Projeto de Disciplina de Processamento de Linguagem Natural com Python

Bem-vindo ao projeto de disciplina de **Processamento de Linguagem Natural com Python**. Ao longo das últimas aulas vimos uma série de aplicações que nos deram a amplitude de possibilidades em trabalhar com textos. Para tal, usamos diversas bibliotecas, onde as que mais se destacaram foram NLTK, SPACY e GENSIM.

Esse notebook servirá de guia para a execução de uma análise de tópicos completa, usando o algoritmo de LDA e recursos para interpretação dos resultados. Utilizaremos notícias da seção "Mercado" extraídas da Folha de S. Paulo no ano de 2016. Complete a análise com os códigos que achar pertinente e responda as questões presentes no Moodle. Boa sorte!

O Notebook

Nesse notebook, você será guiado pela análise de **Extração de Tópicos**. As seguintes tarefas serão realizadas

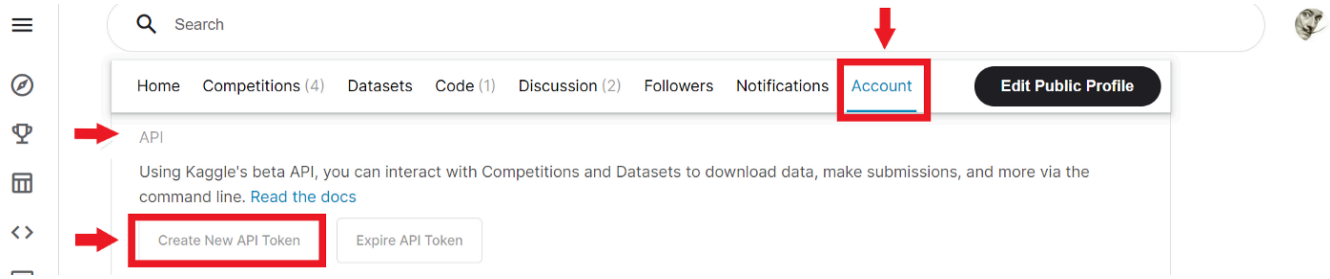
1. Download dos dados provenientes do kaggle
2. Seleção dos dados relevantes para a nossa análise
3. Instalação das principais ferramentas e importação de módulos
4. Pré-processamento usando NLTK
5. Pré-processamento usando Spacy
6. Análise de tópicos usando LDA
7. Análise de NER usando Spacy
8. Visualização dos tópicos usando tokens e entidades.

```
Comece a programar ou gere código com IA.
```

Instruções para baixar os dados

Para baixar os dados será necessário o uso do gerenciador de downloads da Kaggle. A Kaggle, uma subsidiária do grupo Alphabet (Google), é uma comunidade on-line de cientistas de dados e profissionais de aprendizado de máquina.

Para utilizar o gerenciador, será necessário criar uma conta no site Kaggle.com. Com a conta criada, obtenha um token de acesso, no formato kaggle.json



Em posse do token (baixe para seu computador), execute a células da próxima seção para acessar os dados de interesse e baixá-los.

Baixe os dados

Instale o gerenciador kaggle no ambiente do Colab e faça o upload do arquivo kaggle.json

```
# !pip install -q kaggle
# !rm -rf kaggle.json
# from google.colab import files

# files.upload()

%pip install --upgrade pip
%pip install pandas
%pip install tqdm
%pip install -U spacy
%pip install nltk
%pip install pyldavis &> /dev/null
%pip install numpy
%pip install wordcloud seaborn scikit-learn
```

Mostrar saída oculta

Crie a pasta .kaggle

```
!rm -rf .kaggle
!mkdir .kaggle
!cp kaggle.json .kaggle/
!chmod 600 .kaggle/kaggle.json
```

Baixe o dataset

```
import os
os.environ['KAGGLE_CONFIG_DIR'] = os.getcwd() + '/.kaggle'

!kaggle datasets download --force -d marlesson/news-of-the-site-folhauol
```

```
Dataset URL: https://www.kaggle.com/datasets/marlesson/news-of-the-site-folhauol
License(s): CC0-1.0
Downloading news-of-the-site-folhauol.zip to /Users/herbertins/Workspace/machine_learning/nlp-project
  0%|                                               | 0.00/187M [00:00<?, ?B/s]
100%|████████████████████████████████████████| 187M/187M [00:00<00:00, 2.46GB/s]
```

