Universidade do Minho
Escola de Letras, Artes e Ciências Humanas

Análise e Visualização de Dados

Projeto Integrado II

Trabalho Final

Mestrado em Humanidades Digitais

Docentes:

José João Almeida

Álvaro Iriarte Sanromán

Aluno:

Rui Barros PG43616

Data:

30 de maio de 2023

Resumo

Este relatório de projeto resulta no foco de uma análise de quatro obras de Camilo Castelo Branco, fazendo uso da linguagem de programação *Python*. As obras de Camilo analisadas são: "A Infanta Capelista", "A viúva do enforcado", "O carrasco de Vitor Hugo" e "Vulcões de lama".

O objetivo consistia em analisar e apresentar os dados extraídos, que incluem pessoas, lugares, lemas e palavras-chave. É também realizada uma análise de sentimento de cada uma das quatro obras. Os resultados são depois apresentados em listas com o top 10 para comparação, transformados em gráficos. Foi feita uma exportação para ficheiros *.csv* desses mesmos top 10 e foram também gerados *pie charts* para apresentar a análise do sentimento de cada uma das obras.

Este projeto permite-nos testar a qualidade das ferramentas de processamento de linguagem natural disponíveis na atualidade, que além de nos facilitar e acelerar o processo de análise da obra literária de Camilo, também nos permite assistir ao potencial que estas ferramentas demonstram, tanto para sumarizar, extrair e automatizar estas tarefas, especialmente nas Humanidades Digitais.

Introdução

Camilo Castelo Branco é um dos escritores mais conceituados da literatura portuguesa. A sua vida atribulada inspirou a sua vasta obra literária, sendo reconhecido pela sua escrita passional. Como este teve uma vida bastante trágica, isto refletiu-se na sua escrita, afetando-a à medida que a sua saúde se agravava.

Esta análise foca-se em quatro obras de Camilo: "A Infanta Capelista", "A viúva do enforcado", "O carrasco de Vitor Hugo" e "Vulcões de lama". Utilizando ferramentas de processamento de linguagem natural, foram extraídos e apresentados vários tipos de informação de cada obra. Assim, temos uma melhor visão destas obras de Camilo.

Este projeto foi desenvolvido no âmbito do 1º ano do Mestrado em Humanidades Digitais, nas Unidades Curriculares de Análise e Visualização de Dados e Projeto Integrado II.

Metodologia

Como referido acima, esta tarefa foi realizada com *Python*. Para tal, foi necessário importar algumas bibliotecas e módulos:

```
import spacy
from collections import Counter
from nltk.corpus import stopwords
import string
import re
import nltk
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
import matplotlib.pyplot as plt
from pathlib import Path
import pandas as pd

#Portuguese language model
nlp = spacy.load("pt_core_news_sm")

#List of Portuguese stopwords
stop_words = set(stopwords.words("portuguese"))

#Output directory for generated charts and CSV files
base_output_dir = Path("C:/Users/ruypa/OneDrive/Ambiente de Trabalho/MHD/2o
Semestre/Análise e Visualização de Dados/Rui/extractions")

def read_file(filename):
    with open(filename, "r", encoding="utf-8") as file:
        text = file.read()
    return text
```

Criei também a função de análise e de leitura dos ficheiros desejados, que estão definidos no final do programa:

```
analyze_file("C:/Users/ruypa/OneDrive/Ambiente de Trabalho/MHD/2o
Semestre/Análise e Visualização de Dados/Rui/Obra/Camilo-
A_Infanta_Capelista.txt")
```

```
analyze_file("C:/Users/ruypa/OneDrive/Ambiente de Trabalho/MHD/2o
Semestre/Análise e Visualização de Dados/Rui/Obra/Camilo-
A_viuva_do_enforcado.txt")
analyze_file("C:/Users/ruypa/OneDrive/Ambiente de Trabalho/MHD/2o
Semestre/Análise e Visualização de Dados/Rui/Obra/Camilo-
O_carrasco_de_Vitor_Hugo.txt")
analyze_file("C:/Users/ruypa/OneDrive/Ambiente de Trabalho/MHD/2o
Semestre/Análise e Visualização de Dados/Rui/Obra/Camilo-Vulcoes_de_lama.txt")
```

Como o programa foi criado localmente, o *path* terá de ser alterado manualmente em caso de análise e teste do mesmo num computador ou utilizador diferente.

Foi efetuada também uma limpeza dos dados, para remover caracteres indesejados:

```
def clean_text(text):
    #Removes newlines and extra whitespaces
    text = re.sub("\s+", " ", text)
    #Removes punctuation
    text = text.translate(str.maketrans("", "", string.punctuation))
    #Converts to lowercase
    text = text.lower()
    #Removes stopwords
    text = " ".join([word for word in text.split() if word not in stop_words])
    return text
```

1. Extração dos Dados:

1.1. Extração de Entidades:

1.1.1. Extração de Pessoas:

```
def extract_people(doc):
    return [entity.text for entity in doc.ents if entity.label_ == "PER"]
```

Para extrair todas as pessoas mencionadas na obra, foi criada a função *"extract_people"*, que extrai todas as entidades do tipo "PER" (person) dos documentos pretendidos, que fizerem parte da biblioteca portuguesa do *spaCy*. Se estas estiverem etiquetadas como pessoas, são anexadas a uma lista e por fim devolvidas quando o programa for executado:

1.1.2. Extração de Lugares:

```
def extract_places(doc):
    return [ent.text for ent in doc.ents if ent.label_ == "LOC"]
```

Esta extração é exatamente igual à anterior, mas refere-se aos locais (“LOC”).

