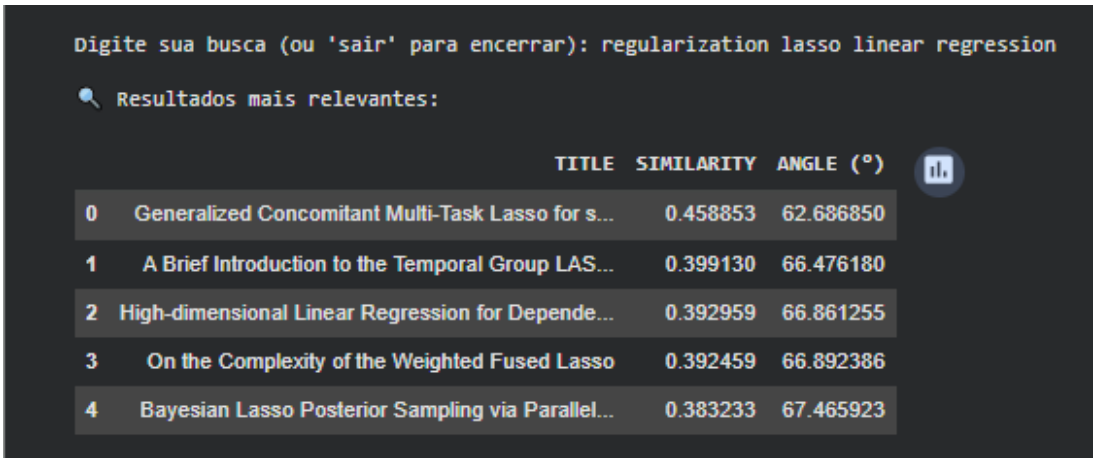


Figura 10: Resultados da busca 2 (consulta: *regularization lasso linear regression*).

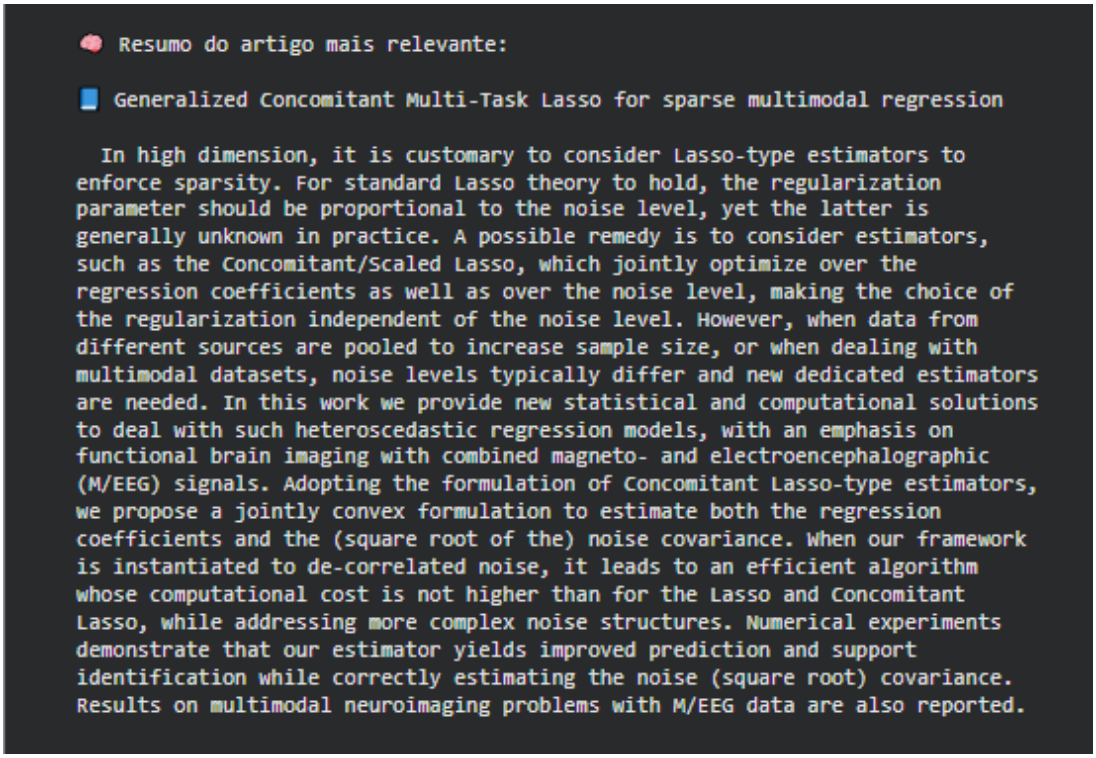


Figura 11: Resumo do artigo mais relevante.

