UNIVERSIDADE
FEDERAL DA PARAÍBA

CLASSIFICADOR DE LETRAS MUSICAIS

KEVIN LEANDRO GOMES
NATANAEL DE LIMA COSTA NETO

29/04/2025

SUMÁRIO

INTRODUÇÃO

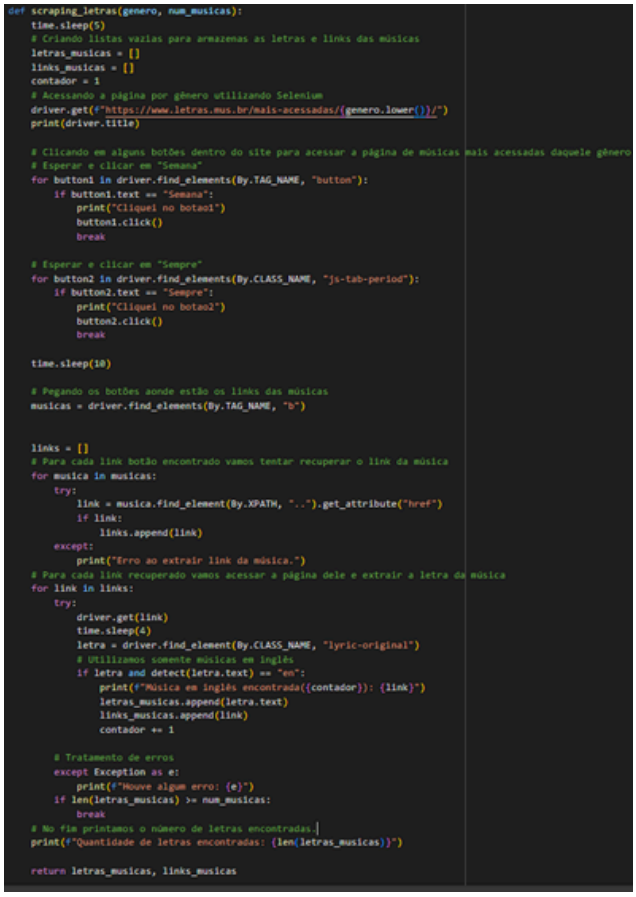
O presente trabalho foca no desenvolvimento de um classificador de letras musicais de diferentes gêneros musicais, utilizando técnicas de processamento de linguagem natural e modelos de aprendizagem de máquina.
Foi realizada a coleta de dados, pré-processamento textual, vetorização e treinamento de modelos, com avaliação de desempenho através de validação. Além disso, também foram realizadas pesquisas e comparações com projetos semelhantes e/ou já existentes.

OBJETIVOS

O principal foco deste trabalho foi analisar se existe algum determinado padrão em letras musicais, se cada gênero pode ser classificado pelas letras das músicas associadas a esse gênero, ou se não existe esse determinado padrão. Além disso, buscou-se construir modelos de classificação utilizando Árvore de Decisão, Máquina de Vetor de Suporte e Redes Neurais, para avaliar o desempenho através de técnicas de validação, com objetivo de verificar qual modelo apresentou melhores métricas considerando os dados coletados.

