Identificador automático de idioma

Problema: Dados um texto de entrada, é possível identificar em qual língua o texto está escrito?

Entrada: "texto qualquer"
Saída: português ou inglês ou francês ou italiano ou...

O processo de Reconhecimento de Padrões

O objetivo desse trabalho é demonstrar o processo de "construção de atributos" e como ele é fundamental para o **Reconhecimento de Padrões (RP)**.

Primeiro um conjunto de "amostras" previamente conhecido (classificado)

```
#
# amostras de texto em diferentes línguas
#
ingles = [
"Hello, how are you?",
"I love to read books.",
"The weather is nice today.",
"Where is the nearest restaurant?",
"What time is it?",
"I enjoy playing soccer.",
"Can you help me with this?",
"I'm going to the movies tonight.",
"This is a beautiful place.",
"I like listening to music.",
"Do you speak English?",
"What is your favorite color?",
"I'm learning to play the guitar.",
"Have a great day!",
"I need to buy some groceries.",
"Let's go for a walk.",
"How was your weekend?",
"I'm excited for the concert.",
"Could you pass me the salt, please?",
"I have a meeting at 2 PM.",
"I'm planning a vacation.",
"She sings beautifully.",
"The cat is sleeping.",
"I want to learn French.",
"I enjoy going to the beach.",
"Where can I find a taxi?",
"I'm sorry for the inconvenience.",
"I'm studying for my exams.",
```

```
"I like to cook dinner at home.",
"Do you have any recommendations for restaurants?",
]

espanhol = [
"Hola, ¿cómo estás?",
"Me encanta leer libros.",
"El clima está agradable hoy.",
"¿Dónde está el restaurante más cercano?",
"¿Qué hora es?",
"Voy al parque todos los días.",
"¿Puedes ayudarme con esto?",
"Me gustaría ir de vacaciones.",
"Este es mi libro favorito.",
"Me gusta bailar salsa.",
"¿Hablas español?",
"¿Cuál es tu comida favorita?",
"Estoy aprendiendo a tocar el piano.",
"¡Que tengas un buen día!",
"Necesito comprar algunas frutas.",
"Vamos a dar un paseo.",
"¿Cómo estuvo tu fin de semana?",
"Estoy emocionado por el concierto.",
"¿Me pasas la sal, por favor?",
"Tengo una reunión a las 2 PM.",
"Estoy planeando unas vacaciones.",
"Ella canta hermosamente.",
"El perro está jugando.",
"Quiero aprender italiano.",
"Disfruto ir a la playa.",
"¿Dónde puedo encontrar un taxi?",
"Lamento las molestias.",
"Estoy estudiando para mis exámenes.",
"Me gusta cocinar la cena en casa.",
"¿Tienes alguna recomendación de restaurantes?",
]

portugues = [
"Estou indo para o trabalho agora.",
"Adoro passar tempo com minha família.",
"Preciso comprar leite e pão.",
"Vamos ao cinema no sábado.",
"Gosto de praticar esportes ao ar livre.",
"O trânsito está terrível hoje.",
"A comida estava deliciosa!",
"Você já visitou o Rio de Janeiro?",
"Tenho uma reunião importante amanhã.",
"A festa começa às 20h.",
"Estou cansado depois de um longo dia de trabalho.",
"Vamos fazer um churrasco no final de semana.",
"O livro que estou lendo é muito interessante.",
"Estou aprendendo a cozinhar pratos novos.",
"Preciso fazer exercícios físicos regularmente.",
"Vou viajar para o exterior nas férias.",
```

```
"Voce gosta de dançar?",
"Hoje é meu aniversário!",
"Gosto de ouvir música clássica.",
"Estou estudando para o vestibular.",
"Meu time de futebol favorito ganhou o jogo.",
"Quero aprender a tocar violão.",
"Vamos fazer uma viagem de carro.",
"O parque fica cheio aos finais de semana.",
"O filme que assisti ontem foi ótimo.",
"Preciso resolver esse problema o mais rápido possível.",
"Adoro explorar novos lugares.",
"Vou visitar meus avós no domingo.",
"Estou ansioso para as férias de verão.",
"Gosto de fazer caminhadas na natureza.",
"O restaurante tem uma vista incrível.",
"Vamos sair para jantar no sábado.",
]
```

