FACULDADE IMPACTA DE TECNOLOGIA – FIT

CURSO: SISTEMAS DE INFORMAÇÃO – SI

DISCIPLINA: COMPUTAÇÃO COGNITIVA – CG

ATIVIDADE CONTÍNUA 4 – AC4

Projeto Disciplina – Grupo **#GetPubli**

Text minning e análise de sentimentos em mídias sociais

EQUIPE

Danilo Lopes do Nascimento - 1700255

Lucas Fernandes Soares de Almeida - 1700195

Matheus Pereira Silva- 1700688

Nayara de Paula Muniz - 1700239

Vítor Crepaldi Carlessi - 1700266

SÃO PAULO

05 de novembro de 2020

SUMÁRIO

ENTENDIMENTO DO NEGÓCIO

Devido a pandemia do corona vírus em 2020, um casal de imigrantes ingleses que moram no Brasil tiveram seu salário reduzido pela metade, para conseguir complementar renda, resolveram lançar um curso extensivo em certificações da língua inglesa para estudantes brasileiros chamado #EasyCertification, pois são nativo no idioma e também já trabalharam em escolas de estudo da língua inglesa no Brasil. O problema encontrado pelo casal foi justamente em qual certificação focar, ficaram em dúvida em basicamente dois exames: IELTS (requisito acadêmico, corporativo e que pode ser aceito para fins de imigração. É reconhecido principalmente, por universidades no Reino Unido, Austrália, Canadá e Nova Zelândia. Inclui redação, interpretação de texto, compreensão auditiva e expressão oral) e TOEFL(permite avaliar as competências em língua inglesa principalmente em contextos acadêmicos), o casal ficou em dúvida pois viu que os dois se equivalem em vários quesitos e para critério de desempate acharam relevante consultar qual está sendo melhor falado nas mídias sociais, principalmente na plataforma do Twitter. Com isso contrataram a #GetPubli, consultoria especializada em mineração de texto e análise de sentimentos em redes sociais para conseguir ajudar a empresa #EasyCertification nessa análise.

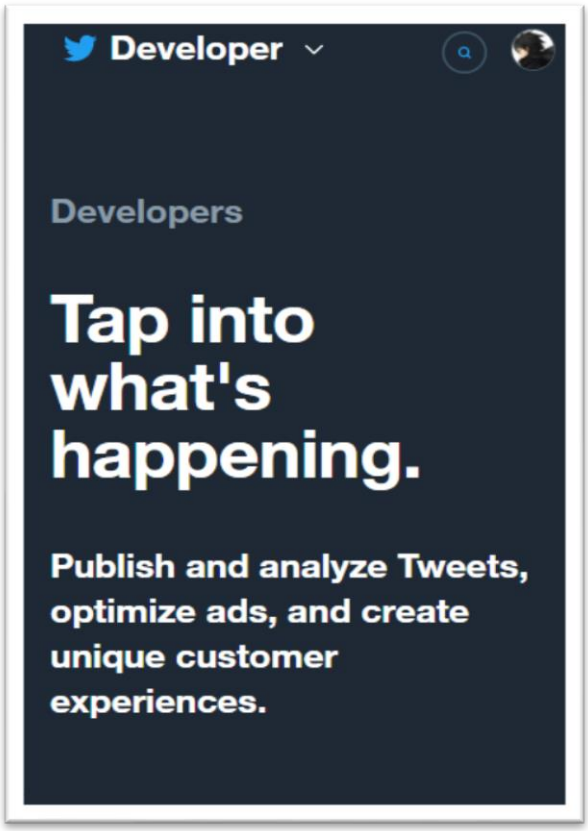
ENTENDIMENTO DOS DADOS

A base de dados será a rede social twitter. Por intermédio do developers twitter, foi criado um APP para consumo da API do twitter em código python e pela utilização da biblioteca tweepy é possível realizar o consumo dessa API para trazer postagens dos usuários.

Iremos coletar 900 tweets, sendo 450 para cada arquivo que será gerado, essa limitação é decorrente da licença registrada na API que indica que seu consumo é apenas de fins acadêmicos.

Quanto ao tipo de dados, o que recebemos é exatamente o tweet que foi publicado, pode ser entendido como uma string.

