Projeto 2 - Classificador Automático de Sentimento

Você foi contratado por uma empresa parar analisar como os clientes estão reagindo a um determinado produto no Twitter. A empresa deseja que você crie um programa que irá analisar as mensagens disponíveis e classificará como "relevante" ou "irrelevante". Com isso ela deseja que mensagens negativas, que denigrem o nome do produto, ou que mereçam destaque, disparem um foco de atenção da área de marketing.

Como aluno de Ciência dos Dados, você lembrou do Teorema de Bayes, mais especificamente do Classificador Naive-Bayes, que é largamente utilizado em filtros anti-spam de e-mails. O classificador permite calcular qual a probabilidade de uma mensagem ser relevante dadas as palavras em seu conteúdo.

Para realizar o MVP (*minimum viable product*) do projeto, você precisa implementar uma versão do classificador que "aprende" o que é relevante com uma base de treinamento e compara a performance dos resultados com uma base de testes.

Após validado, o seu protótipo poderá também capturar e classificar automaticamente as mensagens da plataforma.

Informações do Projeto

Prazo: 13/Set até às 23:59.
Grupo: 1 ou 2 pessoas.

Entregáveis via GitHub:

- Arquivo notebook com o código do classificador, seguindo as orientações abaixo.
- Arquivo Excel com as bases de treinamento e teste totalmente classificado.

NÃO disponibilizar o arquivo com os *access keys/tokens* do Twitter.

Check 3:

Até o dia 06 de Setembro às 23:59, o notebook e o xlsx devem estar no Github com as seguintes evidências:

```
* Conta no twitter criada.
* Produto escolhido.
* Arquivo Excel contendo a base de treinamento e teste já classificado.
```

Sugestão de leitura:
http://docs.tweepy.org/en/v3.5.0/index.html (http://docs.tweepy.org/en/v3.5.0/index.html)
https://monkeylearn.com/blog/practical-explanation-naive-bayes-classifier/ (https://monkeylearn.com/blog/practical-explanation-naive-bayes-classifier/)

Preparando o ambiente

Instalando a biblioteca *tweepy* para realizar a conexão com o Twitter:

In [2]:

```
%%capture

#Instalando o tweepy
!pip install tweepy
```

Importando as Bibliotecas que serão utilizadas. Esteja livre para adicionar outras.

In [1]:

```
import tweepy
import math
import os.path
import pandas as pd
import json
import operator
from random import shuffle
import numpy as np
from itertools import product
from matplotlib_venn import venn2, venn2_circles
```

Autenticando no Twitter

Para realizar a captura dos dados é necessário ter uma conta cadastrada no twitter:

- Conta: ***https://twitter.com/vitorliu_55 (https://twitter.com/vitorliu_55)***

1. Caso ainda não tenha uma: https://twitter.com/signup (https://twitter.com/signup)
2. Depois é necessário registrar um app para usar a biblioteca: https://apps.twitter.com/ (https://apps.twitter.com/)
3. Dentro do registro do App, na aba Keys and Access Tokens, anotar os seguintes campos:
   A. Consumer Key (API Key)
   B. Consumer Secret (API Secret)
4. Mais abaixo, gere um Token e anote também:

   A. Access Token
   B. Access Token Secret
5. Preencha os valores no arquivo "auth.pass"

ATENÇÃO: Nunca divulgue os dados desse arquivo online (GitHub, etc). Ele contém as chaves necessárias para realizar as operações no twitter de forma automática e portanto é equivalente a ser "hackeado". De posse desses dados, pessoas mal intencionadas podem fazer todas as operações manuais (tweetar, seguir, bloquear/desbloquear, listar os seguidores, etc). Para efeito do projeto, esse arquivo não precisa ser entregue!!!

In [4]:

```
#Dados de autenticação do twitter:

#Coloque aqui o identificador da conta no twitter: @fulano

#leitura do arquivo no formato JSON
with open('auth.pass') as fp:
    data = json.load(fp)

#Configurando a biblioteca. Não modificar
auth = tweepy.OAuthHandler(data['consumer_key'], data['consumer_secret'])
auth.set_access_token(data['access_token'], data['access_token_secret'])
```

Coletando Dados

Agora vamos coletar os dados. Tenha em mente que dependendo do produto escolhido, não haverá uma quantidade significativa de mensagens, ou ainda poder haver muitos retweets.

