**Mineração de Texto no PooArt**

Matheus Teodoro

Julia

Valdir Cezar

# 1 INTRODUÇÃO

Vamos acompanhar como utilizamos a mineração de texto em nosso projeto. Um sistema Web que trará a facilidade em encontrar um artista informal ou formal de maneira simples e prática através de uma breve pesquisa no site que será desenvolvido pelo time. O projeto está sendo baseado em um case de sucesso, GetNinjas.

Dentro de nosso sistema decidimos criar uma API em Pyhton com o Django Framework para utilizar a mineração de texto e classificar cada avaliação dada a um artista que já realizou seu trabalho. Utilizamos a biblioteca NLTK essa biblioteca é extremamente útil, pois, possui um extenso dicionário embutido, diversas ferramentas para manipulação, tratamento de bases e muitos outros recursos.

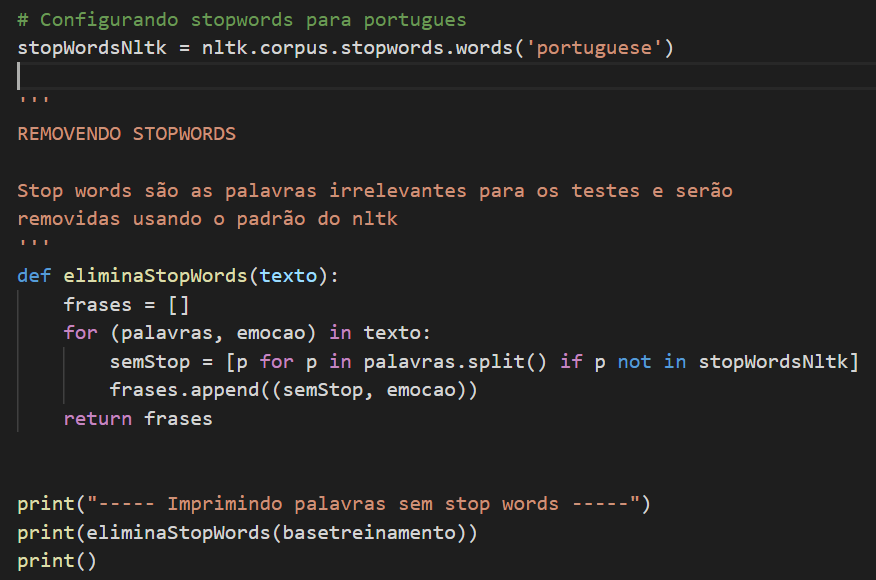
# 2 OBJETIVOS DO PROJETO

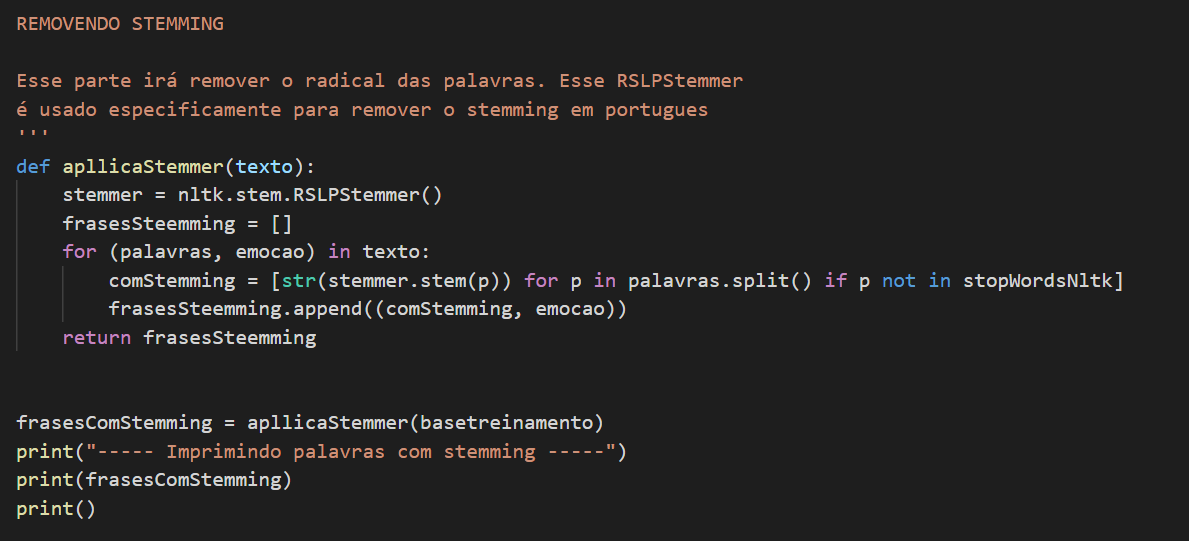
De inicio temos o objetivo de lançar uma versão beta e gratuita onde o usuário precisa apenas se cadastrar e se for um artista adicionar suas habilidades e preencher seu perfil. Em caso de uma empresa, bar, restaurante ou outros, também será solicitado um cadastro.