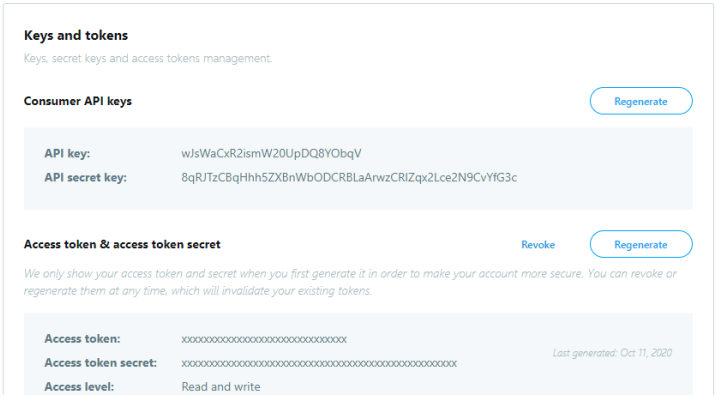
A nível de conhecimento: a query de busca no twitter, quando bem sucedida, retorna um dicionário com diversas informações. O que coletamos foi apenas o usuário e seu tweet, porém diversas outras informações retornam nesse dicionário.



[1] < site inicial da https://developer.twitter.com/en >

PREPARAÇÃO DOS DADOS

Para obtenção dos dados, foi utilizado a API do Twitter. Inicialmente foi criado um APP no developer.twitter . Para consumo da APP criada, foi utilizada a biblíoteca tweepy.



<APP criado no https://developer.twitter.com/en >

Inicialmente, fizemos a autenticação com a API do Twitter passando quatro chaves: consumerKey, consumerSecret, accessToken e accessTokenSecret.

```
consumerKey        = "wJsWaCxR2ismW20UpDQ8YObqV"
consumerSecret     = "8qRJTzCBqHhh5ZXBnWbODCRBLaArwzCRlZqx2Lce2N9CvYfG3c"
accessToken        = "1106181670821154817-VigfLv5XYLS3ZTR9WSFbmiwFwQBD9Q"
accessTokenSecret  = "dhdkuYTBPiElaM6bVz7qpoRiHmhNFSAjxGYWsUtjHlqgL"
```

```
auth = tweepy.OAuthHandler(consumer_key=consumerKey, consumer_secret=consumerSecret)
auth.set_access_token(accessToken, accessTokenSecret)
api = tweepy.API(auth)
```

<Variáveis contendo os valores das keys que estão registradas no APP do Twitter e autenticação na API>

Com a autenticação feita, o próximo passo foi montar uma query de busca no Twitter, onde pegamos os últimos 450 tweets com a hashtag #TOEFL no idioma inglês. Dentro do dicionário que nos retornou, fizemos uma lógica para percorrer os resultados e montar um arquivo CSV com o nome do usuário que fez o tweet e o conteúdo dele. Posteriormente fizemos outra query de busca, agora com a hashtag #IELTS, também pegando os últimos 450 tweets com essa hashtag no idioma inglês. Com a segunda query de busca, montamos também outro CSV com os resultados encontrados em #IELTS.

É importante ressaltar que colocamos um parâmetro no código para não pegar o mesmo retweet mais de uma vez.

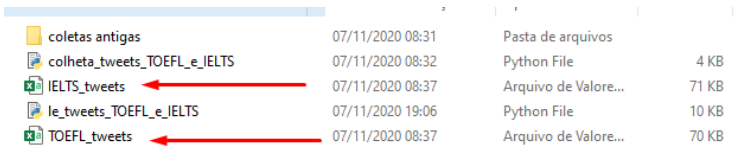
```python
#Termo de busca será no twitter: #TOEFL
searchTerm = "#TOEFL" + "-fielter:retweets" ##Filtrando os reetwets -> não pegar o mesmo reeetweet mais de uma vez

#Tweepy.cursor irá buscar no twitter os últimos 450 twittes com a palavra #TOEFL
tweets = tweepy.Cursor(api.search, q=searchTerm, show_user = True, lang='en').items(450)

#Criação da lista de tweets
listTweets = []

#appendando a primeira linha que saíra no .CSV, com a coluna Usuário e Tweet
row_list = [["Usuario","Tweet"]]

#percorrendo todos os tweets encontrados
for tweet in tweets:
    #appendando o nome do usuário e o tweet feito pelo mesmo na lista
    listTweets.append(list((tweet.user.screen_name, tweet.text)))

for x in listTweets:
    row_list.append(x)

#Codificações para escrever a saida no arquivo .CSV resultados.csv
csv.register_dialect('myDialect',
                     delimiter=',',
                     quoting=csv.QUOTE_ALL)
with open('TOEFL_tweets.csv', 'w', encoding='utf-8-sig', newline='') as file:
    writer = csv.writer(file, dialect='myDialect')
    writer.writerows(row_list)
```