Configurando:

In [8]:

```
#Produto escolhido:
produto = 'Narcos'

#Quantidade mínima de mensagens capturadas:
n = 500
#Quantidade mínima de mensagens para a base de treinamento:
t = 300

#Filtro de língua, escolha uma na tabela ISO 639-1.
lang = 'pt'
```

Capturando os dados do twitter:

In [9]:

```
#Cria um objeto para a captura
api = tweepy.API(auth)

#Inicia a captura, para mais detalhes: ver a documentação do tweepy
i = 1
msgs = []
for msg in tweepy.Cursor(api.search, q=produto, lang=lang).items():
    msgs.append(msg.text.lower())
    i += 1
    if i > n:
        break

#Embaralhando as mensagens para reduzir um possível viés
shuffle(msgs)
```

Salvando os dados em uma planilha Excel:

In [10]:

```
#Verifica se o arquivo não existe para não substituir um conjunto pronto
if not os.path.isfile('./{0}.xlsx'.format(produto)):

    #Abre o arquivo para escrita
    writer = pd.ExcelWriter('{0}.xlsx'.format(produto))

    #divide o conjunto de mensagens em duas planilhas
    dft = pd.DataFrame({'Treinamento' : pd.Series(msgs[:t])})
    dft.to_excel(excel_writer = writer, sheet_name = 'Treinamento', index = False)

    dfc = pd.DataFrame({'Teste' : pd.Series(msgs[t:])})
    dfc.to_excel(excel_writer = writer, sheet_name = 'Teste', index = False)

    #fecha o arquivo
    writer.save()
```

Classificando as Mensagens

Agora você deve abrir o arquivo Excel com as mensagens capturadas e classificar na Coluna B se a mensagem é relevante ou não.
Não se esqueça de colocar um nome para a coluna na célula **B1**.

Fazer o mesmo na planilha de Controle.

Montando o Classificador Naive-Bayes

Com a base de treinamento montada, comece a desenvolver o classificador. Escreva o seu código abaixo:

Opcionalmente:

- Limpar as mensagens removendo os caracteres: enter, :, ", ', (,), etc. Não remover emojis.
- Corrigir separação de espaços entre palavras e/ou emojis.
- Propor outras limpezas/transformações que não afetem a qualidade da informação.

In [114]:

```
#quanto a classificação da tabela de treinamento, optamos por utilizar as siglas "b" e "n"
#para a análise.

df = pd.DataFrame(pd.read_excel("Narcos.xlsx"))

#remoção dos caracteres indesejados:
df.Treinamento = df.Treinamento.replace('-','')
df.Treinamento = df.Treinamento.replace('+','')
df.Treinamento = df.Treinamento.replace('/','')
df.Treinamento = df.Treinamento.replace('_','')
df.Treinamento = df.Treinamento.replace('.','')
df.Treinamento = df.Treinamento.replace(',','')
df.Treinamento = df.Treinamento.replace('  ','')
df.Treinamento = df.Treinamento.replace('!','')
df.Treinamento = df.Treinamento.replace('@','')
df.Treinamento = df.Treinamento.replace('#','')
df.Treinamento = df.Treinamento.replace('$','')
df.Treinamento = df.Treinamento.replace('%','')
df.Treinamento = df.Treinamento.replace('^','')
df.Treinamento = df.Treinamento.replace('&','')
df.Treinamento = df.Treinamento.replace('*','')
df.Treinamento = df.Treinamento.replace('(','')
df.Treinamento = df.Treinamento.replace(')','')
df.Treinamento = df.Treinamento.replace(':','')
df.Treinamento = df.Treinamento.replace(';','')
df.Treinamento = df.Treinamento.replace('\n','')

#separação dos tweets classificados como bons e neutros
bom = df[df.Classificacao == 'b']
neutro = df[df.Classificacao == 'n']

#lista das palavras de todos os tweets:
l_bom = np.sum(bom.Treinamento+'').split()
l_neutro = np.sum(neutro.Treinamento+'').split()

#número de palavras relevantes e irrelevantes:
n_bom = len(l_bom)
n_neutro = len(l_neutro)

#função que aplica o teorema de bayes a um tweet específico
def B(tweet):
    tweet = tweet.split(" ")
    #probabilidade de relevância:
    P_bom = []
    P_neutro = []

    for palavra in range(len(tweet)):
        P_bom.append(l_bom.count((tweet[palavra]))/(n_bom))
        P_neutro.append(l_neutro.count((tweet[palavra]))/(n_neutro))

    total_bom = np.sum(P_bom)
    total_neutro = np.sum(P_neutro)

    if total_bom > total_neutro:
        return "b"
    else:
        return "n"
```

Verificando a performance

Agora você deve testar o seu Classificador com a base de Testes.