Criar o DataFrame com os dados lidos diretamente da plataforma Kaggle

```
import pandas as pd
from tqdm.auto import tqdm
tqdm.pandas()

df = pd.read_csv("news-of-the-site-folhauol.zip")
```

```
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages/tqdm/auto.py:21: TqdmWarning: IProgress
  from .autonotebook import tqdm as notebook_tqdm
```

Atualizar o SPACY e instalar os modelos pt_core_news_lg

```
!python3 -m spacy download pt_core_news_lg

import spacy
from spacy.lang.pt.stop_words import STOP_WORDS
```

```
Collecting pt-core-news-lg==3.8.0
  Downloading https://github.com/explosion/spacy-models/releases/download/pt_core_news_lg-3.8.0/pt_core_news_lg-3.8.0-py
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 568.2/568.2 MB 20.0 MB/s eta 0:00:0000:0100:01
✔ Download and installation successful
You can now load the package via spacy.load('pt_core_news_lg')
```

Instalar os datasets `stopwords`, `punkt` e `rslp` do nltk

```
import nltk
nltk.download("stopwords")
nltk.download("punkt")
nltk.download("punkt_tab")
nltk.download("rslp")
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/herbertins/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     /Users/herbertins/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package punkt_tab to
[nltk_data]     /Users/herbertins/nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
[nltk_data] Downloading package rslp to /Users/herbertins/nltk_data...
[nltk_data]   Package rslp is already up-to-date!
True
```

Carregar os módulos usados ao longo desse notebook

```
import warnings
warnings.filterwarnings('ignore')
```

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import LatentDirichletAllocation as LDA
import numpy as np

from wordcloud import WordCloud

import seaborn as sns
from itertools import chain

from typing import List, Set, Any


SEED = 123
```

Filtrando os dados para utilizar apenas as notícias do ano de 2016 e da categoria "Mercado"

Filtre os dados do DataFrame df e crie um DataFrame news_2016 que contenha apenas notícias de **2016** e da categoria **mercado**.

```
df['date'] = pd.to_datetime(df.date)
news_2016 = df[(df["date"].dt.year == 2016) & (df["category"].str.lower() == "mercado")].copy()
```

NLTK Tokenizer and Stemmer

Crie uma coluna no dataframe `news_2016` contendo os tokens para cada um dos textos. Os tokens devem estar representados pelo radical das palavras (stem). Para tal, complete o conteúdo da função `tokenize`.

```
from nltk.tokenize import word_tokenize
from nltk.stem import RSLPStemmer

def tokenize(text: str) -> List:
  stemmer = RSLPStemmer()
  tokens = word_tokenize(text.lower())
  stems = [stemmer.stem(token) for token in tokens if token.isalpha()]
  return stems

news_2016.loc[:, 'nltk_tokens'] = news_2016.text.progress_map(tokenize)
```

```
100%|██████████| 7943/7943 [00:55<00:00, 142.05it/s]
```

Criar uma documento SPACY para cada texto do dataset

Crie uma coluna `spacy_doc` que contenha os objetos spacy para cada texto do dataset de interesse. Para tal, carregue os modelos `pt_core_news_lg` e aplique em todos os textos (pode demorar alguns minutos...)

```
nlp = spacy.load("pt_core_news_lg")
news_2016['spacy_doc'] = list(nlp.pipe(news_2016['text']))
```

Realize a Lematização usando SPACY

O modelo NLP do spacy oferece a possiblidade de lematizar textos em português (o que não acontece com a biblioteca NLTK). Iremos criar uma lista de tokens lematizados para cada texto do nosso dataset. Para tal, iremos retirar as stopwords, usando uma função que junta stopwords provenientes do NLTK e do Spacy. Essa lista completa, é retornada pela função stopwords (e você não precisa mexer).

Já a função filter retorna True caso o token seja composto por caracters alfabéticos, não estiver dentro da lista de stopwords e o lemma resultante não estiver contido na lista `o", "em", "em o", "em a" e "ano"`.

Crie uma coluna chamada `spacy_lemma` para armazenar o resultado desse pré-processamento.