< Query de busca de #TOEFL no twitter e lógica para montar o CSV TOEFL_tweets >

```python
#Termo de busca será no twitter será #IELTS
searchTerm = "#IELTS" + "-fielter:retweets" ##Filtrando os reetwets -> não pegar o mesmo reeetweet mais de uma vez

#Tweepy.cursor irá buscar no twitter os últimos 450 twittes com a palavra #IELTS
tweets = tweepy.Cursor(api.search, q=searchTerm, show_user = True, lang='en').items(450)

#Criação da lista de tweets
listTweets = []

#appendando a primeira linha que saíra no .CSV, com a coluna Usuário e Tweet
row_list = [["Usuario","Tweet"]]

#percorrendo todos os tweets encontrados
for tweet in tweets:
    #appendando o nome do usuário e o tweet feito pelo mesmo na lista
    listTweets.append(list((tweet.user.screen_name, tweet.text)))

for x in listTweets:
    row_list.append(x)

#Codificações para escrever a saida no arquivo .CSV IELTS_tweets.csv
csv.register_dialect('myDialect',
                     delimiter=',',
                     quoting=csv.QUOTE_ALL)
with open('IELTS_tweets.csv', 'w', encoding='utf-8-sig', newline='') as file:
    writer = csv.writer(file, dialect='myDialect')
    writer.writerows(row_list)

#Alerta que os arquivos foram gerados com sucesso
```

< Query de busca de #IELTS no twitter e lógica para montar o CSV IELTS_tweets>

Ao término da etapa de preparação dos dados, teremos então os arquivos CSV IELTS_tweets e TOEFL_tweets, cada um com 450 tweets.



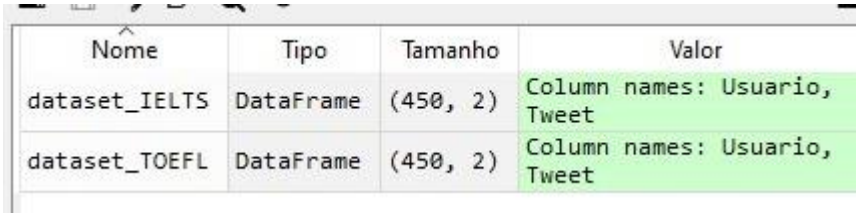
Obs. Existe a pasta coletas antigas, que tem justamente a coleta desses dois csv's em datas antigas, quando estávamos desenvolvendo o projeto. E possível trocar esses csv's pelos antigos e ter uma noção de como estavam sendo classificados naquela época (mês de outubro).

MODELAGEM

Agora em outro arquivo .py, fazemos o upload dos dois csvs.

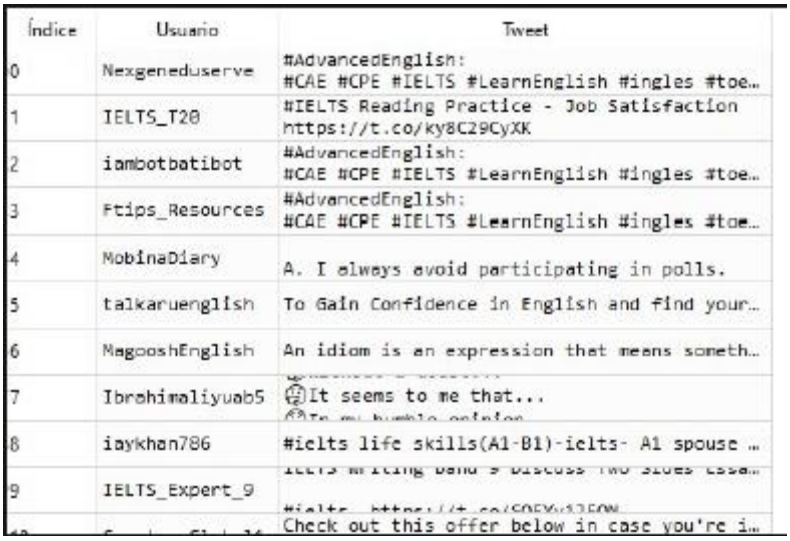
```
## Carregando os dados dos arquivos gerados TOEFL_tweets.csv e IELTS_tweets.csv
##Arquivo TOEFL_tweets.csv
dataset_TOEFL = pd.read_csv('TOEFL_tweets.csv')

##Arquivo IELTS_tweets.csv
dataset_IELTS = pd.read_csv('IELTS_tweets.csv')
```

< Fazendo o upload dos CSVs >



Nome	Tipo	Tamanho	Valor
dataset_IELTS	DataFrame	(450, 2)	Column names: Usuario, Tweet
dataset_TOEFL	DataFrame	(450, 2)	Column names: Usuario, Tweet