Você deve extrair as seguintes medidas:

- Porcentagem de positivos falsos (marcados como relevante mas não são relevantes)
- Porcentagem de positivos verdadeiros (marcado como relevante e são relevantes)
- Porcentagem de negativos verdadeiros (marcado como não relevante e não são relevantes)
- Porcentagem de negativos falsos (marcado como não relevante e são relevantes)

Opcionalmente:

- Criar categorias intermediárias de relevância baseado na diferença de probabilidades. Exemplo: muito relevante, relevante, neutro, irrelevante e muito irrelevante.

In [131]:

```
df = pd.read_excel('Narcos.xlsx', sheetname = 'Teste')
df['Bayes'] = pd.Series()

for i in range(df.shape[0]):
    bayes = B(df.Teste[i])
    df.Bayes[i] = bayes

df.tail()
```

```
C:\Users\jorge\Anaconda3\lib\site-packages\ipykernel\__main__.py:6: SettingW
ithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/s
table/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pand
as-docs/stable/indexing.html#indexing-view-versus-copy)
C:\Users\jorge\Anaconda3\lib\site-packages\pandas\core\indexing.py:141: Sett
ingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/s
table/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pand
as-docs/stable/indexing.html#indexing-view-versus-copy)
  self._setitem_with_indexer(indexer, value)
```

Out[131]:

	Teste	Classificacao	Bayes
195	vamo #masterchefbr quero assistir narcos jdndjd	n	b
196	vou começar a última temporada de narcos agora	b	b
197	rt @mqdesigns13: playeras de #pabloescobar #pl...	n	n
198	tô quase terminado narcos mas não quero que acabe	b	b
199	focado no objetivo da noite\n\nterminar a temp...	b	b

In [158]:

```
lista = []

for i in range(len(df.Teste)):
    if df.Classificacao[i] == 'b':
        if df.Bayes[i] == 'b':
            lista.append(1)
        if df.Bayes[i] == 'n':
            lista.append(2)
    elif df.Classificacao[i] == 'n':
        if df.Bayes[i] == 'b':
            lista.append(3)
        if df.Bayes[i] == 'n':
            lista.append(4)

#número de "bons" e "neutros" de acordo com a classificação manual dos tweets
b = 120
n = 80

VB = lista.count(1)
FB = lista.count(3)
VN = lista.count(4)
FN = lista.count(2)

print('porcentagem de Verdadeiros Bons: {}'.format(VB/2))
print('porcentagem de Falsos Bons: {}'.format(FB/2))
print('porcentagens de Verdadeiros Neutros: {}'.format(VN/2))
print('porcentagem de Falsos Neutros: {}'.format(FN/2))
```

```
porcentagem de Verdadeiros Bons: 57.5
porcentagem de Falsos Bons: 29.5
porcentagens de Verdadeiros Neutros: 10.5
porcentagem de Falsos Neutros: 2.5
```

Concluindo

Escreva aqui a sua conclusão.
Faça um comparativo qualitativo sobre as medidas obtidas.
Explique como são tratadas as mensagens com dupla negação e sarcasmo.
Proponha um plano de expansão. Por que eles devem continuar financiando o seu projeto?

Opcionalmente:

- Discorrer por que não posso alimentar minha base de Treinamento automaticamente usando o próprio classificador, aplicado a novos tweets.
- Propor diferentes cenários de uso para o classificador Naive-Bayes. Cenários sem intersecção com este projeto.
- Sugerir e explicar melhorias reais no classificador com indicações concretas de como implementar (não é preciso codificar, mas indicar como fazer e material de pesquisa sobre o assunto).

```
Feita a verificação de performance, pode-se dizer que o analisador de sentimentos
 em questão é eficiente, na medida que condiz com a classificação de teste com cer
```

ca de 68% de precisão. Isso comprova que o modelo proposto por Bayes é eficaz quando se trata da análise do sentimento que uma palavra isolada carrega dentre suas diversas aplicações. Porém, o mesmo apresenta erros de precisão (falsos positivos e negativos) que prejudicam a análise e são decorrentes da enorme complexidade que compõe a linguagem, assim como casos de ironia no qual uma palavra considerada positiva é usada com sentido oposto, ou mesmo abreviações derivadas do meio da coleta dos dados. Quanto a mensagens com dupla negação e sarcasmo, o classificador não é capaz de tratá-las como incidentes isolados, pois sua maneira de comparação depende da maioria das vezes em que essa palavra foi empregada e acaba por atribuir um valor incorreto à relevância de certas palavras que compõem o tweet, dificultando sua interpretação.

Como expansão do projeto, seria interessante adicionar um incremento no tamanho da base de dados a precisão tende a aumentar, sendo que o modelo é baseado puramente em probabilidades calculadas pelo Teorema de Bayes e a maior amostragem significaria uma projeção mais precisa da chance de certa palavra ser considerada boa, neutra ou negativa para a análise do tweet como um todo.