```
def stopwords() -> Set:
    return set(list(nltk.corpus.stopwords.words("portuguese")) + list(STOP_WORDS))

complete_stopwords = stopwords()

def filter(w: spacy.lang.pt.Portuguese) -> bool:
  return w.is_alpha and w.text.lower() not in complete_stopwords and w.lemma_ not in ['o', 'em', 'em o', 'em a', 'ano']
```

```
def lemma(doc: spacy.lang.pt.Portuguese) -> List[str]:
  lemmas = [w.lemma_ for w in doc if filter(w)]
  return lemmas

news_2016.loc[:, 'spacy_lemma'] = news_2016.spacy_doc.progress_map(lemma)
```

```
100%|██████████| 7943/7943 [00:02<00:00, 3351.71it/s]
```

Reconhecimento de entidades nomeadas

Crie uma coluna `spacy_ner` que armazene todas as organizações (APENAS organizações) que estão contidas no texto.

```
def NER(doc: spacy.lang.pt.Portuguese):
  organizations = [ent.text for ent in doc.ents if ent.label_ == 'ORG']
  return organizations

news_2016.loc[:, 'spacy_ner'] = news_2016.spacy_doc.progress_map(NER)
```

```
100%|██████████| 7943/7943 [00:00<00:00, 37839.87it/s]
```

Bag-of-Words

Crie uma coluna `tfidf` no dataframe `news_2016`. Use a coluna `spacy_lemma` como base para cálculo do TFIDF. O número máximo de features que iremos considerar é 5000. E o token, tem que ter aparecido pelo menos 10 vezes (`min_df`) nos documentos.

```
class Vectorizer:
  def __init__(self, doc_tokens: List):
    self.doc_tokens = doc_tokens
    self.tfidf = None


  def vectorizer(self):
      docs = [" ".join([t for t in tokens if len(t) >= 3]) for tokens in self.doc_tokens]

      self.tfidf = TfidfVectorizer(
          lowercase=True,
          stop_words=list(complete_stopwords),
          max_features=5000,
          min_df=10,
          tokenizer=lambda x: x.split(),
          preprocessor=lambda x: x,
          ngram_range=(1, 2)
      ).fit(docs)

      return self.tfidf

  def __call__(self):
    if self.tfidf is None:
        self.vectorizer()
    return self.tfidf

doc_tokens = news_2016.spacy_lemma.values.tolist()
vectorizer = Vectorizer(doc_tokens)

def tokens2tfidf(tokens):
    tokens = ' '.join(tokens)
    array =  vectorizer().transform([tokens]).toarray()[0]
    return array


news_2016.loc[:, 'tfidf'] = news_2016.spacy_lemma.progress_map(tokens2tfidf)
```

```
100%|██████████| 7943/7943 [00:05<00:00, 1442.07it/s]
```

Extração de Tópicos

Realize a extração de 9 tópicos usando a implementação do sklearn do algoritmo Latent Dirichlet Allocation. Como parâmetros, você irá usar o número máximo de iterações igual à 100 (pode demorar) e o `random_seed` igual a `SEED` que foi setado no início do notebook

```
N_TOKENS = 9

corpus = np.stack(news_2016['tfidf'].values)
lda = LDA(n_components=N_TOKENS, max_iter=100, random_state=SEED)
lda.fit(corpus)
```

