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://colab.research.google.com/drive/1qj0qbZKlZsgS5j1FI1PELQOW4fwW2Xe1?usp=sharing

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

Na linha 971 do arquivo colab, no trecho:

"nltk.download('stopwords')\n",

"nltk.download('punkt')\n",

"nltk.download('punkt_tab')\n",

"nltk.download('rslp')"

- stopwords (para remoção de palavras comuns)
- punkt (para tokenização)
- punkt_tab
- rslp (Removedor de Sufixos da Língua Portuguesa, usado para stemming)

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

O código que atualiza o spaCy e instala o pacote pt_core_news_lg está na célula que começa na linha 874 do notebook

Questionário_projeto.ipynb

. Esta célula contém os seguintes comandos:

1. !pip install --quiet spacy

(para atualizar o spaCy)

2. !python -m spacy download pt_core_news_lg

(para instalar o modelo em português)

3. E depois carrega o modelo usando

spacy.load('pt_core_news_lg')

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

O download dos dados diretamente do Kaggle está na célula que começa na linha 354 do notebook

. O comando específico usado é:

CopyInsert

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

Esta célula faz parte de uma sequência de células que:

1. Primeiro instala o gerenciador Kaggle
2. Configura as credenciais necessárias
3. E finalmente executa o download do dataset de notícias da Folha UOL

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

   O dataframe news_2016 (com 7943 notícias) é criado na célula que começa na linha 1490, usando o comando:

news_2016 = df.loc[(df['date'].dt.year == 2016) & (df['category'] == 'mercado')]

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

A função que tokeniza e realiza o stemming usando NLTK está na célula que começa na linha 1647, contendo a função tokenize() que usa nltk.tokenize.word_tokenize

para tokenização e nltk.stem.RSLPStemmer() para stemming.

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

A função que realiza a lematização usando spaCy está na célula que começa na linha 2281, contendo a função lemma() que usa o atributo lemma do spaCy para lematização.

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

   A diferença entre stemming e lematização:

- Stemming é um processo mais simples que corta os sufixos das palavras para chegar ao seu radical (stem), sem considerar o contexto ou significado.

- Lematização é um processo mais sofisticado que reduz a palavra à sua forma base (lema) considerando o contexto e a análise morfológica da palavra.

Exemplos com 4 palavras em português:

1. "correndo"
    - Stemming: "corr"
    - Lematização: "correr"
2. "brasileiros"
    - Stemming: "brasil"
    - Lematização: "brasileiro"
3. "melhores"
    - Stemming: "melhor"
    - Lematização: "bom"
4. "estudando"
    - Stemming: "estud"
    - Lematização: "estudar"