1.2. Extração dos Lemas:

```
def extract_lemmas(text):
    tokens = nltk.word_tokenize(text, language='portuguese')
    lemmatizer = nltk.stem.WordNetLemmatizer()
    lemmas = [lemmatizer.lemmatize(token) for token in tokens if
token.isalpha() and token.lower() not in stop_words]
    return lemmas
```

Fazendo uso uma vez mais do *spaCy*, é criada uma função para extrair os lemas, com o nome *“extract_lemmas”*. Por fim, é devolvida uma lista com os lemas, que mais tarde tem os dados transportados para um gráfico.

1.3. Extração de Palavras-Chave:

Esta função filtra as *stops words* e carateres não alfabéticos das obras, deixando os *tokens* restantes. Estas palavras-chaves são bastante úteis para uma futura análise ou resumo das obras.

```
def extract_keywords(doc):
    keywords = []
    for token in doc:
        if token.is_alpha and not token.is_stop and token.pos_ in ["NOUN",
"ADJ"]:
            keywords.append(token.text)
            for child in token.children:
                if child.pos_ == "ADJ":
                    keywords.append(child.text)
    return keywords
```

1.4. Análise de Sentimento:

Era também requerida uma análise de sentimento de cada uma das obras analisadas. Para tal, serviu-se da biblioteca *“vaderSentiment”*. Esta função assume a obra como *input*, e testa a polaridade da mesma, de forma a extrair o sentimento geral de cada uma das obras, que é devolvido no final, ao correr o programa.

```
def sentiment_analysis(text):
    analyzer = SentimentIntensityAnalyzer()
    sentiment = analyzer.polarity_scores(text)
    return sentiment
```

1.5. Dados pré-limpeza e estruturação:

De seguida procedemos à criação de uma função que permitisse analisar as obras pretendidas. Utilizando o spaCy, esta faz o programa ler os ficheiros desejados, procedendo depois à extração das várias entidades desejadas.

```
def analyze_file(filename):
    text = read_file(filename)
    doc = nlp(text)

    people = extract_people(doc)
    places = extract_places(doc)
    orgs = extract_orgs(doc)
    lemmas = extract_lemmas(doc)
    mwe = extract_mwe(doc)
    keywords = extract_keywords(doc)
    dates = extract_dates(doc)
    sentiment = sentiment_analysis(text)
```

2. Limpeza:

2.1. Extração de Datas:

Infelizmente não consegui fazer com que esta função resultasse. A ideia assentava em filtrar entidades das obras e extrair apenas as entidades que correspondessem à etiqueta *"DATE"*. Com isto em mente, o objetivo seria extrair essas datas e juntá-las a uma lista.

```
def extract_dates(doc):
    dates = []
    for ent in doc.ents:
        if ent.label_ == "DATE":
            dates.append(ent.text)
    return dates
```

2.2. Tentativas falhadas:

2.2.1. Infelizmente, os dados extraídos apresentaram bastantes erros na filtragem. Foram utilizadas algumas bibliotecas e módulos diferentes, mas nenhuma funcionou com o sucesso pretendido. Entre outras, foram testadas:

2.2.1.1. *Fuzzywuzzy*:

Esta biblioteca foi testada com o intuito de fundir alguns dos nomes presentes no texto, nomes esses que aparecem no próprio top 10 de alguns dos gráficos. Esta possui um algoritmo que calcula a similaridade entre *strings* das obras, fazendo com que haja uma fusão de entidades para evitar repetições.

```
from fuzzywuzzy import fuzz

def extract_people(doc):
    people = []
    merged_people = []

    for ent in doc.ents:
        if ent.label_ == "PER":
            people.append(ent.text)

    for person in people:
        is_similar = False
        for merged_person in merged_people:
            similarity_score = fuzz.ratio(person, merged_person)
            if similarity_score >= 80:  # Adjust the similarity threshold as
needed
                is_similar = True
                break
        if not is_similar:
            merged_people.append(person)

    return merged_people
```

2.2.1.2. *Dateparser:*

Esta biblioteca tem como função extrair informação acerca de datas de um texto. Também foi implementada na tentativa de extrair este tipo de informação, mas sem sucesso:

```
import dateparser

def extract_dates(doc):
```

```
    dates = []
    for ent in doc.ents:
        if ent.label_ == "DATE":
            date = dateparser.parse(ent.text)
            if date is not None:
                dates.append(date)
    return dates
```

2.3. Exibição dos top 10:

A ideia inicial era apresentar o top10 no terminal após correr o programa e fazer com que este apresentasse os gráficos automaticamente. No entanto, decidi alterar um pouco esta estrutura de apresentação. No entanto, mantive apenas a extração dos top 10s, de modo a facilitar a limpeza e análise dos dados.

3. Visualização:

Com a automatização em mente, decidi estabelecer também uma pasta onde o programa guardava automaticamente os gráficos e ficheiros *.csv*.

3.1. Exportação para *.csv* e criação dos gráficos:

Estas funções permitiram a exportação dos dados obtidos para um ficheiro *.csv* e também a criação dos gráficos de barras com os dados. Assim, havia uma maior possibilidade de transporte dos mesmos.

```
def export_to_csv(data, filename):
    df = pd.DataFrame(data, columns=["Extracted Data", "Frequency"])
    df.to_csv(filename, index=False)

def create_bar_plot(data, title, xlabel, ylabel, filename):
    entities = [entity for entity, count in data]
    counts = [count for entity, count in data]

    plt.figure(figsize=(10, 6))
    plt.bar(entities, counts)
    plt.title(title)
    plt.xlabel(xlabel)
```

```
    plt.ylabel(ylabel)
    plt.xticks(rotation=45, ha="right")
    plt.tight_layout()
    plt.savefig(filename)
    plt.close()
```

3.2. Função de análise dos ficheiros:

Criei também esta função para facilitar a leitura e extração dos dados de cada um dos ficheiros desejados.