METODOLOGIA E DESENVOLVIMENTO

Coleta de dados: Afim de obter maior variedade de gêneros musicais, as músicas foram coletadas somente na língua inglesa. Inicialmente, tentou-se utilizar bibliotecas específicas para sites de músicas a fim de realizar a coleta automática das letras para cada gênero musical (Rock, Pop, Hip-Hop/Rap e Country). Entretanto, devido a dificuldades e limitações no uso de algumas APIs, como Spotipy e GeniusAPI, optou-se pela utilização da ferramenta Selenium para realizar a raspagem direta dos dados em sites de músicas, recuperando diretamente do HTML das páginas as informações relevantes para o projeto, como a letra da música, o link para acessá-la e o gênero correspondente. Todos esses dados foram coletados a partir do site https://www.letras.mus.br/. Para o propósito deste projeto, foram utilizadas 500 músicas de cada gênero musical.



```python
def scraping_letras(genero, num_musicas):
    time.sleep(5)
    # Criando listas vazias para armazenas as letras e links das músicas
    letras_musicas = []
    links_musicas = []
    contador = 1
    # Acessando a página por gênero utilizando Selenium
    driver.get(f"https://www.letras.mus.br/mais-acessadas/{genero.lower()}/")
    print(driver.title)

    # Clicando em alguns botões dentro do site para acessar a página de músicas mais acessadas daquele gênero
    # Esperar e clicar em "Semana"
    for button1 in driver.find_elements(By.TAG_NAME, "button"):
        if button1.text == "Semana":
            print("Cliquei no botao1")
            button1.click()
            break

    # Esperar e clicar em "Sempre"
    for button2 in driver.find_elements(By.CLASS_NAME, "js-tab-period"):
        if button2.text == "Sempre":
            print("Cliquei no botao2")
            button2.click()
            break

    time.sleep(10)

    # Pegando os botões aonde estão os links das músicas
    musicas = driver.find_elements(By.TAG_NAME, "b")


    links = []
    # Para cada link botão encontrado vamos tentar recuperar o link da música
    for musica in musicas:
        try:
            link = musica.find_element(By.XPATH, "..").get_attribute("href")
            if link:
                links.append(link)
        except:
            print("Erro ao extrair link da música.")
    # Para cada link recuperado vamos acessar a página dele e extrair a letra da música
    for link in links:
        try:
            driver.get(link)
            time.sleep(4)
            letra = driver.find_element(By.CLASS_NAME, "lyric-original")
            # Utilizamos somente músicas em inglês
            if letra and detect(letra.text) == "en":
                print(f"Música em inglês encontrada({contador}): {link}")
                letras_musicas.append(letra.text)
                links_musicas.append(link)
                contador += 1

        # Tratamento de erros
        except Exception as e:
            print(f"Houve algum erro: {e}")
        if len(letras_musicas) >= num_musicas:
            break
    # No fim printamos o número de letras encontradas
    print(f"Quantidade de letras encontradas: {len(letras_musicas)}")

    return letras_musicas, links_musicas
```

Funcão de Scraping diretamente do código.

Processamento dos dados: Todas as letras foram coletadas e armazenadas em arquivos .csv, de onde posteriormente foram retiradas e processadas, de modo a reduzir o vocabulário e eliminar partições textuais desinteressantes para os modelos (como pontuações, símbolos, espaços em branco e contrações). Também foram removidas stopwords utilizando os pacotes do NLTK, que possuem listas de stopwords por idioma. Ainda com o uso do pacote NLTK, foram aplicadas duas técnicas de normalização textual: Stemming e Lematização, optando-se pelo uso da Lematização, pois ela proporcionou uma redução significativa do vocabulário e melhores resultados. Por fim, cada música foi convertida em um vetor numérico utilizando a técnica de TF-IDF (Term Frequency–Inverse Document Frequency).

Exemplo de como fica a letra da música pós tratamento.

	Palavra	Valor_TFIDF
815	breathin	0.951369
3447	keep	0.197730
158	air	0.113803
2249	fabricated	0.059461
2284	fallin	0.059425
5441	runnin	0.056591

Exemplo de como fica o vetor TF-IDF.

Treinamento dos Modelos: Os dados foram separados em conjuntos de treino e teste, utilizando um fator aleatório de separação e utilizando 20% dos dados totais para o conjunto de teste. Três modelos de aprendizagem foram utilizados no processo de classificação, de modo a obter aquele com melhor desempenho(melhores métricas). Nos 3 modelos foi utilizada a técnica de validação cruzada para encontrar os melhores hiperparâmetros para os modelos treinados, particionando o conjunto de treinamento em 5 partes de tamanhos iguais(5-Fold), utilizando 1 das partes para treinamento e outra para validação, repetindo esse processo 5 vezes alternando os conjuntos de validação e testando diferentes combinações de hiperparâmetros.

Para o SVM(Support Vector Machine) o melhor valor de Gamma e C encontrados foram: Gamma = 1 e C = 1.

Para a Árvore de Decisão, o melhor valor para o alfa encontrado foi: 0.0116

Já para a arquitetura da Rede Neural, foram utilizadas 2 camadas ocultas, com 8 e 4 neurônios respectivamente, utilizando o tamanho de 32 para o batch e 30 épocas, essa arquitetura foi escolhida após a análise da dimensão dos dados e diversos testes com variados números de camadas e neurônios por camada. Também foi feito uma redução de dimensionalidade dos dados utilizando a ferramenta PCA, que reduz o número de dimensões através da análise de probabilidades.

