Projeto de conclusão de disciplina – NLP – Bruno Adalberto dos Santos

Implementar técnicas de lematização

1. Qual o endereço do seu notebook (colab) executado? Use o botão de compartilhamento do colab para obter uma url.

   https://github.com/BrunoBersan/analise-dados-pnl-noticias

   analise_topicos.ipynb

   https://github.com/BrunoBersan/analise-dados-pnl-noticias/blob/main/analise_topicos.ipynb

2. Em qual célula está o código que realiza o download dos pacotes necessários para tokenização e stemming usando nltk?

   Na Celula: IN [72] contém os downloads dos pacotes

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

```
import nltk

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

3. Em qual célula está o código que atualiza o spacy e instala o pacote pt_core_news_lg?

   Na Celula: IN [71]

   Atualizar o SPACY e instalar os modelos pt_core_news_lg

```
!pip install spacy
!python -m spacy download pt_core_news_lg

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

4. Em qual célula está o download dos dados diretamente do kaggle?

   Tive muitos problemas para executar o projeto no Colab. Várias vezes o processo era interrompido no meio, então optei por usar o VS Code. No meu código deixei comentado o processo de Download com o Kaggle pré-existente no projeto.

   Celula in [69]

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

from kaggle.api.kaggle_api_extended import KaggleApi

api = KaggleApi()
api.authenticate()

dataset = 'marlesson/news-of-the-site-folhauol'
api.dataset_download_files(dataset, path='.', unzip=True, force=True)

print("Download concluído!")
```

5. Em qual célula está a criação do dataframe news_2016 (com examente 7943 notícias)?

   Célula In [74]:

```
df['date'] = pd.to_datetime(df.date)

news_2016 = df[df['date'].dt.year == 2016]
news_2016 = news_2016[news_2016['category'] == 'mercado']

news_2016.dropna()
news_2016.drop_duplicates(keep='last')

#confirmação se o filtro foi de fato aplicado.
print(news_2016['date'].dt.year.unique())
news_2016.count()
```

```
2016]
title          7943
text           7943
date           7943
category       7943
subcategory       0
link           7943
dtype: int64
```

6. Em qual célula está a função que tokeniza e realiza o stemming dos textos usando funções do nltk?

   Celula IN [75]

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

def clear_text(text):
  if isinstance(text, str):
          text = text.lower()
          text = re.sub(r"\d+", "", text)
          text = re.sub(r"[^\w\s]", "", text)
          return text
  return ""


def tokenize(text: str) -> List:
  """
  Function for tokenizing using `nltk.tokenize.word_tokenize`

  Returns:
    - A list of stemmed tokens (`nltk.stem.RSLPStemmer`)
    IMPORTANT: Only tokens with alphabetic
               characters will be returned.
  """

  try:
      text = clear_text(text)
      if not text:
          return []
      tokens = word_tokenize(text)
      stemmer = RSLPStemmer()
      stemmed_tokens = [stemmer.stem(token) for token in tokens if token.isalpha()]
      return stemmed_tokens
  except Exception as e:
      print(f"Erro no texto: {text}, Tipo: {type(text)}, Erro: {e}")
      return []

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

7. Em qual célula está a função que realiza a lematização usando o spacy?

Célula IN [77]

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

complete_stopwords = stopwords()

def filter(w: spacy.lang.pt.Portuguese) -> bool:
    """
    Filter stopwords and undesired tokens
    """
    undesired = {"o", "em", "em o", "em a", "ano", "dizer",
                 "afirmar", "di", "data", "setor", "voltar", "passar", "levar", "trocar",
                 "anterior", "fazer", "ficar", "listar", "chegar", "caso", "necessario", "haver", "ser",
                 "poder", "dia", "informar", "hoje", "ir", "bom", "precisar", "ante", "subir", "efetuar",
                 "mostrar", "realizar", "apresentar", "considerar"
                 }
    return w.is_alpha and w.text.lower() not in complete_stopwords and w.lemma_.lower() not in undesired and len(w.text)


def lemma(doc: spacy.lang.pt.Portuguese) -> List[str]:
    """
    Apply spacy lemmatization on the tokens of a text

    Returns:
       - a list representing the standardized (with lemmatisation) vocabulary
    """
    tokens = [w.lemma_.lower() for w in doc if filter(w)]
    return tokens if tokens else ['vazio']

news_2016.loc[:, 'spacy_lemma'] = news_2016.spacy_doc.progress_map(lemma)
news_2016 = news_2016[news_2016.spacy_lemma.map(lambda x: len(x) > 2)]
```

8. Baseado nos resultados qual a diferença entre stemming e lematização, qual a diferença entre os dois procedimentos? Escolha quatro palavras para exemplificar.