De seguida, estas dados eram guardados na pasta indicada, a que dei o nome de "extractions", onde o programa guarda automaticamente os ficheiros criados com as extrações.

```
def analyze_file(filename, output_dir):
    text = read_file(filename)
    cleaned_text = clean_text(text)
    doc = nlp(cleaned_text)

    people = extract_people(doc)  #Extracts people
    places = extract_places(doc)  #Extracts places
    lemmas = extract_lemmas(cleaned_text)  #Extracts lemmas
    keywords = extract_keywords(doc)  #Extracts keywords

    #Performs sentiment analysis on the cleaned text
    sentiment = sentiment_analysis(cleaned_text)

    #Counts the frequency of each entity
    people_count = Counter(people).most_common(10)
    places_count = Counter(places).most_common(10)
    lemmas_count = Counter(lemmas).most_common(10)
    keywords_count = Counter(keywords).most_common(10)

    #Creates the output directory for this file
    output_dir = base_output_dir / output_dir
    output_dir.mkdir(parents=True, exist_ok=True)
```

3.3. Exportação e definição dos ficheiros:

Aqui podemos ver o código onde são estabelecidos os nomes e títulos que dei a cada ficheiro. É possível também ver as definições para os *pie charts*:

```
#Exports the entity counts to CSV files
    export_to_csv(people_count, output_dir / "people.csv")
    export_to_csv(places_count, output_dir / "places.csv")
    export_to_csv(lemmas_count, output_dir / "lemmas.csv")
    export_to_csv(keywords_count, output_dir / "keywords.csv")

    #Creates bar plots for entity counts
    create_bar_plot(people_count, f"Top 10 People in
{file_mappings[file]}", "Person", "Count", output_dir / "people.png")
    create_bar_plot(places_count, f"Top 10 Places in
{file_mappings[file]}", "Place", "Count", output_dir / "places.png")
    create_bar_plot(lemmas_count, f"Top 10 Lemmas in
{file_mappings[file]}", "Lemma", "Count", output_dir / "lemmas.png")
    create_bar_plot(keywords_count, f"Top 10 Keywords in
{file_mappings[file]}", "Keyword", "Count", output_dir / "keywords.png")

    #Creates a pie chart for sentiment scores
    labels = ['Negative', 'Neutral', 'Positive']
    sizes = [sentiment['neg'], sentiment['neu'], sentiment['pos']]
    colors = ['#ff9999', '#66b3ff', '#99ff99']
    explode = (0.1, 0, 0)  #explode the 'Negative' slice

    plt.figure(figsize=(6, 6))
    plt.pie(sizes, labels=labels, colors=colors, explode=explode,
autopct='%1.1f%%', startangle=90)
    plt.title(f"Sentiment Distribution in {file_mappings[file]}")
    plt.axis('equal')
    plt.savefig(output_dir / "sentiment_pie.png")
    plt.close()
```

3.4. Análise de sentimento:

De acordo com os gráficos apresentados abaixo, é possível concluir que aproximadamente 98% do texto de cada uma das obras possui linguagem neutra, sendo:

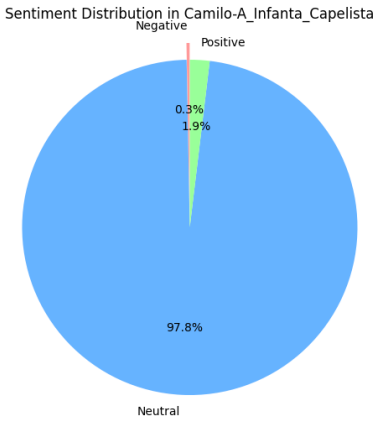

Figura 1 "Camilo-A_Infanta_Capelista"

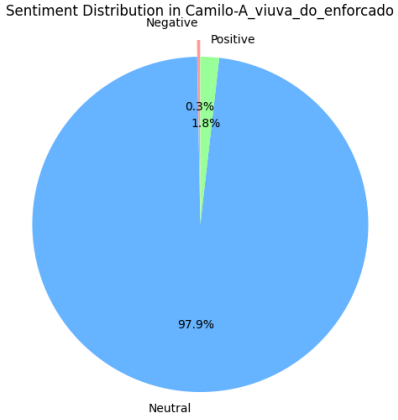

Figura 2 "Camilo- A_viuva_do_enforcado "

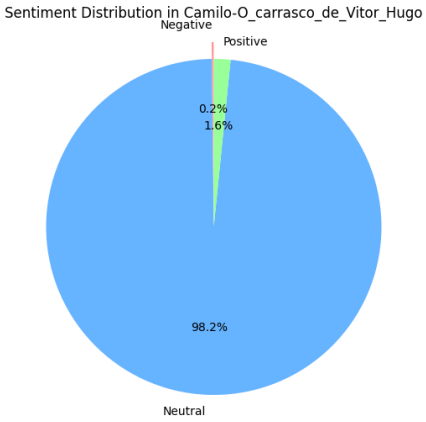

Figura 2 "Camilo-O_carrasco_de_Vitor_Hugo"

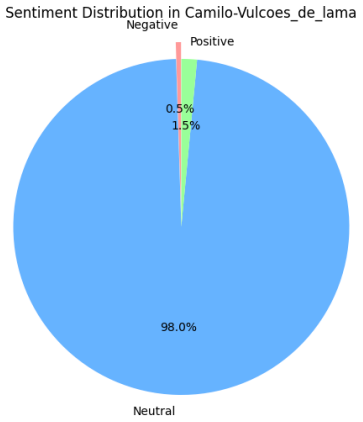

Figura 4 "Camilo-Vulcoes_de_lama"

3.5. Gráficos:

Com o objetivo de tornar a tarefa completamente automatizada em *Python*, apresentarei todos os gráficos extraídos após correr o programa:

Nota: Após uma melhor análise, decidi excluir os gráficos dos lemas, uma vez que estes não produziram os resultados desejados. Estes encontram-se na mesma na pasta das extrações, apenas não os incluí no relatório devido à baixa qualidade da mesma extração.