```
LatentDirichletAllocation(max_iter=100, n_components=9, random_state=123)
```

Atribua a cada text, um (e apenas um) tópic.

Crie uma coluna `topic` onde o valor é exatamente o tópico que melhor caracteriza o documento de acordo com o algoritmo de LDA.

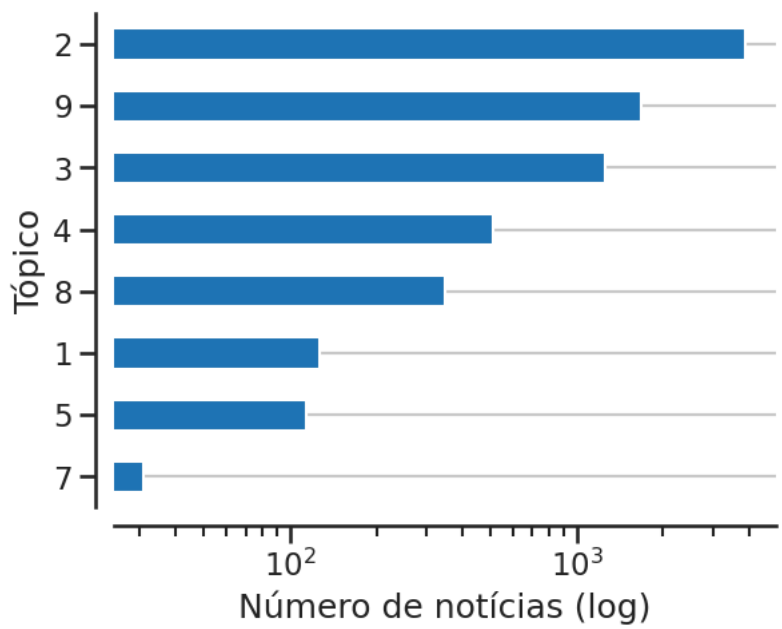
```
def get_topic(tfidf: np.array):
  topic_distribution = lda.transform(tfidf.reshape(1, -1))
  return np.argmax(topic_distribution) + 1

news_2016['topic'] = news_2016.tfidf.progress_map(get_topic)
```

```
100%|██████████| 7943/7943 [00:01<00:00, 4031.63it/s]
```

Número de documentos vs tópicos

Esse gráfico nos mostra quantos documentos foram caracterizados por cada tópico.

```
with sns.axes_style("ticks"):
    sns.set_context("talk")
    ax = news_2016['topic'].value_counts().sort_values().plot(kind = 'barh')
    ax.yaxis.grid(True)
    ax.set_ylabel("Tópico")
    ax.set_xlabel("Número de notícias (log)")
    sns.despine(offset = 10)
    ax.set_xscale("log")
```



Crie uma nuvem de palavra para cada tópico.

Use as colunas `spacy_lemma` e `topic` para essa tarefa.

```
import matplotlib.pyplot as plt

def plot_wordcloud(text:str, ax:plt.Axes) -> plt.Axes:
  wordcloud = WordCloud(width=800, height=400, background_color='white').generate(text)
  ax.imshow(wordcloud, interpolation='bilinear')
  ax.axis("off")
  return ax

def plot_wordcloud_for_a_topic(topic:int, ax:plt.Axes) -> plt.Axes:
```

```
    topic_news = news_2016[news_2016['topic'] == topic]
    list_of_words = chain(*topic_news.spacy_lemma.values.tolist())
    string_complete = ' '.join(list_of_words)
    if not string_complete:
        return None
    return plot_wordcloud(string_complete, ax)

fig, axis = plt.subplots(3, 3, figsize=(16, 12))

axis_  = axis.flatten()
for idx, ax in enumerate(axis_):
    ax_ = plot_wordcloud_for_a_topic(idx+1, ax)
    if ax_ is None:
        plt.delaxes(ax)
        continue
    ax.set_title(f"Tópico {idx + 1}")
fig.tight_layout()

fig.savefig('imagens/nuvens_palavras.png', dpi=300)
```

Crie uma nuvem de entidades para cada tópico.

Use as colunas `spacy_lemma` e `topic` para essa tarefa.

```
def plot_wordcloud_entities_for_a_topic(topic:int, ax:plt.Axes) -> plt.Axes:
    topic_news = news_2016[news_2016['topic'] == topic]
    list_of_docs = topic_news.spacy_ner.apply(lambda l : [w.replace(" ", "_") for w in l])
    list_of_words = chain(*list_of_docs)
    string_complete = ' '.join(list_of_words)
    if not len(string_complete):
        return None
    return plot_wordcloud(string_complete, ax)

fig, axis = plt.subplots(3, 3, figsize=(16, 12))

axis_  = axis.flatten()
for idx, ax in enumerate(axis_):
    ax_ = plot_wordcloud_entities_for_a_topic(idx+1, ax)
    if ax_ is None:
        plt.delaxes(ax)
        continue
    ax.set_title(f"Tópico {idx + 1}")
fig.tight_layout()

fig.savefig('imagens/nuvens_entidades.png', dpi=300)
```