Stemming é o processo que reduz a palavra a um radical (stem), muitas vezes de maneira abrupta, sem considerar o contexto linguístico ou a canonicidade da palavra. Isso pode gerar um ganho de performance em análises de textos, porém pode gerar palavras que não sejam reais e nem de fácil compreensão.

Exemplo 1: Correr → Corr

Exemplo 2: Correndo → Corr

Exemplo 3: Facilmente → Facil

Exemplo 4: Amando → Am

Como podemos ver nessa análise, para as 4 palavras sugeridas, apenas 1 seria facilmente identificada por nós.

Lematização é o processo que reduz a palavra a sua forma canônica(lema), ou seja, a forma presente em um dicionário, por exemplo. Ela considera o contexto gramatical da palavra, gerando palavras coerentes dentro dos idiomas do processamento. Ela requer mais poder de processamento, porém é um processo linguisticamente correto.

Usando as mesmas palavras do exemplo anterior:

Exemplo 1: Correr → Correr

Exemplo 2: Correndo → Correr

Exemplo 3: Facilmente → Facilmente

Exemplo 4: Amando → Amar

Podemos ter uma impressão de que o lema de "Facilmente" seria "Fácil", porém "Facilmente" é um advérbio derivado do adjetivo "Fácil", mas não é o mesmo lema gramatical.

Construir um modelo de reconhecimento de entidades (NER) usando Spacy

9. Em qual célula o modelo pt_core_news_lg está sendo carregado? Todos os textos do dataframe precisam ser analisados usando os modelos carregados. Em qual célula isso foi feito?

Célula in [76]:

```
nlp = spacy.load("pt_core_news_lg")
news_2016['text'] = news_2016['text'].fillna('').astype(str)

def create_spacy_docs(texts):
    docs = list(tqdm(nlp.pipe(texts), total=len(texts), desc="Processando textos com SpaCy"))
    return docs

news_2016.loc[:, 'spacy_doc'] = create_spacy_docs(news_2016['text'])
```

Neste código o modelo pt_core_news_lg foi carregado e utilizado no create_spacy_docs. Utilizei também o .pipe ao invés de passar os textos diretamente no construtor, pois ele fica mais rápido e mais eficiente ao lidar com grandes volumes de textos devivo ao processamento em lote que minimiza overheads(custos extras)

10. Indique a célula onde as entidades dos textos foram extraídas. Estamos interessados apenas nas organizações.

Celula IN [78]

Reconhecimento de entidades nomeadas

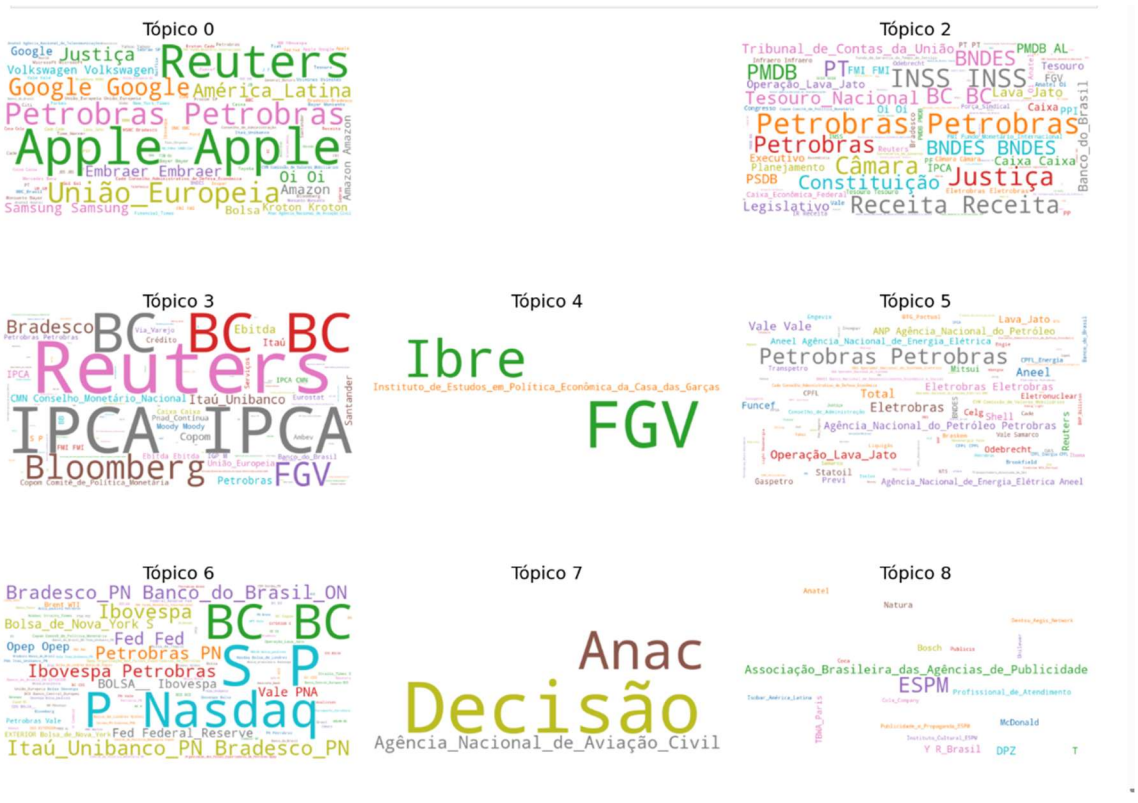
Crie uma coluna `spacy_ner` que armazene todas as organizações (APENAS organizações) que estão contidas no texto.

```
def NER(doc: spacy.lang.pt.Portuguese):
  """
  Return the list of organizations for a SPACY document
  """
  return [ent.text for ent in doc.ents if ent.label_ == "ORG"]
```