3.5.1. "Camilo-A_Infanta_Capelista":

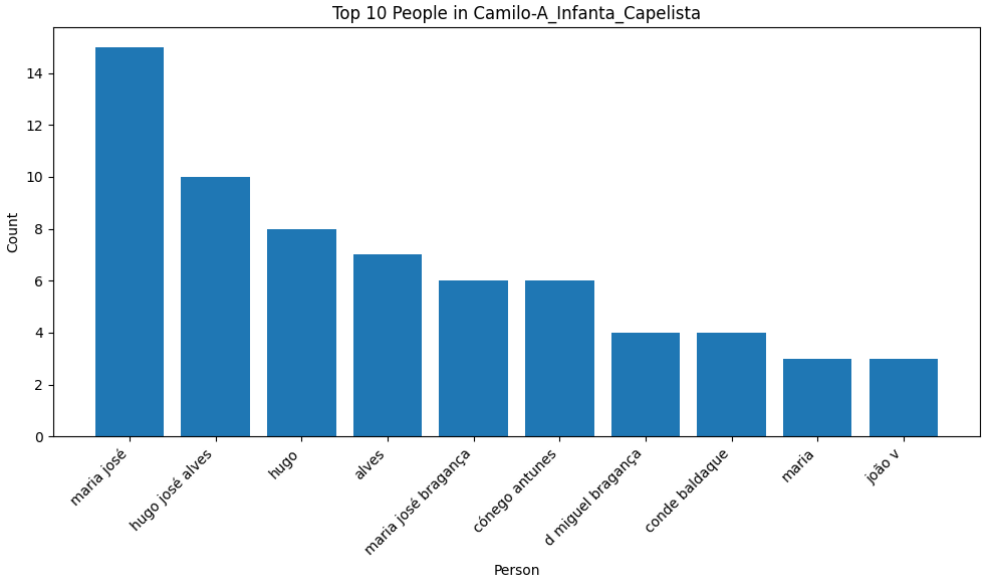

Figura 5 Top10 People

Nesta primeira obra analisada, e tal como nas restantes, é possível observar as personagens, lugares e palavras-chave mais importantes. A entidade com nome "Hugo José Alves" é a mais mencionada nesta obra, como 3 das barras deste top 10 lhe pertencem. Apenas não fui capaz de os juntar numa só.

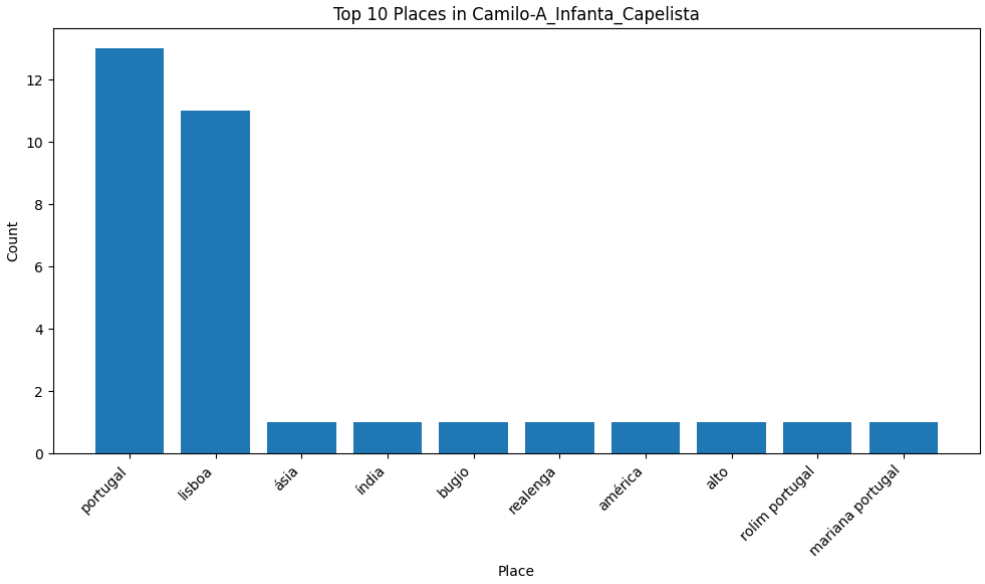

Figura 6 Top10 Places

Aqui destaco para “portugal e “lisboa”, os locais que mais vezes foram mencionados na obra. Estes encontram-se em letra minúscula uma vez que durante a limpeza do texto todos os tokens foram convertidos para minúsculas.