A "amostras" de texto precisa ser "transformada" em **padrões**

Um padrão é um conjunto de características, geralmente representado por um vetor e um conjunto de padrões no formato de tabela. Onde cada linha é um padrão e as colunas as características e, geralmente, na última coluna a **classe**

```
import random

pre_padroes = []
for frase in ingles:
  pre_padroes.append( [frase, 'inglês'])

for frase in espanhol:
  pre_padroes.append( [frase, 'espanhol'])

for frase in portugues:
  pre_padroes.append( [frase, 'português'])

random.shuffle(pre_padroes)
print(pre_padroes)
```

```
[['Estou aprendendo a cozinhar pratos novos.', 'português'], ['Vamos fazer
```

O DataFrame do pandas facilita a visualização.

```
import pandas as pd
dados = pd.DataFrame(pre_padroes)
dados
```

	0	1
0	Estou aprendendo a cozinhar pratos novos.	português
1	Vamos fazer um churrasco no final de semana.	português

2	Tenho uma reunião importante amanhã.	português
3	Me encanta leer libros.	espanhol
4	¿Qué hora es?	espanhol
...	...	...
87	Ella canta hermosamente.	espanhol
88	¿Me pasas la sal, por favor?	espanhol
89	O restaurante tem uma vista incrível.	português
90	What time is it?	inglês
91	Where can I find a taxi?	inglês

92 rows × 2 columns

Next steps: Generate code with dados View recommended plots

Construção dos atributos

Esse é o coração desse trabalho e que deverá ser desenvolvido por vocês. Pensem em como podemos "medir" cadas frase/sentença e extrair características que melhorem o resultado do processo de identificação.

Após a criação de cada novo atributo, execute as etapas seguintes e registre as métricas da matriz de confusão. Principalmente acurácia e a precisão.

```
# a entrada é o vetor pre_padroes e a saída desse passo deverá ser "padrões"
import re

def tamanhoMedioFrases(texto):
  palavras = re.split("\s",texto)
  #print(palavras)
  tamanhos = [len(s) for s in palavras if len(s)>0]
  #print(tamanhos)
  soma = 0
  for t in tamanhos:
    soma=soma+t
  return soma / len(tamanhos)


def contar_consoantes(texto):
   # retorna a quantidade de consoantes em uma frase
  padrao_consoantes = r'[bcdfghjklmnpqrstvwxyzBCDFGHJKLMNPQRSTVWXYZ]' #padrão re

  consoantes = re.findall(padrao_consoantes, texto)

  return len(consoantes)
```

```
def contar_acentuados(texto):

  #retorna o numero de caracteres com acentos gráficos
  padrao_acentos = r'[áéíóúàèìòùâêîôûãõäëïöüñçÁÉÍÓÚÀÈÌÒÙÂÊÎÔÛÃÕÄËÏÖÜÇÑ]'#padrão

  acentos = re.findall(padrao_acentos, texto)

  return len(acentos)

def contar_dupla_vogal(texto):
  padrao_duplas_vogais = r'(?i)[aeiouy]{2}'
  duplas_vogais = re.findall(padrao_duplas_vogais, texto)
  return len(duplas_vogais)

def contar_termino_palavras(texto):
  padrao_terminos = r'\b\w*(?:m|an|ón|ing)\b'
  terminos = re.findall(padrao_terminos, texto, re.IGNORECASE)
  return len(terminos)

def contar_pontuacao(texto):
  padrao_pontuacao = r'(|¿|!)'
  pontuacao = re.findall(padrao_pontuacao, texto)

  return len(pontuacao)

def caracteristicas_ingles(texto):
  padrao_ingles = r'(\'|wh|th|ing|oo|ee|nn)'
  pd_ingles = re.findall(padrao_ingles, texto)

  return len(pd_ingles)

def caracteristicas_espanhol(texto):
  padrao_espanhol = r'\w(?:¿|el|la|ue|ie|ón|an|oy|ay|ñ)'
  pd_espanhol = re.findall(padrao_espanhol, texto)

  return len(pd_espanhol)

def caracteristicas_portugues(texto):
  padrao_portugues = r'\w(?:nh|lh|rr|ss|am)'
  pd_portugues = re.findall(padrao_portugues, texto)

  return len(pd_portugues)

def extraiCaracteristicas(frase):
  # frase é um vetor [ 'texto', 'lingua' ]
  texto = frase[0]
  pattern_regex = re.compile('[^\w+]', re.UNICODE)
  texto = re.sub(pattern_regex,' ',texto)
  #print(texto)
  caracteristica1=tamanhoMedioFrases(texto)
  caracteristica2=contar_consoantes(texto)
  caracteristica3=contar_acentuados(texto)
  caracteristica4=caracteristicas_ingles(texto)
```

```
  caracteristica5=caracteristicas_espanhol(texto)
  caracteristica6=caracteristicas_portugues(texto)
  # acrescente as suas funcoes no vetor padrao
  padrao = [caracteristica1, caracteristica2, caracteristica3, caracteristica4,
  return padrao

def geraPadroes(frases):
  padroes = []
  for frase in frases:
    padrao = extraiCaracteristicas(frase)
    padroes.append(padrao)
  return padroes

# converte o formato [frase classe] em
# [caracteristica_1, caracteristica_2,... caracteristica n, classe]
padroes = geraPadroes(pre_padroes)

#
# apenas para visualizacao
print(padroes)

dados = pd.DataFrame(padroes)
dados
```

```
[[5.833333333333333, 20, 0, 0, 0, 1, 'português'], [4.5, 21, 0, 0, 1, 2, 'p
```