< upload dos Dataset's feito com sucesso no explorador de variáveis do spyder >



Índice	Usuario	Tweet
0	Nexgeneduserve	#AdvancedEnglish: #CAE #CPE #IELTS #LearnEnglish #ingles #toe…
1	IELTS_T20	#IELTS Reading Practice - Job Satisfaction https://t.co/ky8C29CyXK
2	iambotbatibot	#AdvancedEnglish: #CAE #CPE #IELTS #LearnEnglish #ingles #toe…
3	Ftips_Resources	#AdvancedEnglish: #CAE #CPE #IELTS #LearnEnglish #ingles #toe…
4	MobinaDiary	A. I always avoid participating in polls.
5	talkaruenglish	To Gain Confidence in English and find your…
6	MagooshEnglish	An idiom is an expression that means someth…
7	Ibrahimaliyuab5	It seems to me that...
8	iaykhan786	#ielts life skills(A1-B1)-ielts- A1 spouse …
9	IELTS_Expert_9	
		Check out this offer below in case you're i…

< Exemplo de como ficou o dataset no spyder com a coluna usuário e o seu tweet >

Foi inserido a coluna Sentimento nos dataset's, ela recebeu o valor de não classificado por enquanto, vai receber o valor de neutro, positivo ou negativo decorrente da sua classificação pela polaridade. Também foi inserido a coluna Subjetividade, que irá receber o valor de objetivo, subjetivo ou neutro decorrente da sua classificação pela subjetividade. Pela utilização da biblioteca TextBlob conseguimos extrair dos tweets, valores de polaridade e subjetividade.

```
##Coluna sentimento no dataset_TOEFL
dataset_TOEFL['Sentimento']    = 'Não classificado'

##Coluna sentimento no dataset_IELTS
dataset_IELTS['Sentimento']    = 'Não classificado'

##Coluna subjetividade no dataset_TOEFL
dataset_TOEFL['Subjetividade'] = 'Não classificado'

##Coluna subjetividade no dataset_IELTS
dataset_IELTS['Subjetividade'] = 'Não classificado'
```

<Inserção de novas colunas no Dataset>

Percorremos os dois dataset's pela utilização do comando "*for*" classificando as colunas anteriormente criadas, de acordo com sua polaridade e subjetividade. Quanto a polaridade, de acordo com a documentação, os valores obtidos podem estar no range de -1 até 1, considerando -1 como sentimento negativo e 1 como positivo.

```
## Lógica para avaliar o sentimento como Positivo, Neutro ou Negativo no arquivo TOEFL_tweets
for index, row in dataset_TOEFL.iterrows():
    analysis        =  TextBlob(row['Tweet'])
    #Classificação de Polaridade -> TOEFL
    if (analysis.sentiment.polarity == 0):
        dataset_TOEFL.loc[index, 'Sentimento'] = 'Neutro'
        neutral_TOEFL = neutral_TOEFL + 1
    elif (analysis.sentiment.polarity < 0):
        dataset_TOEFL.loc[index, 'Sentimento'] = 'Negativo'
        negative_TOEFL = negative_TOEFL + 1
    elif (analysis.sentiment.polarity > 0):
        dataset_TOEFL.loc[index, 'Sentimento'] = 'Positivo'
        positive_TOEFL = positive_TOEFL + 1
```

< Lógica para classificar o Sentimento com base na polaridade no dataset TOEFL >

```
## Lógica para avaliar o sentimento como Positivo, Neutro ou Negativo no arquivo IELTS_tweets
for index, row in dataset_IELTS.iterrows():
    analysis        =  TextBlob(row['Tweet'])
    #Classificação de Polaridade -> IELTS
    if (analysis.sentiment.polarity == 0):
        dataset_IELTS.loc[index, 'Sentimento'] = 'Neutro'
        neutral_IELTS = neutral_IELTS + 1
    elif (analysis.sentiment.polarity < 0):
        dataset_IELTS.loc[index, 'Sentimento'] = 'Negativo'
        negative_IELTS = negative_IELTS + 1
    elif (analysis.sentiment.polarity > 0):
        dataset_IELTS.loc[index, 'Sentimento'] = 'Positivo'
        positive_IELTS = positive_IELTS + 1
```