**2.1 FUNCIONALIDADES**

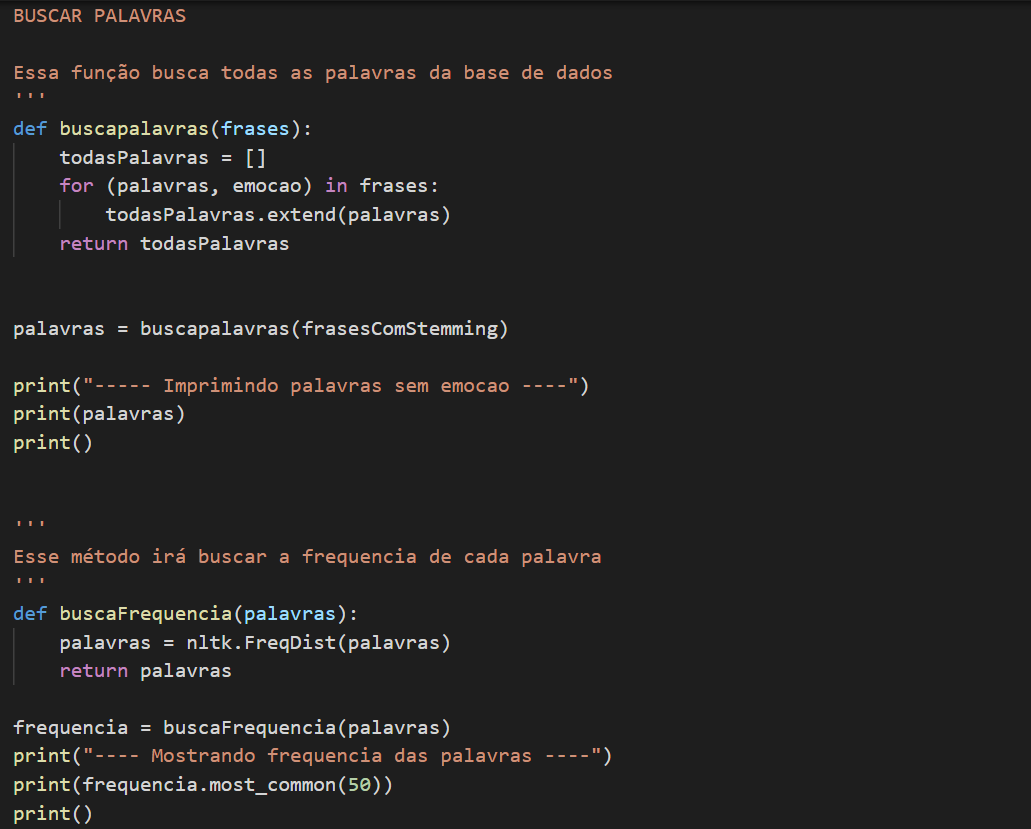
* **Sistema de classificação de avaliações com NLTK**