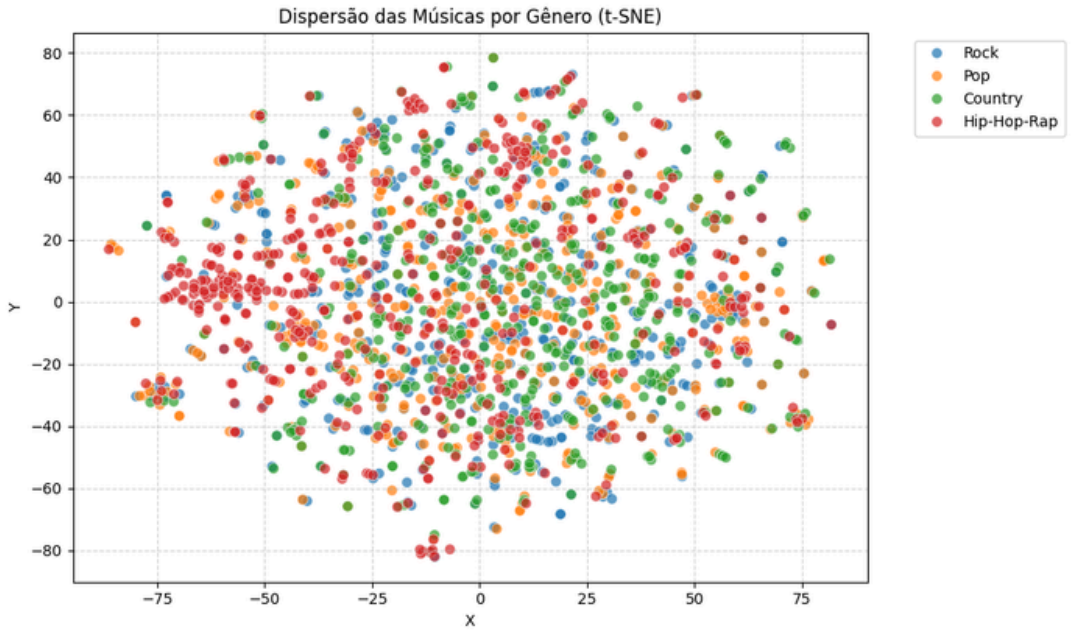


Gráfico de dispersão das músicas.

Por fim, para efeito de comparação e testes, utilizamos uma LSTM com representações das palavras em forma de embeddings e um conjunto de dados maior, contendo mais letras(para este teste utilizamos Jazz e Metal, pois o dataset não contava com tantos dados para Hip-Hop e Country). Utilizando o tamanho de 128 para a dimensão dos embeddings e 64 neurônios na camada do LSTM.