< Lógica para classificar o Sentimento com base na polaridade no dataset IELTS >

Quanto a subjetividade, também com o que é mostrado na documentação. Os valores obtidos podem estar no range 0 até 1, considerando 0 objetivo e 1 subjetivo.

```
## Lógica para avaliar a subjetividade como Neutro, Objetivo ou Subjetivo no arquivo IELTS_tweets
#Classificação de Subjetividade -> IELTS
if     (analysis.sentiment.subjectivity == 0.5):
    dataset_IELTS.loc[index, 'Subjetividade'] = 'Neutro'
    sub_neutral_IELTS = sub_neutral_IELTS + 1
elif   (analysis.sentiment.subjectivity < 0.5):
    dataset_IELTS.loc[index, 'Subjetividade'] = 'Objetivo'
    sub_objective_IELTS = sub_objective_IELTS + 1
elif (analysis.sentiment.subjectivity > 0.5):
    dataset_IELTS.loc[index, 'Subjetividade'] = 'Subjetivo'
    sub_subjectivity_IELTS = sub_subjectivity_IELTS + 1
```

<Lógica para classificar a Subjetividade, no dataset IELTS>

```
## Lógica para avaliar a subjetividade como Neutro, Objetivo ou Subjetivo no arquivo TOEFL_tweets
#Classificação de Subjetividade -> TOEFL
if     (analysis.sentiment.subjectivity == 0.5):
    dataset_TOEFL.loc[index, 'Subjetividade'] = 'Neutro'
    sub_neutral_TOEFL = sub_neutral_TOEFL + 1
elif   (analysis.sentiment.subjectivity < 0.5):
    dataset_TOEFL.loc[index, 'Subjetividade'] = 'Objetivo'
    sub_objective_TOEFL = sub_objective_TOEFL + 1
elif (analysis.sentiment.subjectivity > 0.5):
    dataset_TOEFL.loc[index, 'Subjetividade'] = 'Subjetivo'
    sub_subjectivity_TOEFL = sub_subjectivity_TOEFL + 1
```

<Lógica para classificar a Subjetividade, no dataset TOEFL >

Ao término da classificação da polaridade e subjetividade nas novas colunas de cada dataset, existe uma lógica para a classificação de qual prova é melhor conceituada no twitter. Ela leva em consideração apenas a polaridade, como temos 450 tweets em cada csv, podemos dizer que o melhor exame é o que tem a maior quantidade de tweets positivos, caso haja empate nesse quesito, escolhemos o que teve menos tweets negativos. É importante ressaltar que as variáveis positive_TOEFL, negative_TOEFL e neutral_TOEFL armazenaram a quantidade de valores positivos, negativos e neutros que foram classificados na coluna Sentimento do Dataset. O mesmo aconteceu com IELTS, com as variáveis positive_IELTS, negative_IELTS e neutral_IELTS. Então para comparação da melhor opção de prova, basta comparação os valores que foram obtidos nessas variáveis.

```
##Lógica para comparação da melhor opção
if (positive_IELTS == positive_TOEFL) and (negative_IELTS < negative_TOEFL):
    melhor_opcao = 'IELTS'

if (positive_IELTS == positive_TOEFL) and (negative_TOEFL < negative_IELTS):
    melhor_opcao = 'TOEFL'


if (positive_IELTS > positive_TOEFL):
    melhor_opcao = 'IELTS'
else:
    melhor_opcao = 'TOEFL'
```

< Comparação da prova mais conceituada **>**

A prova mais conceituada é mostrada em tela ao usuário, assim como seus valores de Polaridade e Subjetividade, a segunda prova também é apresentada, juntamente também aos seus valores.

```python
##Exibição dos valores encontrados
##Exame IELTS
print("----------------------Exame IELTS----------------------")
print('Quanto a polaridade: ')
print('O número de tweets positivos  em IELTS_tweets.csv é: ', positive_IELTS)
print('O número de tweets negativos  em IELTS_tweets.csv é: ', negative_IELTS)
print('O número de tweets neutros    em IELTS_tweets.csv é: ', neutral_IELTS)
print('Quanto a subjetividade: ')
print('O número de tweets subjetivos em IELTS_tweets.csv é: ', sub_subjectivity_IELTS)
print('O número de tweets objetivos  em IELTS_tweets.csv é: ', sub_objective_IELTS)
print('O número de tweets neutros    em IELTS_tweets.csv é: ', sub_neutral_IELTS)
print("----------------------Exame IELTS----------------------")

##Exame TOEFL
print("----------------------Exame TOEFL----------------------")
print('Quanto a polaridade: ')
print('O número de tweets positivos  em TOEFL_tweets.csv é: ', positive_TOEFL)
print('O número de tweets negativos  em TOEFL_tweets.csv é: ', negative_TOEFL)
print('O número de tweets neutros    em TOEFL_tweets.csv é: ', neutral_TOEFL)
print('Quanto a subjetividade: ')
print('O número de tweets subjetivos em TOEFL_tweets.csv é: ', sub_subjectivity_TOEFL)
print('O número de tweets objetivos  em TOEFL_tweets.csv é: ', sub_objective_TOEFL)
print('O número de tweets neutros    em TOEFL_tweets.csv é: ', sub_neutral_TOEFL)
print("----------------------Exame TOEFL----------------------")
```

