```
# Instalar unidecode
!pip install unidecode
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: unidecode in /usr/local/lib/python3.9/dist-packages (1.3.6)
```

```
import nltk
nltk.download('punkt')
nltk.download('stopwords')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
True
```

```
import re
import numpy as np
from unidecode import unidecode
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
```

```
sentences = [
"A educação é o processo de facilitar o aprendizado",
"A educação é o processo de aquisição de conhecimentos, habilidades, valores, crenças e hábitos",
"A educação é uma prática social que visa o desenvolvimento do ser humano, de suas potencialidades, habilidades e competências",
"Bitcoin é uma criptomoeda descentralizada, sendo um dinheiro eletrônico para, transações ponto a ponto" ]

sw = nltk.corpus.stopwords.words('portuguese')

def text_clean(raw_text):
  # Remover acentos e padronizar em lower
  clean_text = unidecode(raw_text.lower())
  # Substituir quebras e tabulações por espaços
  clean_text = re.sub('[\n\t]', " ", clean_text)
  # Manter somente caraceteres alfanumericos
  clean_text = re.sub('[^ a-zA-Z]+', "", clean_text)
  # Remover stopwords e juntar palavras
  resultwords = [word for word in clean_text.split() if word not in sw]
  clean_sw_text = ' '.join(resultwords)
  return(clean_sw_text)

# Apresentar frases
print(sentences)
# Aplicar função no texto
clean_sentences = []
for frase in sentences:
  clean_sentences.append(text_clean(frase))

# Apresentar frases preprocessadas
print(clean_sentences)
```

```
# Transformar frases com vetorização
tfid_v = TfidfVectorizer()
features = tfid_v.fit_transform(clean_sentences)
# Apresentar palavras
print('Palavras: ', tfid_v.get_feature_names_out())
# Vetorizar Features (transposição)
vetPalavras = features.toarray().transpose()
# Apresentar vetor
print('Vetor:\n', vetPalavras)
```

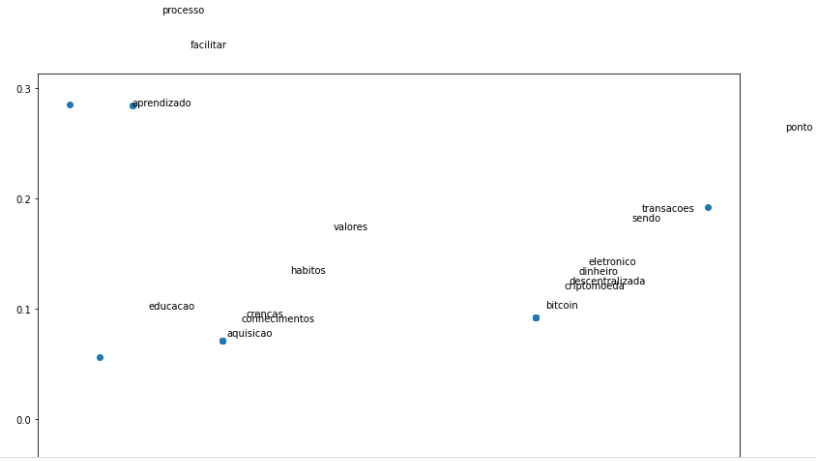
```
['A educação é o processo de facilitar o aprendizado', 'A educação é o processo de aquisição de conhecimentos, habilidades, valores
['educacao processo facilitar aprendizado', 'educacao processo aquisicao conhecimentos habilidades valores crencas habitos', 'educa
Palavras:  ['aprendizado' 'aquisicao' 'bitcoin' 'competencias' 'conhecimentos'
 'crencas' 'criptomoeda' 'descentralizada' 'desenvolvimento' 'dinheiro'
 'educacao' 'eletronico' 'facilitar' 'habilidades' 'habitos' 'humano'
 'ponto' 'potencialidades' 'pratica' 'processo' 'sendo' 'social'
 'transacoes' 'valores' 'visa']
Vetor:
 [[0.57457953 0.         0.         0.        ]
 [0.         0.387766   0.         0.        ]
 [0.         0.         0.         0.30151134]
 [0.         0.         0.35291425 0.        ]
 [0.         0.387766   0.         0.        ]
 [0.         0.387766   0.         0.        ]
 [0.         0.         0.         0.30151134]
```

```
      [0.         0.         0.         0.30151134]
      [0.         0.         0.35291425 0.        ]
      [0.         0.         0.         0.30151134]
      [0.36674667 0.24750601 0.22526059 0.        ]
      [0.         0.         0.         0.30151134]
      [0.57457953 0.         0.         0.        ]
      [0.         0.30571917 0.27824164 0.        ]
      [0.         0.387766   0.         0.        ]
      [0.         0.         0.35291425 0.        ]
      [0.         0.         0.         0.60302269]
      [0.         0.         0.35291425 0.        ]
      [0.         0.         0.35291425 0.        ]
      [0.4530051  0.30571917 0.         0.        ]
      [0.         0.         0.         0.30151134]
      [0.         0.         0.35291425 0.        ]
      [0.         0.         0.         0.30151134]
      [0.         0.387766   0.         0.        ]
      [0.         0.         0.35291425 0.        ]]
```

```
pca = PCA(2)
results = pca.fit_transform(vetPalavras)
print('PCA:\n', results)

plt.figure(figsize = (13, 13))
plt.scatter(results[:, 0], results[:, 1])
size = len(results[:, 0])
for i in range(size):
  pos = i*5
  plt.annotate(
      tfid_v.get_feature_names_out()[i],
      (results[i, 0], results[i, 1]),
      textcoords="offset points",
      xytext=(pos,pos), ha='left')
plt.show()
```

```
PCA:
 [[-0.2598563   0.28496302]
 [-0.1469151   0.07122359]
 [ 0.24667478  0.09252767]
 [-0.02103663 -0.27847188]
 [-0.1469151   0.07122359]
 [-0.1469151   0.07122359]
 [ 0.24667478  0.09252767]
 [ 0.24667478  0.09252767]
 [-0.02103663 -0.27847188]
 [ 0.24667478  0.09252767]
 [-0.30059681  0.05660509]
 [ 0.24667478  0.09252767]
 [-0.2598563   0.28496302]
 [-0.1497744  -0.15898291]
 [-0.1469151   0.07122359]
 [-0.02103663 -0.27847188]
 [ 0.46325483  0.19270763]
 [-0.02103663 -0.27847188]
 [-0.02103663 -0.27847188]
 [-0.33806256  0.28523564]
 [ 0.24667478  0.09252767]
 [-0.02103663 -0.27847188]
 [ 0.24667478  0.09252767]
 [-0.1469151   0.07122359]
 [-0.02103663 -0.27847188]]
```



ponto

✓ 1s conclusão: 15:27