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

```
  0%|          | 0/7943 [00:00<?, ?it/s]
34207    [Ofcom, Cavamos, LEMA, Membro, Ordem do Impéri...
34238    [Oi, TIM, Claro, Vivo, GERCINA, FERNANDA BRIGA...
34245    [Executivo dos três Estados, Assembleias locai...
34248    [MIT, Instituto de Tecnologia de Massachusetts...
34249    [NACIONAIS  , Presidente da Singularity Univer...
                               ...
94280                                      [FGV, Ibre]
94295    [Nielsen, Nielsen, Nielsen, Nielsen, OFF!, ALT...
94301                    [Jovens, LCA, LCA, MAIOR NA BASE]
94310    [LCA Consultores, GO Associados, PME, PROCURA,...
94314    [Eletronuclear, Engevix, Operação Lava Jato, E...
Name: spacy_ner, Length: 7943, dtype: object
```

11. Cole a figura gerada que mostra a nuvem de entidades para cada tópico obtido (no final do notebook)



Criar modelos utilizando vetorização de textos baseado em Bag of Words

12. Quando adotamos uma estratégia frequentista para converter textos em vetores, podemos fazê-lo de diferentes maneiras. Mostramos em aula as codificações One-Hot, TF e TF-IDF. Explique a principal motivação em adotar TF-IDF frente as duas outras opções.

Brevemente, explicarei o conceito por trás de cada estratégia.

One-hot: Neste processo de vetorização, se uma palavra existir ela é “taxada“ como 1 e se não existir como 0. O principal problema dessa abordagem é que ela não é “frequentista”, ou seja, ela não consegue identificar o nível de importância de uma palavra com base na quantidade de vezes que ela aparece no documento. Isso é complicado para avaliar o nível de importância que uma palavra tem dentro de um texto.

TF (Frequência de termos): Diferente do One-hot, aqui a frequência de vezes em que as palavras aparecem, faz diferença. As palavras mais usadas recebem números maiores. O problema dessa abordagem é que palavras como “o”, “e”, “de” aparecem muito, mas não dizem muito sobre o conteúdo do texto.

TF-IDF (Frequência de termos – Frequência inversa de Documentos): Essa é uma técnica mais inteligente. Ela combina duas ideias que são TF, ou seja, quantas vezes a palavra aparece no texto e IDF, que é uma espécie de raridade da palavra em outros textos. Se uma palavra aparece muito em um texto específico, mas não aparece em muitos outros textos, ela é considerada importante.

Com base nisso, TF-IDF seria uma escolha mais inteligente para análises de tópicos, pois vai conseguir identificar corretamente a raridade das palavras em cada tópico, dando mais peso a palavras especiais e únicas e menos peso para palavras comuns que não ajudam muito na compreensão dos textos.