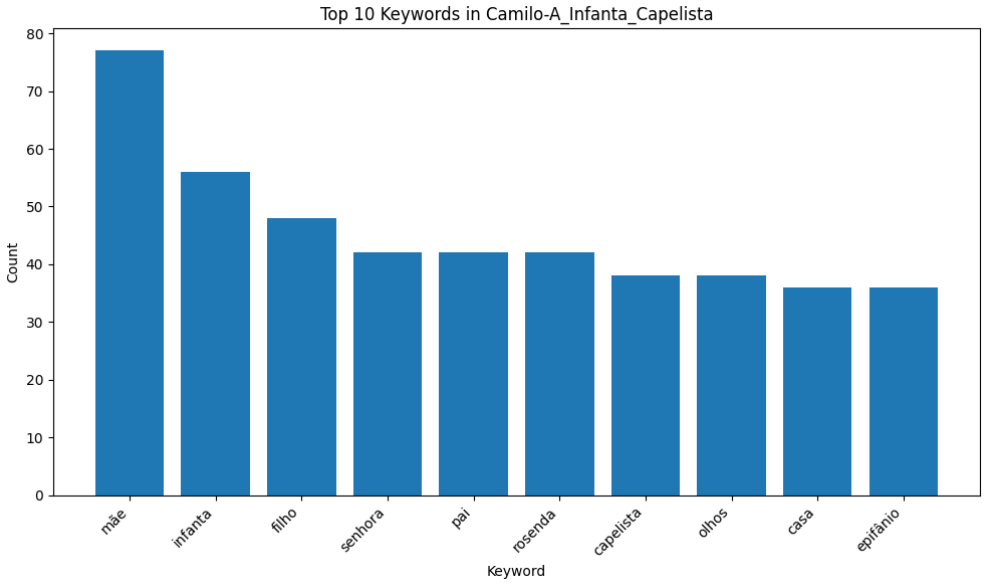

Figura 8 Top10 Keywords

Destaco aqui para a palavra “mãe”, mencionada quase 80 vezes na obra.

3.5.2. “Camilo-A_viuva_do_enforcado”:

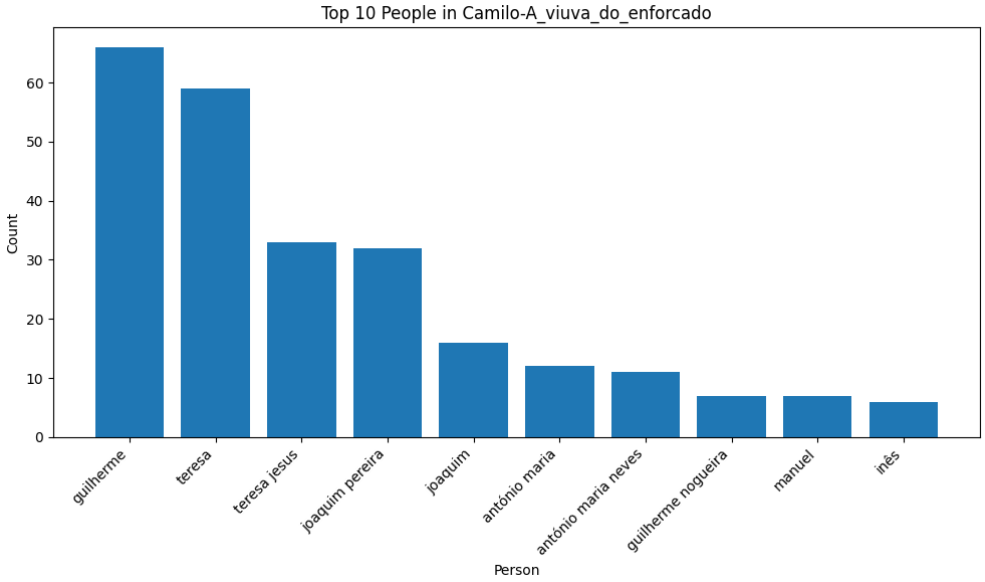

Figura 9 Top10 People

Nesta segunda obra, "Teresa" é a personagem que mais vezes foi mencionada, uma vez que a segunda e a terceira pessoas do gráfico são a mesma. Faltou aqui a fusão dos dois termos, algo que não consegui fazer.

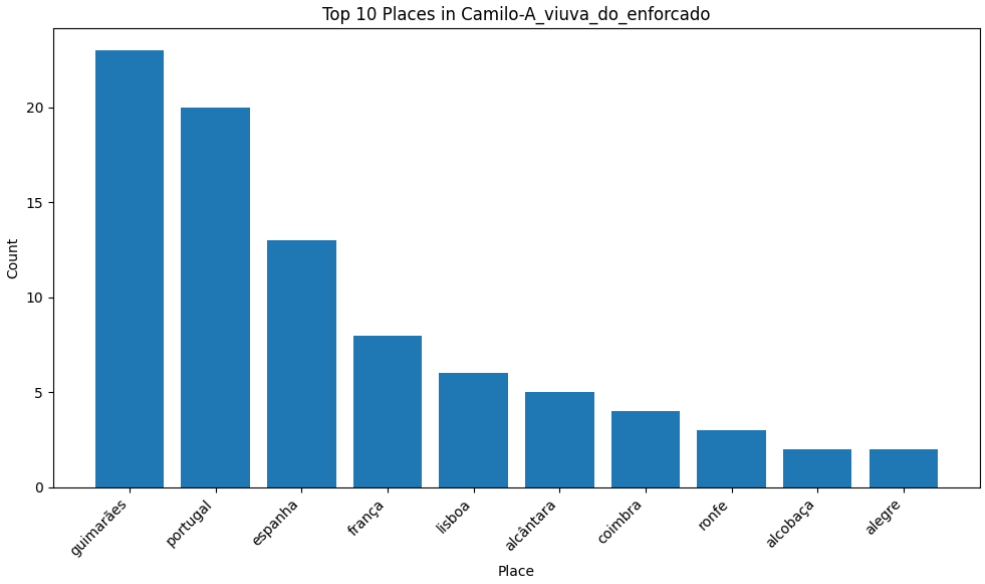

Figura 10 Top10 Places

Destaco aqui três países que Camilo menciona frequentemente ao longo desta obra: Portugal, Espanha e França.