## 2.1.1.Mineração de Texto e API

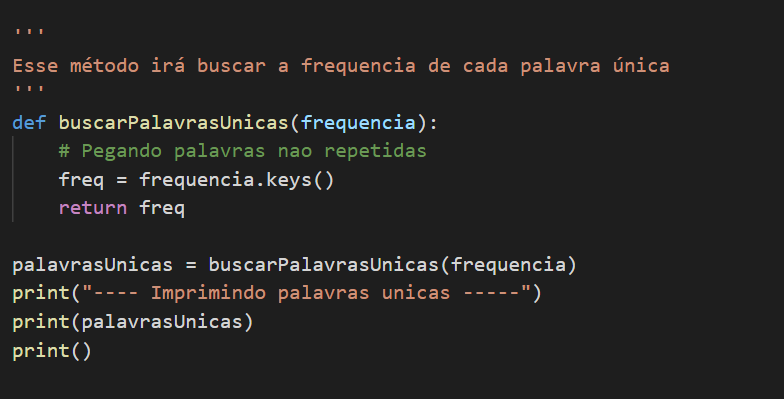




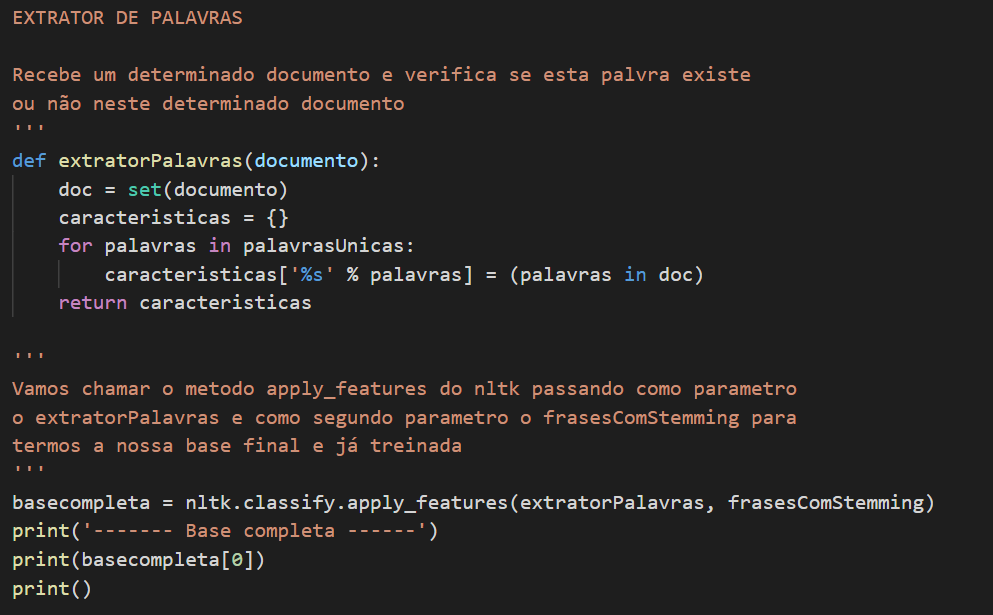
Usando o NLTK, esse pacote contém radicais de várias línguas assim como **Stopwords**, e o que rola é que elas variam de língua para língua, então precisamos referenciar o português.