13. Indique a célula onde está a função que cria o vetor de TF-IDF para cada texto.

Célula IN[79]:

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


  def vectorizer(self):
      """
      Convert a list of tokens to tfidf vector
      Returns the tfidf vector and attribute it to self.tfidf
      """
      docs = [' '.join(tokens) for tokens in self.doc_tokens]
      self.vectorizer_model = TfidfVectorizer(
                  max_features=5000,
                  min_df=10
              )
      self.tfidf = self.vectorizer_model.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)
```

14. Indique a célula onde estão sendo extraídos os tópicos usando o algoritmo de LDA.

    Célula IN [80] (primeiro print) e Célula IN [82] (segundo print)

```
N_TOKENS = 9

corpus = np.array(news_2016.tfidf.tolist())
lda = LDA(
    n_components=N_TOKENS,
    max_iter=100,
    random_state=SEED,
    n_jobs=-1,
)

lda.fit(corpus)
```

```
def get_topic(tfidf: np.array):
    """
    Get topic for a lda trained model
    """
    topic_distribution = lda.transform(tfidf.reshape(1, -1))
    return int(np.argmax(topic_distribution))

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

15. Indique a célula onde a visualização LDAVis está criada.

    QUESTÃO ANULADA.

16. Cole a figura com a nuvem de palavras para cada um dos 9 tópicos criados.

Com 9 tópicos, percebi uma sobreposição de palavras-chave entre alguns clusters, o que prejudica a distinção entre temas. Por isso, reduzir para 5 traria mais clareza.

17. Escreva brevemente uma descrição para cada tópico extraído. Indique se você considera o tópico extraído semanticamente consistente ou não.

Tópico 0: Empresas e mercado brasileiro

Palavras-chave: empresa, brasil, país, mercado, cliente, companhia, investimento

Tema: Foco em atividades empresariais e relacionamento com clientes.

Tópico 1: Eliminado

Palavras-chave: ##

Tema: ##

Tópico 2: Governo e investimentos

Palavras-chave: Governo, acordo, país, empresas, bilhão, projeto, medida

Tema: Indicações de parcerias público-privadas ou algo sobre políticas de incentivo econômico.

Tópico 3: Setor energético e empresas estatais

Palavras-chave: estatal, petrobras, produção, operação, energia, leilão

Tema: Destaque para petrobras, leilões e produção

Tópico 4: Indicadores econômicos e desempenho do país

Palavras-chave: queda, país, inflação, alta, trimestre, resultado, banco central

Tema: Podemos ter assuntos sobre inflação, crescimento, PIB e variações trimestrais. No geral, desempenho do país.

Tópico 5: Debates e análises econômicas

Palavras-chave: conjuntura, econômico, análise, centro, cebrap, série, debate

Tema: provavelmente envolvendo centros de pesquisa como o CEBRAP.

Tópico 6: Política monetária e câmbio

Palavras-chave: moeda, banco central, swap, cambial, juros, petróleo, mercado

Tema: atuação do Banco Central, dólar, petróleo e taxas de juros.

Tópico 7: Infraestrutura de transporte aéreo

Palavras-chave: tarifa, embarque, aeroporto, decisão, reajustar, viracopos, guarulhos

Tema: tarifas aeroportuárias, voos e decisões da ANAC.

Tópico 8: Educação e marketing

Palavras-chave: faculdade, espm, evento, marketing, site, programa, parceria

Tema: eventos promovidos por instituições como ESPM, marketing digital e parcerias.

Criar modelos baseados em Word Embedding

18. Neste projeto, usamos TF-IDF para gerar os vetores que servem de entrada para o algoritmo de LDA. Quais seriam os passos para gerar vetores baseados na técnica de Doc2Vec?

A Doc2Vec é uma técnica neural baseada em embeddings, e não apenas em frequência de termos. Para realizar a aplicação dela, os passos seriam um pouco diferentes:

1 – Pré-processamento do texto (nenhuma novidade por aqui)

Tokenização, remoção de Stopwords, normalização e se preciso, lematização ou stemming se necessário.

2- Construção de documentos rotulados.

Cada documento precisa ser identificado com uma etiqueta única, como se desse um nome para cada texto. Isso é necessário porque o Doc2Vec precisa saber qual vetor pertence a qual documento durante o treinamento. Isso é feito criando objetos chamados TaggedDocument, que associam o conteúdo do documento à sua respectiva tag.

3 – Treinamento Doc2Vec

O modelo tem dois modos principais: o Distributed Memory (DM), que tenta prever uma palavra com base no contexto do parágrafo (semelhante ao CBOW), e o Distributed Bag of Words (DBOW), que ignora a ordem das palavras e tenta prever palavras aleatórias do documento (semelhante ao Skip-Gram). Durante esse treinamento, o modelo aprende a representar cada documento como um vetor de números em um espaço de alta dimensão, de forma que textos com significados parecidos fiquem com vetores próximos.

Esses vetores, que capturam de forma densa o significado do texto, podem ser usados para diversas tarefas: agrupar documentos por similaridade, classificar textos, visualizar relações semânticas e até mesmo aplicar outras técnicas de modelagem de tópicos mais modernas, como o BERTopic ou o Top2Vec.