<Exibição dos valores encontrados – Código Python>

```
----------------------Exame IELTS----------------------
Quanto a polaridade:
O número de tweets positivos  em IELTS_tweets.csv é:  185
O número de tweets negativos  em IELTS_tweets.csv é:  35
O número de tweets neutros    em IELTS_tweets.csv é:  230
Quanto a subjetividade:
O número de tweets subjetivos em IELTS_tweets.csv é:  83
O número de tweets objetivos  em IELTS_tweets.csv é:  337
O número de tweets neutros    em IELTS_tweets.csv é:  30
----------------------Exame IELTS----------------------
----------------------Exame TOEFL----------------------
Quanto a polaridade:
O número de tweets positivos  em TOEFL_tweets.csv é:  138
O número de tweets negativos  em TOEFL_tweets.csv é:  48
O número de tweets neutros    em TOEFL_tweets.csv é:  264
Quanto a subjetividade:
O número de tweets subjetivos em TOEFL_tweets.csv é:  67
O número de tweets objetivos  em TOEFL_tweets.csv é:  350
O número de tweets neutros    em TOEFL_tweets.csv é:  33
----------------------Exame TOEFL----------------------
```

<Exibição dos valores encontrados – Saída pela utilização do comando PRINT>

```
##Resultado da comparação
print("---------------------Resultado-----------------------")
if   melhor_opcao == 'IELTS':
    print("O Exame IELTS é o melhor conceituado no Twitter!!",
          "\nTem", positive_IELTS, "tweets positivos,", negative_IELTS, "tweets negativos", "e", neutral_IELTS, "tweets neutr
          "\nQuanto a subjetividade, tem ", sub_subjectivity_IELTS, "tweets subjetivos,", sub_objective_IELTS, "tweets objeti
          "\nO segundo colocado é o exame TOEFL",
          "\nTem", positive_TOEFL, "tweets positivos,", negative_TOEFL, "tweets negativos", "e", neutral_TOEFL, "tweets neutr
          "\nQuanto a subjetividade, tem ", sub_subjectivity_TOEFL, "tweets subjetivos,", sub_objective_TOEFL, "tweets objeti
          ),
elif melhor_opcao == 'TOEFL':
    print("O Exame TOEFL é o melhor conceituado no Twitter!!",
          "\nTem", positive_TOEFL, "tweets positivos,", negative_TOEFL, "tweets negativos", "e", neutral_TOEFL, "tweets neutr
          "\nQuanto a subjetividade, tem ", sub_subjectivity_TOEFL, "tweets subjetivos,", sub_objective_TOEFL, "tweets objeti
          "\nO segundo colocado é o exame IELTS",
          "\nTem", positive_IELTS, "tweets positivos,", negative_IELTS, "tweets negativos", "e", neutral_IELTS, "tweets neutr
          "\nQuanto a subjetividade, tem ", sub_subjectivity_IELTS, "tweets subjetivos,", sub_objective_IELTS, "tweets objeti
          )
print("---------------------Resultado-----------------------")
```