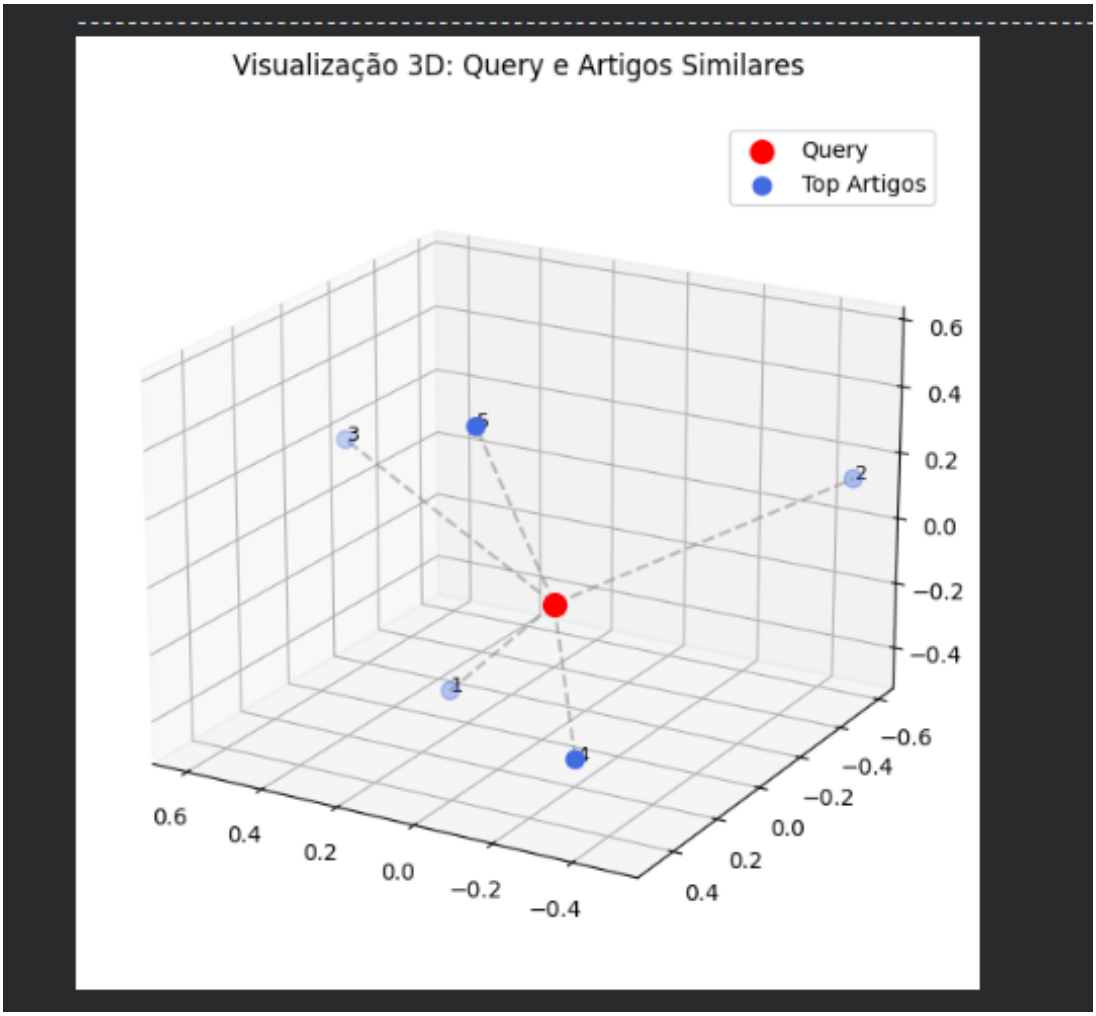


Figura 12: Visualização 3D dos vetores da consulta e artigos mais similares.

Na segunda busca, o sistema apresentou resultados consistentes com o tema proposto, retornando artigos sobre regularização e métodos *Lasso* em regressões lineares. As visualizações tridimensionais mostraram a proximidade espacial entre os vetores, indicando que o *PCA* manteve uma boa separabilidade sem perda significativa de coerência semântica.

CONCLUSÃO

O sistema desenvolvido demonstrou a aplicabilidade do TF-IDF na representação numérica de textos e a eficiência da similaridade de cosseno na identificação de artigos semanticamente relacionados. A combinação dessas técnicas, aliada à visualização tridimensional, proporcionou uma análise intuitiva e fundamentada das relações entre documentos científicos, evidenciando o potencial das abordagens de Processamento de Linguagem Natural para a organização e recuperação de informações em bases textuais extensas.