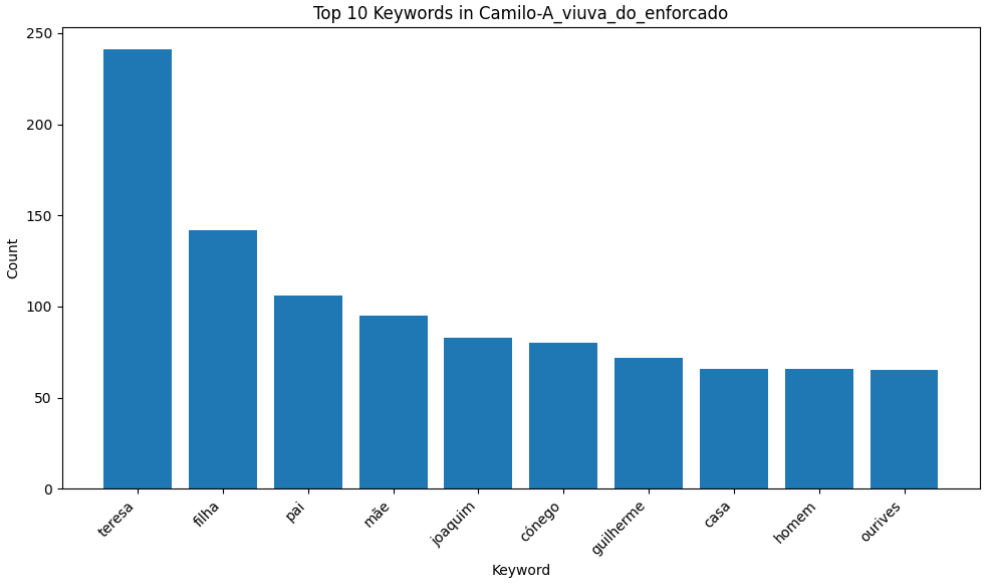

Figura 12 Top10 Keywords

De igual modo, nesta obra, "Teresa" é a palavra-chave que se destaca, sendo mencionada quase mais uma centena de vezes do que a segunda, "filha".

3.5.3. "Camilo-O_carrasco_de_Vitor_Hugo":

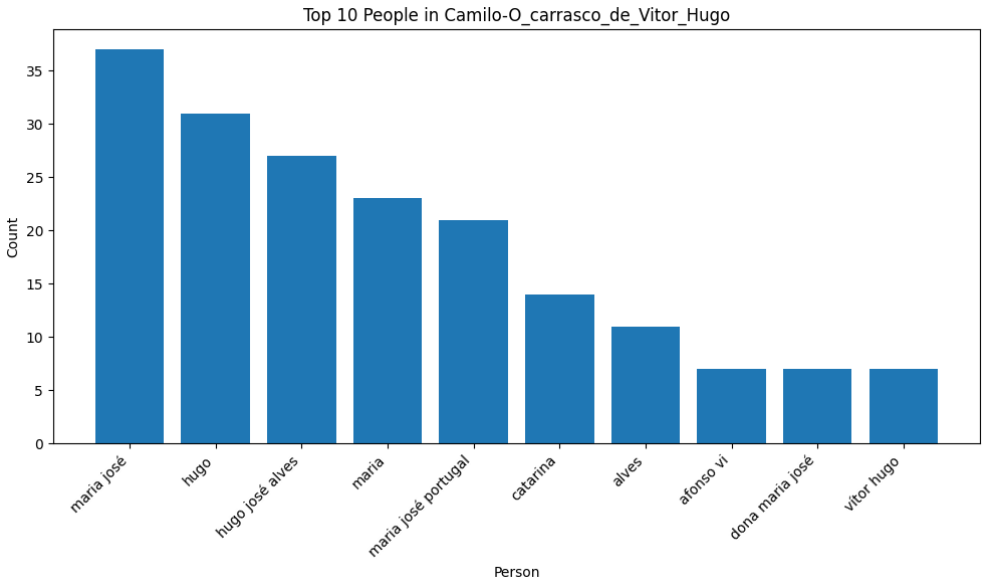

Figura 9-Top10 People

Uma vez mais temos "Hugo José Alves" como a pessoa mais mencionada pelo autor, desta vez na terceira obra analisada.