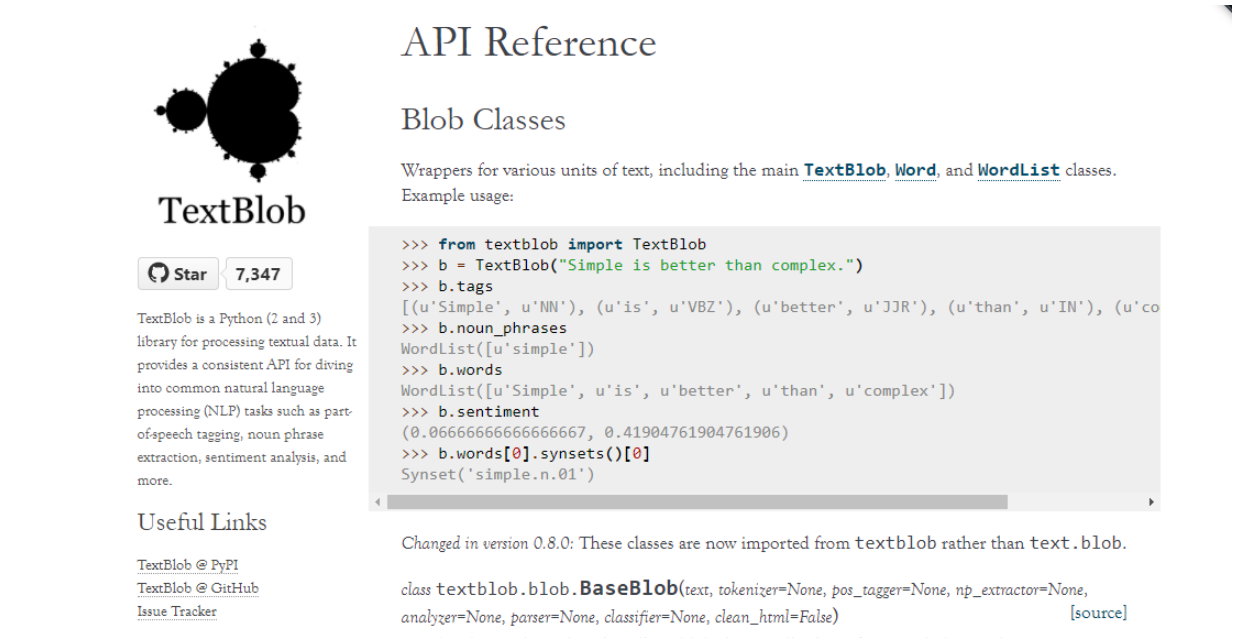
<Resultado da comparação – Código Python>

```
---------------------Resultado-----------------------
O Exame IELTS é o melhor conceituado no Twitter!!
Tem 185 tweets positivos, 35 tweets negativos e 230 tweets neutros
Quanto a subjetividade, tem  83 tweets subjetivos, 337 tweets objetivos e 30 tweets neutros
O segundo colocado é o exame TOEFL
Tem 138 tweets positivos, 48 tweets negativos e 264 tweets neutros
Quanto a subjetividade, tem  67 tweets subjetivos, 350 tweets objetivos e 33 tweets neutros
---------------------Resultado-----------------------
```

<Resultado da comparação - Saída pela utilização do comando PRINT >

AVALIAÇÃO DE DESEMPENHO

Em avaliação de desempenho, precisamos dizer o quanto a nossa lógica é confiável. Nós apoiamos nosso código fonte na utilização da biblioteca TextBlob, que é uma biblioteca Python para processamento de dados textuais. Ela é a principal biblioteca utilizada por desenvolvedores para processamento de dados, possui a licença MIT licensed, que é uma licença criada pelo Instituto de Tecnologia de Massachusetts (MIT). Não é divulgado em sua documentação e em seu github, valores da sua eficiência, ou precisão de acerto. Porém temos certeza que é uma biblioteca muito confiável, está presente em projetos universitário em todo o mundo.



TextBlob

Star 7,347

TextBlob is a Python (2 and 3) library for processing textual data. It provides a consistent API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, and more.

Useful Links

TextBlob @ PyPI
TextBlob @ GitHub
Issue Tracker

API Reference

Blob Classes

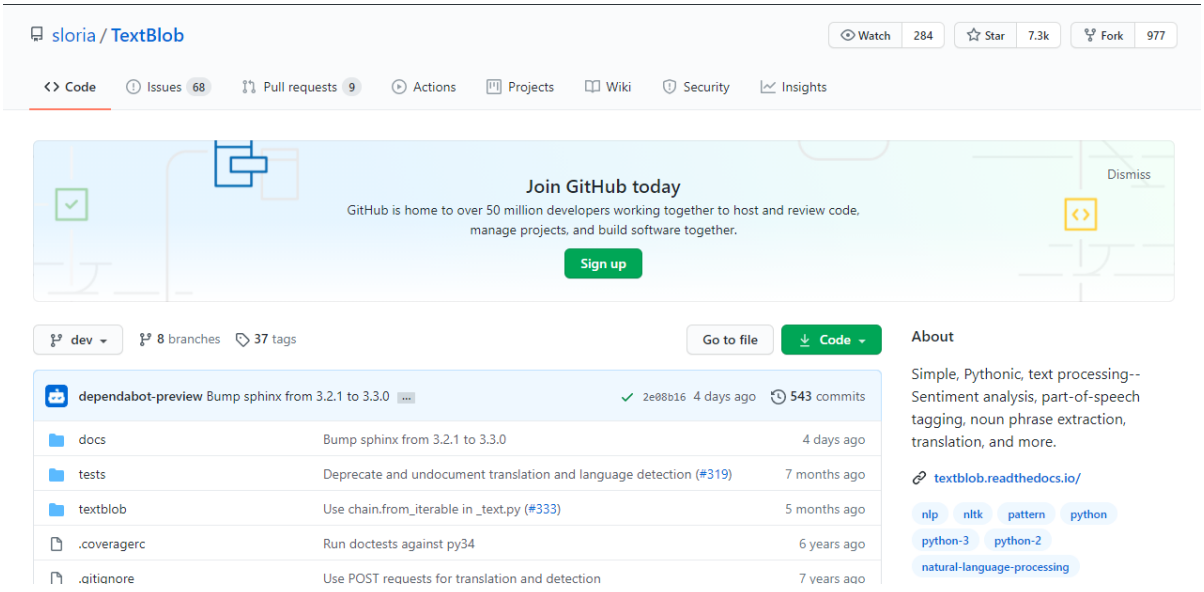
Wrappers for various units of text, including the main **TextBlob**, **Word**, and **WordList** classes. Example usage:

```
>>> from textblob import TextBlob
>>> b = TextBlob("Simple is better than complex.")
>>> b.tags
[(u'Simple', u'NN'), (u'is', u'VBZ'), (u'better', u'JJR'), (u'than', u'IN'), (u'co
>>> b.noun_phrases
WordList([u'simple'])
>>> b.words
WordList([u'Simple', u'is', u'better', u'than', u'complex'])
>>> b.sentiment
(0.06666666666666667, 0.41904761904761906)
>>> b.words[0].synsets()[0]
Synset('simple.n.01')
```