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?**

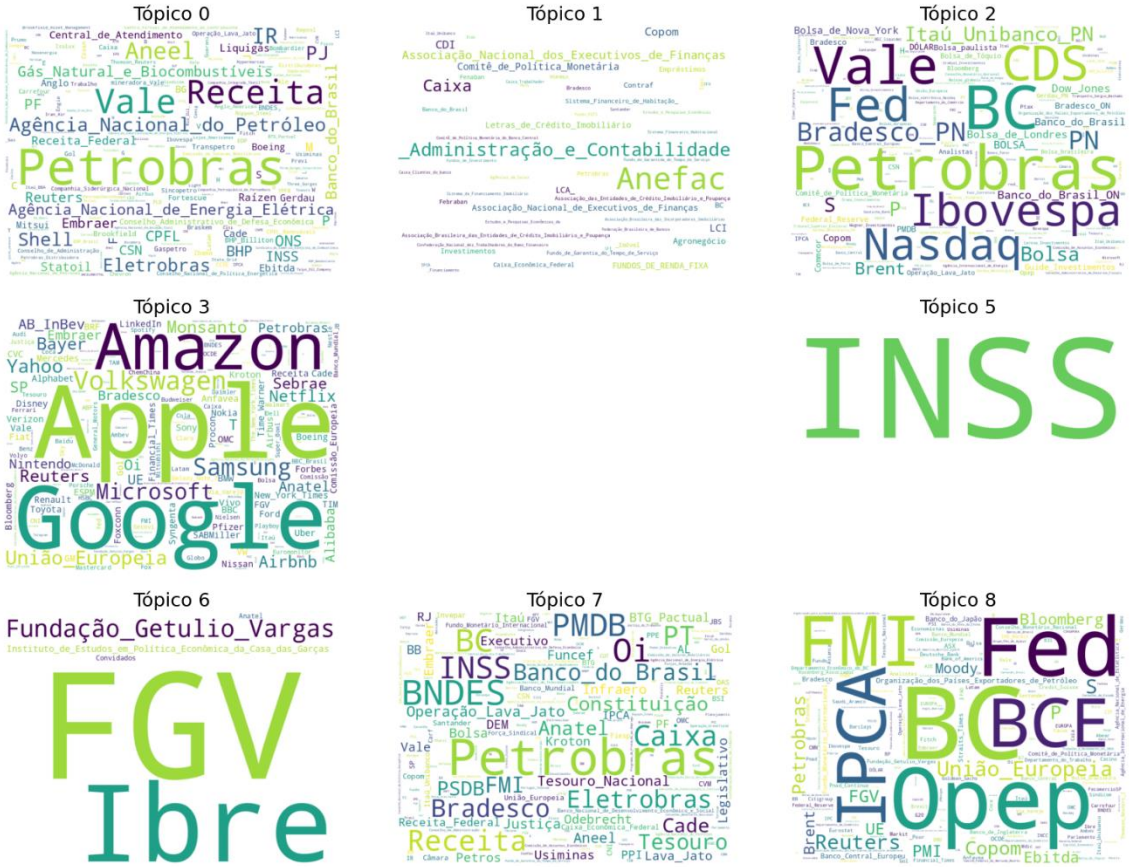
   O modelo pt_core_news_lg foi carregado na célula que contém o código nlp = pacy.load('pt_core_news_lg') .

   Antes disso, houve uma célula de instalação do modelo usando !python -m spacy download pt_core_news_lg.

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

    A extração de entidades foi feita na seção "Reconhecimento de entidades nomeadas" do notebook, que aparece após a linha 2303.

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



11. 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.

A principal motivação para usar TF-IDF (Term Frequency-Inverse Document Frequency) em vez de One-Hot Encoding ou TF (Term Frequency) pura é que o TF-IDF considera não apenas a frequência dos termos em um documento específico, mas também a importância relativa desses termos em toda a coleção de documentos.

O TF-IDF reduz o peso de palavras que aparecem muito frequentemente em todos os documentos (como artigos, preposições) e aumenta o peso de palavras que são mais distintivas e importantes para documentos específicos. Isso resulta em uma representação mais significativa que:

- Supera a limitação do One-Hot Encoding, que trata todas as palavras como igualmente importantes e não considera frequências
- Melhora sobre o TF puro, que pode dar muito peso a palavras comuns que aparecem frequentemente mas têm pouco valor discriminativo

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

A linha do notebook que está extraindo cada texto de TFIDF é a linha 2633.

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

LDA (Latent Dirichlet Allocation) é implementado no notebook. A implementação está na célula [17] do notebook, que começa na linha 3308 do arquivo.

Aqui está o trecho de código relevante:

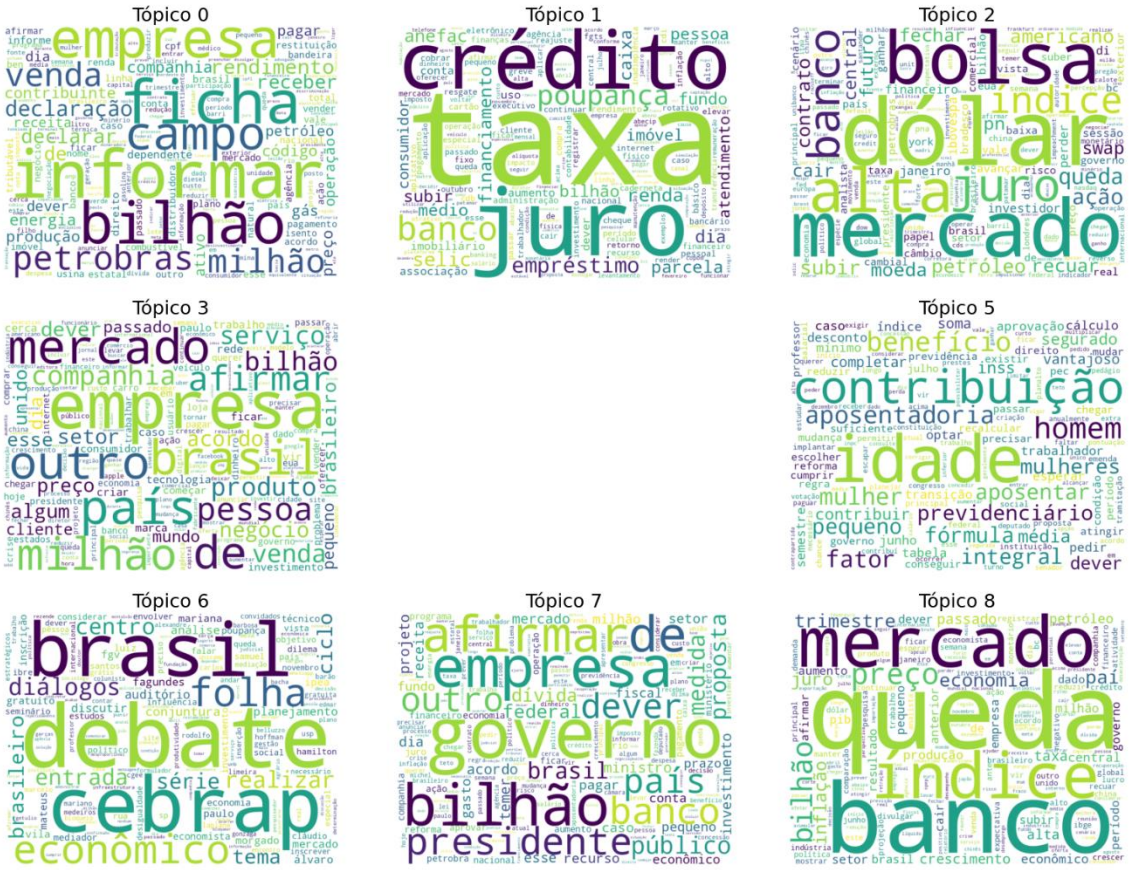
```
from sklearn.decomposition import LatentDirichletAllocation as LDA


lda = LDA(n_components=N_TOKENS,
        max_iter=100,
        random_state=SEED
       ).fit(corpus)
```

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

Segundo o professor essa questão não precisa ser respondida.

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



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

Tópico 0: Relacionado ao setor empresarial e financeiro, com foco em vendas, declaração e campos de negócio. Palavras como "empresa", "forma", "declaração" e "bilhão" sugerem aspectos corporativos e fiscais. Consistência: Alta - as palavras formam um campo semântico coerente relacionado a negócios.

Tópico 1: Centrado em crédito e operações bancárias, com termos como "taxa", "juro", "banco", "empréstimo" e "poupança". Consistência: Alta - vocabulário claramente relacionado ao setor financeiro e bancário.

Tópico 2: Foca em mercado financeiro, especialmente bolsa de valores, com termos como "bolsa", "índice", "alta" e "juros". Consistência: Alta - termos consistentemente relacionados ao mercado de capitais.

Tópico 3: Aborda mercado e negócios, com ênfase em valores ("bilhão", "milhão") e aspectos comerciais ("empresa", "setor", "produto"). Consistência: Alta - vocabulário coerente do campo empresarial e comercial.

Tópico 5: Relacionado à previdência e contribuições, com termos como "contribuição", "aposentadoria", "benefício" e "previdenciário". Consistência: Alta - campo semântico bem definido sobre previdência social.

Tópico 6: Focado em debates e discussões sobre o Brasil, com termos como "debate", "Brasil", "folha", "economia", "febrap". Consistência: Média - apresenta alguma dispersão temática, mas mantém foco em discussões nacionais.

Tópico 7: Centrado em governo e administração pública, com palavras como "governo", "presidente", "público", "federal". Consistência: Alta - vocabulário consistente relacionado à administração pública.

Tópico 8: Relacionado ao mercado financeiro e índices econômicos, com termos como "mercado", "queda", "índice", "banco". Consistência: Alta - termos coerentemente relacionados ao mercado financeiro.

Em geral, a maioria dos tópicos apresenta alta consistência semântica, com vocabulário bem relacionado dentro de cada grupo. Apenas o Tópico 6 mostra uma dispersão um pouco maior, embora ainda mantenha certa coerência temática.