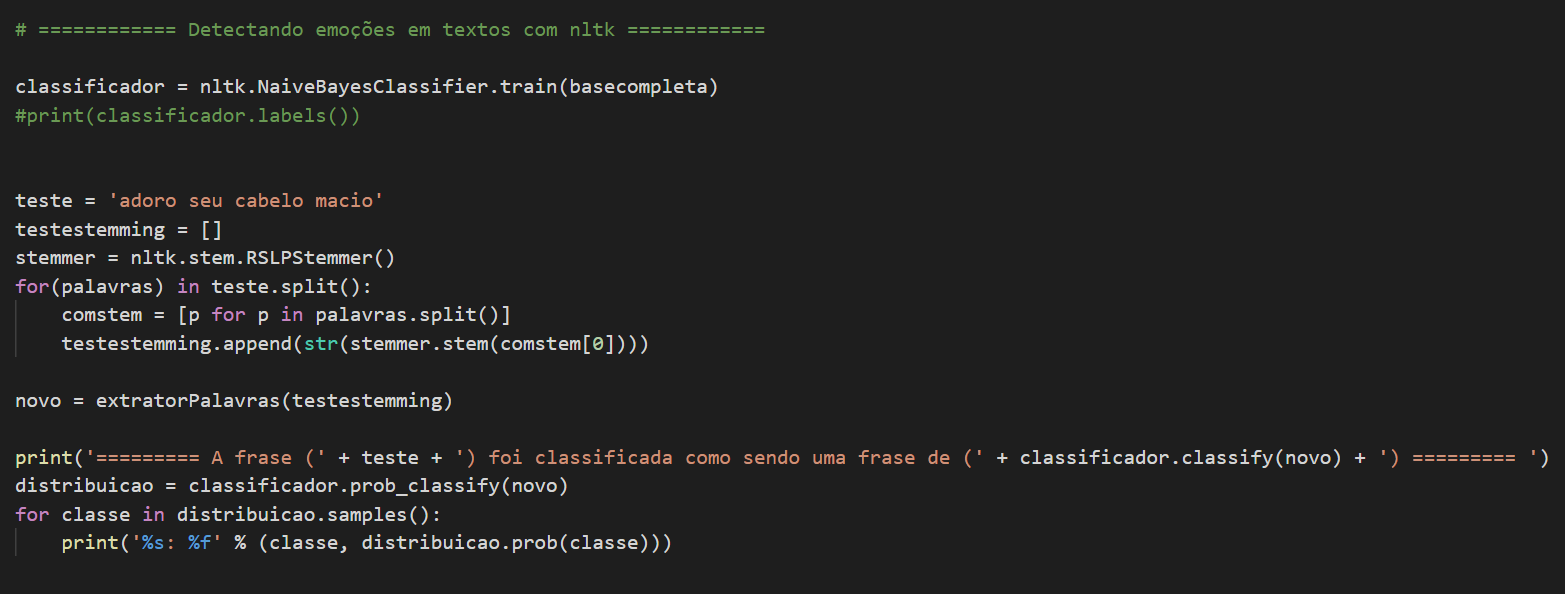


Agora que temos nossa lista de radicais iremos calcular a **frequência com que eles aparecem.**



Isso vai produzir um efeito de excluir as palavras repetidas.





Agora podemos classificar as emoções com base no conjunto que testamos.

Utilizando uma Api desenvolvida em Python com Django Framework, criamos apenas a funcionalidade do cadastrar, listar e classificar as avaliações com a através do código que desenvolvemos com a biblioteca NLTK.

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Passamos apenas os seguintes paramentros:

Descrição: Nesse campo guardamos a avaliação

Id\_author: Nesse campo guardamos o id de quem fez a avaliação.

Id\_artista: Nesse campo guardamos o id do artista avaliado.

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

A api classifica a emoção da avaliação e a grava no banco SQL lite!

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Avaliações já classificadas e guardadas no banco de dados

# 3 CONSIDERAÇÕES FINAIS

Agora que construímos uma API para guardar nosso algoritmo podemos classificá-las e, com isso, podemos fazer desde uma simples contagem do tipo das frases que coletamos até ver quantas são de alegria ou de raiva com o serviço prestado pelo artista.

Podemos também criar grupos do porquê as pessoas sentem raiva do nosso produto e fazer uma outra mineração para clusterizar as frases de raiva, atacando os problemas que são mais recorrentes.

A API desenvolvida tem como objetivo trabalhar em conjunto com a principal, desenvolvida em Java com banco My SQL, que é responsável pelas demais funcionalidades.

A ideia foi desmembrar a parte de mineração toda para p Python inclusive guardar as avaliações em outro banco de dados.

# 