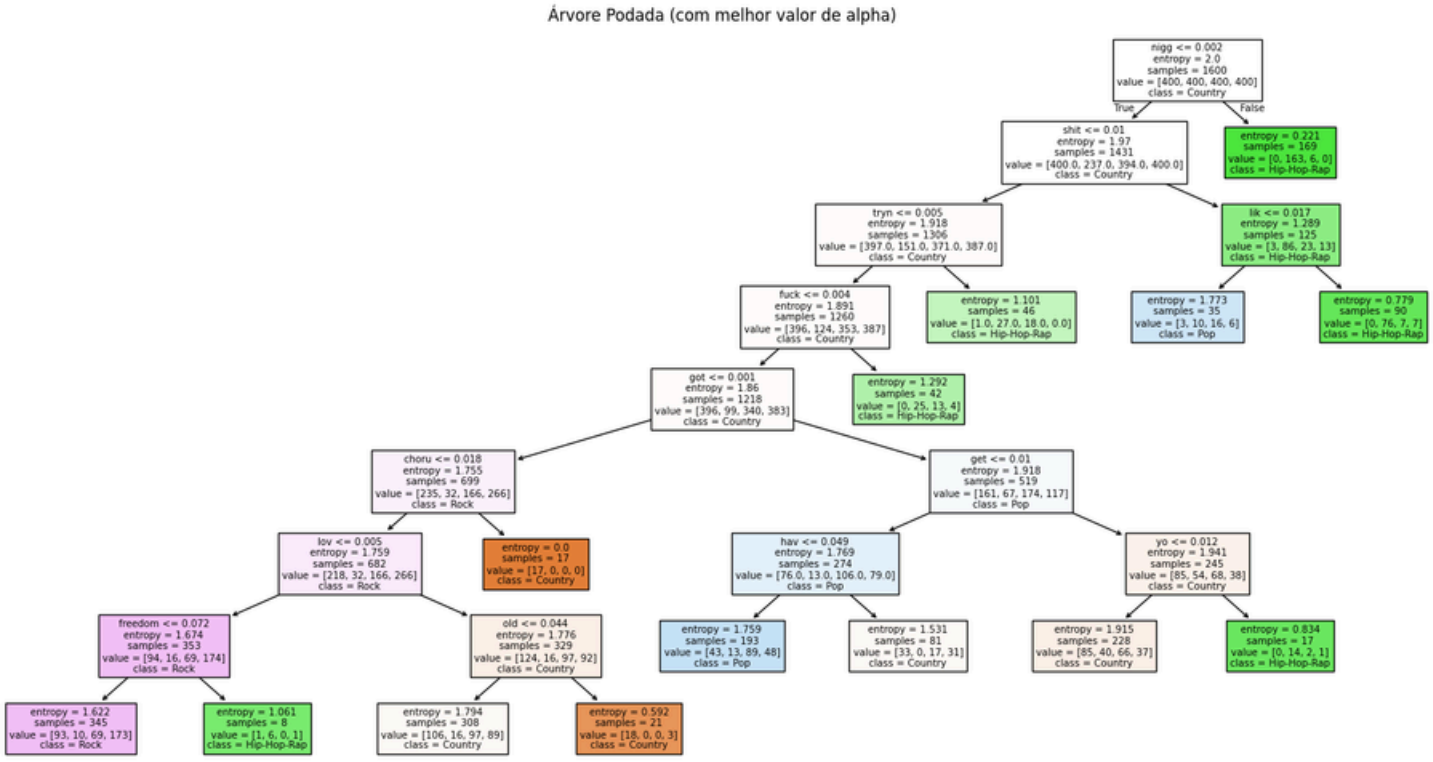
RESULTADOS

Foi observado um comportamento inadequado dos modelos de classificação: a maioria não generalizou bem para novos dados, principalmente a Rede Neural, que não conseguiu aprender adequadamente, mesmo após a redução de dimensionalidade. Notou-se overfitting no modelo de SVM, o que pode ser resultado da natureza dos dados, pois estes apresentam muito ruído — especialmente nos gêneros musicais Pop e Rock, que estão bastante sobrepostos. Artistas como o Coldplay, por exemplo, estão classificados em ambas as categorias, devido à forma como o site de onde foi feito o scraping organiza seus gêneros musicais.

Outros gêneros, como Rap, são bem mais facilmente separáveis pelos modelos, pois estão mais distantes dos demais gêneros. A única forma de tratar essa questão seria realizando uma rotulação manual dos dados, para evitar esse tipo de ruído, já que a maioria dos sites e fontes de dados pesquisados para o scraping apresentava essa mesma problemática.

No modelo de Rede Neural, observou-se claro underfitting após a redução de dimensionalidade: o modelo não conseguiu aprender adequadamente. Já com muitas dimensões, o modelo apresentou overfitting, pois existem muitos atributos em comparação ao número de amostras disponíveis.

O modelo que obteve a melhor métrica foi a LSTM, que com mais dados e uma melhor representação das palavras (camada de Embeddings), conseguiu uma acurácia superior aos outros modelos sem gerar overfit, ainda assim, generos muito semelhantes, Pop e Rock por exemplo, são mais dificeis de serem separados pelo modelo, limitando sua acurácia, testando sem o gênero Pop, obtivemos uma acurácia maior. É possível que com mais dados para cada gênero, obtenhamos melhores métricas.



Melhor árvore obtida.

	precision	recall	f1-score	support
Jazz	0.80	0.72	0.75	2000
Metal	0.68	0.73	0.70	2000
Pop	0.67	0.45	0.54	2000
Rock	0.44	0.60	0.51	2000
accuracy			0.62	8000
macro avg	0.65	0.62	0.62	8000
weighted avg	0.65	0.62	0.62	8000

Métricas da LSTM

	precision	recall	f1-score	support
Jazz	0.80	0.76	0.78	2000
Metal	0.72	0.78	0.75	2000
Rock	0.59	0.58	0.59	2000
accuracy			0.71	6000
macro avg	0.71	0.71	0.71	6000
weighted avg	0.71	0.71	0.71	6000

Métricas da LSTM sem o gênero ruídoso

CONCLUSÃO

Por fim, concluímos que a quantidade de dados utilizada para o treinamento dos modelos era muito pequena e apresentava muitos ruídos. Ainda assim, conseguimos realizar uma boa análise das causas desses problemas nos modelos e identificamos as dificuldades associadas aos rótulos das músicas disponíveis para a raspagem de dados. Uma provável melhoria para esse classificador seria utilizar um dataset com um conjunto maior de letras de diferentes gêneros.

Acreditamos firmemente que, ao se dispor de uma quantidade maior de amostras e amostras com menos ruídos (por meio de uma rotulagem feita de forma correta), o classificador poderia melhorar significativamente.

Neste processo, aprendemos muito sobre classificação textual, sobre aprendizagem de forma geral e sobre como a classificação de textos ainda é um grande desafio.