17. 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?

Doc2Vec (ou Paragraph Vector) é uma extensão do Word2Vec que aprende representações vetoriais contínuas para documentos inteiros, não apenas palavras individuais. Aqui estão os passos principais:

Preparação dos Dados:

- Pré-processamento do texto (similar ao que já é feito para TF-IDF)
- Tokenização dos documentos
- Remoção de stopwords e limpeza do texto

Principais Diferenças em Relação ao TF-IDF:

- Doc2Vec captura semântica e contexto, não apenas frequência de palavras
- Os vetores são densos (ao contrário dos esparsos do TF-IDF)
- Requer treinamento do modelo (é um método baseado em aprendizado)
- Geralmente produz vetores de menor dimensionalidade

Vantagens:

- Captura melhor relações semânticas entre documentos
- Pode lidar melhor com sinônimos e palavras relacionadas
- Vetores de tamanho fixo, independente do tamanho do vocabulário

Considerações:

- Ajuste os hiperparâmetros (vector_size, window, min_count, etc.) conforme necessário
- O treinamento pode levar mais tempo que TF-IDF
- Recomenda-se experimentar tanto PV-DM (dm=1) quanto PV-DBOW (dm=0)

Para usar estes vetores com LDA, você precisaria apenas substituir a matriz TF-IDF pelos vetores gerados pelo Doc2Vec como entrada para o algoritmo LDA.

Vou criar uma memória sobre esta alternativa de vetorização, pois pode ser útil para referência futura.

18. 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.

Para responder sobre qual abordagem de vetorização (TF-IDF ou Doc2Vec) seria mais adequada para usar com K-Médias, vou analisar os aspectos principais considerando dimensionalidade e relação semântica:

Doc2Vec seria a escolha mais adequada para usar com K-Médias pelos seguintes motivos:

1. Dimensionalidade:

- Doc2Vec gera vetores densos com dimensionalidade controlada (através do parâmetro vector_size)
- TF-IDF gera vetores esparsos com dimensionalidade igual ao tamanho do vocabulário
- K-Médias tem melhor performance com vetores densos e de menor dimensionalidade, pois:
    - Reduz o impacto da "maldição da dimensionalidade"
    - Facilita o cálculo de distâncias entre pontos
    - Melhora a convergência do algoritmo

2. Relação Semântica:

- Doc2Vec captura relações semânticas e contextuais entre palavras/documentos
- Documentos com significados similares tendem a ter vetores próximos mesmo usando palavras diferentes
- Isso permite que o K-Médias agrupe documentos semanticamente relacionados mesmo quando não compartilham exatamente os mesmos termos

3. Praticidade:

- A dimensionalidade fixa do Doc2Vec (definida por vector_size) facilita a experimentação com diferentes números de clusters
- Com TF-IDF seria necessário aplicar redução de dimensionalidade (como PCA) antes do K-Médias

Em resumo, Doc2Vec é mais adequado por gerar representações densas de baixa dimensionalidade que preservam relações semânticas, características ideais para clustering com K-Médias.

19. 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 o algoritmo lda2vec é especialmente benéfico em cenários onde se deseja manter a qualidade das representações vetoriais de palavras (como no word2vec), mas também se quer interpretar os tópicos de maneira clara e compreensível, como no LDA (Latent Dirichlet Allocation).

Mais especificamente, o artigo destaca os seguintes cenários vantajosos para o uso do lda2vec:

- Análise de grandes volumes de texto não estruturado, onde é importante capturar relações semânticas entre palavras (capacidade do word2vec), mas também compreender os temas principais dos textos (ponto forte do LDA).
- Quando se deseja identificar tópicos interpretáveis com palavras coerentes entre si, enquanto ainda se utiliza embeddings densos e semanticamente ricos.
- Em aplicações onde se quer associar documentos a tópicos de forma eficiente, mantendo representações vetoriais úteis para tarefas de machine learning posteriores.

Em resumo, o lda2vec é indicado quando se busca o melhor dos dois mundos: vetores de palavras ricos semanticamente + tópicos interpretáveis, sendo útil, por exemplo, em sistemas de recomendação, organização de conteúdo textual, e análise de sentimentos com base em tópicos.