Changed in version 0.8.0: These classes are now imported from `textblob` rather than `text.blob`.

class `textblob.blob.`**`BaseBlob`**(*text*, *tokenizer=None*, *pos_tagger=None*, *np_extractor=None*, *analyzer=None*, *parser=None*, *classifier=None*, *clean_html=False*) [source]

< Documentação da Biblíoteca TextBlob >



sloria / TextBlob

Watch 284 Star 7.3k Fork 977

Code Issues 68 Pull requests 9 Actions Projects Wiki Security Insights

Join GitHub today

GitHub is home to over 50 million developers working together to host and review code, manage projects, and build software together.

Sign up

Dismiss

dev 8 branches 37 tags Go to file Code

dependabot-preview Bump sphinx from 3.2.1 to 3.3.0 2e08b16 4 days ago 543 commits

docs	Bump sphinx from 3.2.1 to 3.3.0	4 days ago
tests	Deprecate and undocument translation and language detection (#319)	7 months ago
textblob	Use chain.from_iterable in _text.py (#333)	5 months ago
.coveragerc	Run doctests against py34	6 years ago
.gitignore	Use POST requests for translation and detection	7 years ago

About

Simple, Pythonic, text processing--Sentiment analysis, part-of-speech tagging, noun phrase extraction, translation, and more.

textblob.readthedocs.io/

nlp nltk pattern python python-3 python-2 natural-language-processing

< Github oficial da biblioteca TextBlob https://github.com/sloria/TextBlob >

TextBlob: Simplified Text Processing

pypi v0.15.3 travis passing

Homepage: https://textblob.readthedocs.io/

TextBlob is a Python (2 and 3) library for processing textual data. It provides a simple API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more.

```
from textblob import TextBlob

text = '''
The titular threat of The Blob has always struck me as the ultimate movie
monster: an insatiably hungry, amoeba-like mass able to penetrate
virtually any safeguard, capable of--as a doomed doctor chillingly
describes it--"assimilating flesh on contact.
Snide comparisons to gelatin be damned, it's a concept with the most
devastating of potential consequences, not unlike the grey goo scenario
proposed by technological theorists fearful of
artificial intelligence run rampant.
'''

blob = TextBlob(text)
blob.tags           # [('The', 'DT'), ('titular', 'JJ'),
                    #  ('threat', 'NN'), ('of', 'IN'), ...]

blob.noun_phrases   # WordList(['titular threat', 'blob',
                    #            'ultimate movie monster',
                    #            'amoeba-like mass', ...])
```

< Um breve início da documentação TextBlobhttps://github.com/sloria/TextBlob#textblob-simplified-text-processing >

DISTRIBUIÇÃO (IMPLANTAÇÃO)

Ao longo de tudo que foi detalhado até o momento, mostramos que com dois arquivos .py, um para autenticação e coleta de dois csv's, e outro para para classificações, conseguimos mostrar em tela qual o melhor resultado e seus valores em comparação ao outro exame.

Para implementação, a ideia é a criação de um APP onde o usuário vai clicar no botão "comparar" e irá disparar toda essa solução, mostrando ao término do processamento na tela do aplicativo qual a melhor opção de prova.