Depois que o modelo está treinado, é possível obter o vetor de qualquer documento usando a chave com que ele foi etiquetado ou, se for um documento novo, pode inferir um vetor com base nas palavras que ele contém.

Embora não tenha usado Doc2Vec diretamente neste projeto, apliquei conceitos semelhantes ao criar embeddings com a OpenAI para um agente IA com busca semântica em um banco Qdrant que estou desenvolvendo para a empresa em que trabalho. Por mais que seja diferente do Doc2Vec, o princípio de vetorização semântica foi aplicado com sucesso.

19. Em uma versão alternativa desse projeto, optamos por utilizar o algoritmo de K-Médias para gerar os clusters (tópicos). Qual das abordagens (TF-IDF ou Doc2Vec) seria mais adequada como processo de vetorização? Justifique com comentários sobre dimensionalidade e relação semântica entre documentos.

Se optarmos em utilizar o K-médias para gerar os clusters(tópicos), a vetorização com Doc2Vec geralmente é mais adequada do que o TF-IDF devido a alguns fatores:

- Relação semântica entre os documentos: TF-IDF representa os documentos com base na frequência das palavras. Ele não consegue entender sinônimos nem o contexto diretamente, ou seja, dois textos que falam sobre a mesma coisa, escritos de formas diferentes, podem, e provavelmente vão, ficar muito distantes um do outro.

Doc2Vec gera vetores densos que capturam o significado do texto como um todo, o que permite que documentos semanticamente similares fiquem mais próximos, mesmo que usem palavras diferentes. Isso é ideal para o K-Means, que depende fortemente da **distância** entre vetores para formar os clusters.

- Dimensionalidade: O TF-IDF costuma gerar vetores com altíssima dimensionalidade (por exemplo 10.000 dimensões ou mais dependendo do vocabulário). Isso pode ser problemático para o K-means, que não lida bem com dados esparsos em espaços muito grandes.

Doc2Vec, por outro lado, gera vetores densos e de baixa dimensão (geralmente entre 100 e 300 dimensões), o que é muito mais apropriado para algoritmos como o K-Means, que são sensíveis à forma do espaço vetorial.

- Eficiência e desempenho : Rodar o K-Means em vetores TF-IDF pode ser mais lento e menos eficaz na separação semântica dos clusters, enquanto com Doc2Vec, além de ser mais leve computacionalmente, o agrupamento tende a ser mais coerente em termos de conteúdo.

20. Leia o artigo "Introducing our Hybrid lda2vec Algorithm" (https://multithreaded.stitchfix.com/blog/2016/05/27/lda2vec/#topic=38&lambda=1&term=) .
O algoritmo lda2vec pretende combinar o poder do word2vec com a interpretabilidade do algoritmo LDA. Em qual cenário o autor sugere que há benefícios para utilização deste novo algoritmo?

O autor sugere que os benefícios do lda2vec se destacam quando se trabalha com grandes volumes de texto não estruturados e se deseja tanto a capacidade de identificar tópicos interpretáveis (como no LDA) quanto capturar relações semânticas sutis entre palavras (como o word2vec). Em outras palavras, o cenário ideal é quando é necessário explorar e analisar corpora complexos, por exemplo, os comentários e posts do Hacker News, onde, além de descobrir tópicos coerentes e facilmente interpretáveis, é importante ter a inteligência semântica que permite operar sobre esses dados de forma mais refinada (como fazer “álgebra de palavras”). Essa combinação oferece insights mais ricos e uma compreensão mais profunda dos temas emergentes no conjunto de dados.

Se quisermos apenas descobrir os assuntos principais (os **tópicos**) de um monte de textos de forma que uma pessoa consiga entender, use o **LDA** que é um método já bem conhecido e confiável, com ferramentas prontas para isso.

Mas se quisermos fazer algo mais sofisticado como, por exemplo, ligar os temas de um texto com a quantidade de curtidas que ele recebeu, ou prever os interesses de um usuário com base nos temas dos textos que ele lê, aí o lda2vec pode ser interessante. Ele é uma mistura dos dois métodos anteriores.

Porém, há desvantagens:

O lda2vec é experimental e foi criado mais como uma forma de testar ideias do que como uma ferramenta pronta para uso. Ele também precisa de muito poder de processamento, além do mais o autor não garante que ele seja melhor ou pior do que o LDA ou o word2vec, depende do caso.

Achei interessante o potencial do lda2vec, mas no contexto do nosso projeto, talvez o ganho não justificasse o custo computacional e o processo de entendimento humano.