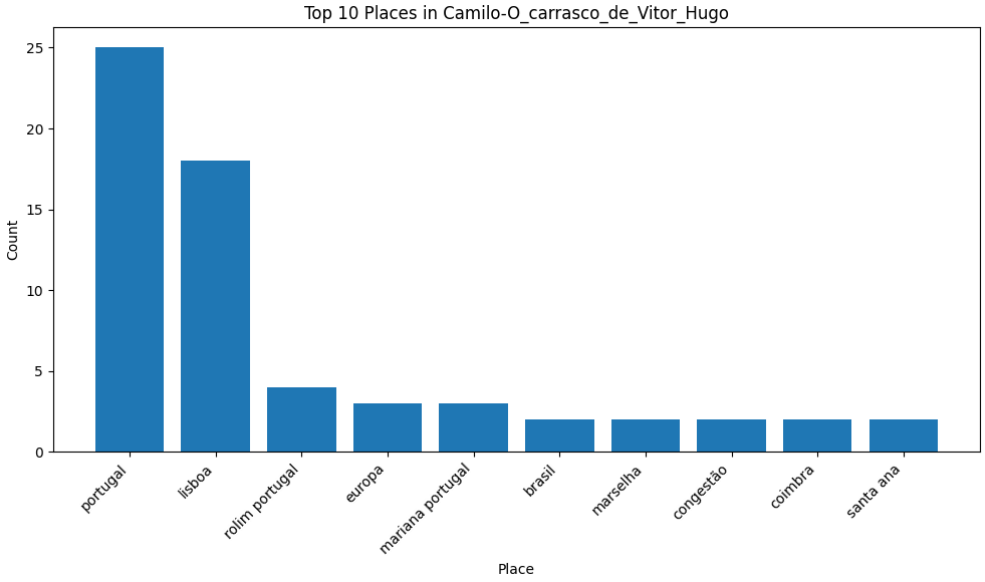

Figura 10-Top10 Places

Portugal e Lisboa uma vez mais como os locais que se destacam significativamente dos restantes deste top 10.

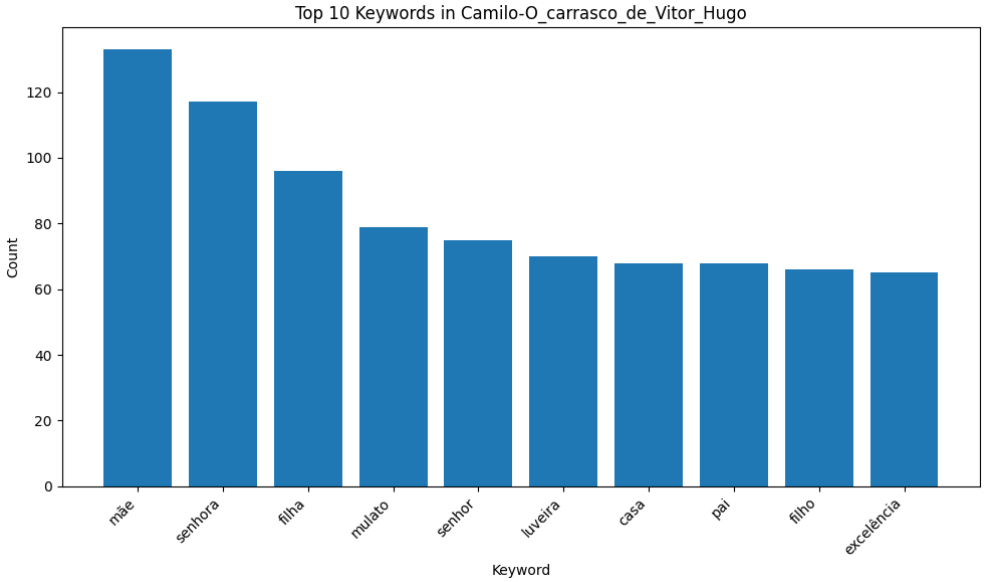

Figura 18-Top10 Keywords

Neste caso, as palavras-chave desta obra permitem-nos ter uma idea geral do que que esta apresenta, uma vez que inclui bastantes termos úteis para uma análise geral da obra.

3.5.4. "Camilo-Vulcoes_de_lama":

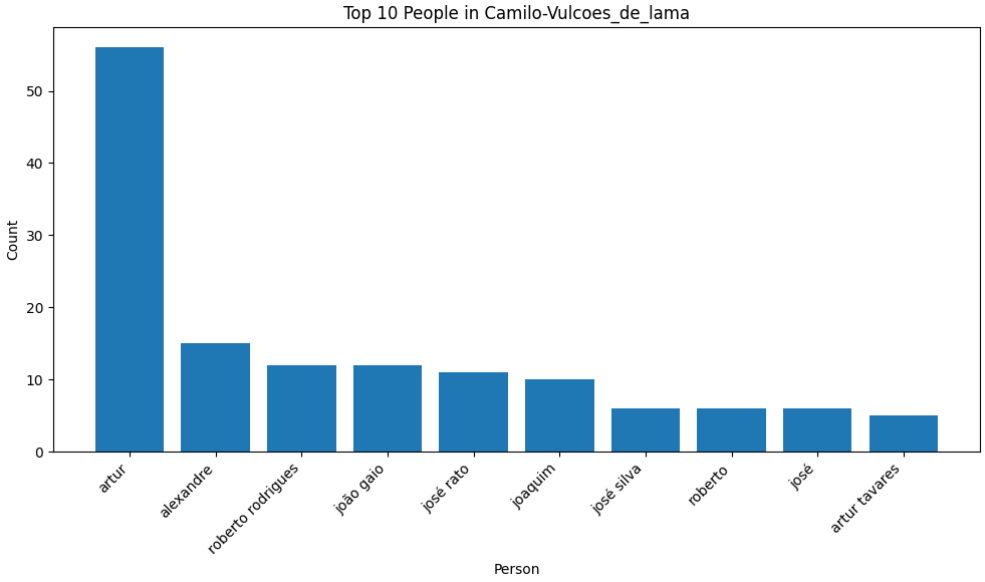

Figura 19-Top10 People

Aqui podemos ver que a personagem “Artur” é mencionada quase quatro vezes mais do que a segunda, “Alexandre”, sendo esta e as restantes 8 são mencionadas pouco mais de uma dezena de vezes ao longo da obra, enquanto a primeira ultrapassa as 50 menções.