	0	1	2	3	4	5	6
0	5.833333	20	0	0	0	1	português
1	4.500000	21	0	0	1	2	português
2	6.200000	15	2	0	2	2	português
3	4.750000	11	0	1	1	0	espanhol
4	3.000000	4	1	0	0	0	espanhol
...	...	...	...	...	...	...	...
87	7.000000	12	0	0	2	1	espanhol
88	3.333333	12	0	0	0	0	espanhol
89	5.166667	17	1	0	2	0	português
90	3.000000	7	0	0	0	0	inglês
91	3.000000	10	0	0	1	0	inglês

92 rows × 7 columns

Next steps: Generate code with dados View recommended plots

Treinando o modelo com SVM

Separando o conjunto de treinamento do conjunto de testes

```
from sklearn.model_selection import train_test_split
import numpy as np

#from sklearn.metrics import confusion_matrix

vet = np.array(padroes)
classes = vet[:,-1]         # classes = [p[-1] for p in padroes]
#print(classes)
padroes_sem_classe = vet[:,0:-1]
#print(padroes_sem_classe)
X_train, X_test, y_train, y_test = train_test_split(padroes_sem_classe, classes,
```

Com os conjuntos separados, podemos "treinar" o modelo usando a SVM.

```
from sklearn import svm
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

treinador = svm.SVC()  #algoritmo escolhido
modelo = treinador.fit(X_train, y_train)

#
# score com os dados de treinamento
acuracia = modelo.score(X_train, y_train)
print("Acurácia nos dados de treinamento: {:.2f}%".format(acuracia * 100))

#
# melhor avaliar com a matriz de confusão
y_pred = modelo.predict(X_train)
cm = confusion_matrix(y_train, y_pred)
print(cm)
print(classification_report(y_train, y_pred))

#
# com dados de teste que não foram usados no treinamento
print('métricas mais confiáveis')
y_pred2 = modelo.predict(X_test)
cm = confusion_matrix(y_test, y_pred2)
print(cm)
print(classification_report(y_test, y_pred2))
```

```
Acurácia nos dados de treinamento: 71.01%
[[19  2  2]
 [ 5 14  3]
 [ 8  0 16]]
              precision    recall  f1-score   support

    espanhol       0.59      0.83      0.69        23
```

```
          inglês       0.88      0.64      0.74        22
      português       0.76      0.67      0.71        24

       accuracy                           0.71        69
      macro avg       0.74      0.71      0.71        69
   weighted avg       0.74      0.71      0.71        69

   métricas mais confiáveis
   [[4 1 2]
    [3 4 1]
    [3 0 5]]
                 precision    recall  f1-score   support

      espanhol       0.40      0.57      0.47         7
        inglês       0.80      0.50      0.62         8
      português       0.62      0.62      0.62         8

       accuracy                           0.57        23
      macro avg       0.61      0.57      0.57        23
   weighted avg       0.62      0.57      0.57        23
```

```
teste = r'\w(?:¿|el|la|ue|ie|ón|an|oy|ay|ñ)'
teste2 = r'(ñ|el|la|ón)'
valor = re.findall(teste2, "el hizo cumpleaños, cabrón")
print(valor)
```

```
['el', 'ñ', 'ón']
```