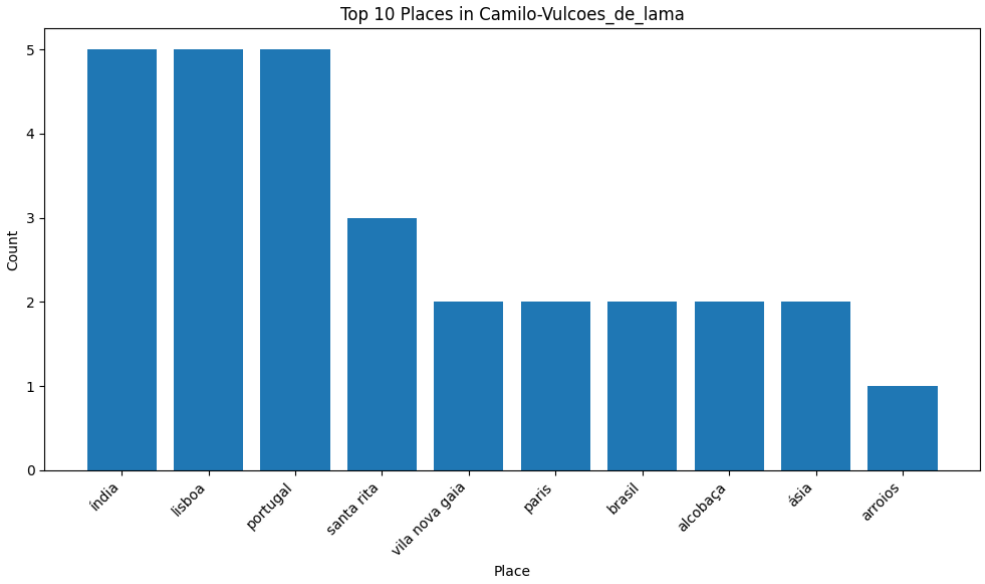

Figura 20-Top10 Places

Nesta extração dos locais, podemos reparar que Camilo menciona três países, duas capitais e também um continente ao longo desta obra.

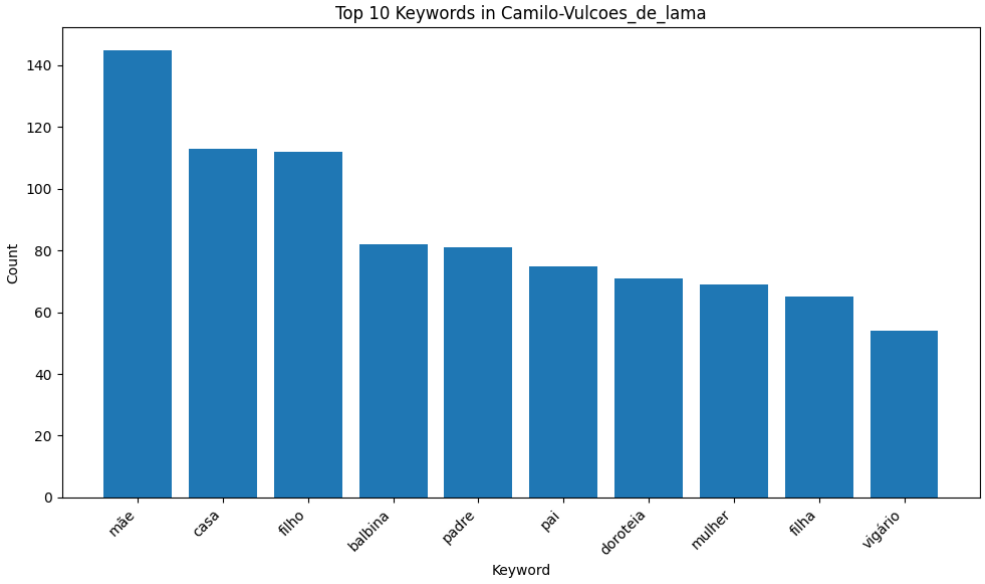

Figura 24-Top10 Keywords

“Mãe”, “casa” e “filho” são as palavras-chave que mais sobressaem ao olhar para este gráfico.

Conclusão

Em suma, esta tarefa permitiu analisar com maior precisão algumas das obras de Camilo Castelo Branco, servindo-se de várias ferramentas para tal. Ao utilizar estes modelos e bibliotecas com *Python*, conseguimos ter uma visão mais aprofundada da escrita de Camilo.

Estas ferramentas demonstram ainda a forte capacidade de análise e extração destas ferramentas de processamento de linguagem natural. Permitem-nos poupar imenso tempo e, desde que programadas corretamente, permitem extrair dados que nos levariam dias a obter de outra forma.

Incorporadas com boas técnicas de visualização de dados, podemos assim analisar grandes quantidades de texto num curto espaço de tempo. Neste caso, e nesta Unidade Curricular, as obras escritas pelo brilhante Camilo Castelo Branco.

Referências

Frazão, D. (2023, April 25). Biografia de Camilo castelo branco. eBiografia. https://www.ebiografia.com/camilo_castelo_branco/

NLTK. (n.d.). https://www.nltk.org/index.html

OpenAI. (2023). ChatGPT (Version GPT-3.5) [AI model]. Retrieved from https://openai.com/blog/chatgpt

Simplified text processing. TextBlob. (n.d.). https://textblob.readthedocs.io/en/dev/#

Spacy · industrial-strength natural language processing in python. · Industrial-strength Natural Language Processing in Python. (n.d.). https://spacy.io/

Visualization with python. Matplotlib. (n.d.). https://matplotlib.org/

Link: https://github.com/ruipabarros/AVD2023_RuiBarros