Ciências da Computação
Fundamentos de Programação - 2021.2
Professor: Eldrey Galindo

Componentes do Grupo:

André Felipe Gomes de Sousa - afgs@cesar.school - Turma A
Davi Aleixo de Carvalho - dac2@cesar.school - Turma A
Diogo Costa Maranhão Rodrigues - dcmr@cesar.school - Turma A
Lucas Silvestre de Barros - lsb5@cesar.school - Turma B

1. INTRODUÇÃO

A presente atividade é desenvolvida para a disciplina de fundamentos da programação e consiste na criação de um algoritmo na linguagem python, capaz de analisar textos no idioma português brasileiro e de polarizar as palavras aprendidas de acordo com os sentimentos por elas transmitidos, com a finalidade de interpretar o sentimento de uma pessoa com base na análise texto.

2. EXPOSIÇÃO DAS ETAPAS DO DESENVOLVIMENTO.

Verificou-se que para a realização de um algoritmo de análise de sentimento é necessário a realização das seguintes etapas:

a) Coleta de conteúdo e definição de polaridades:

Inicialmente é necessária a criação de uma amostra de palavras já classificadas que serão utilizadas como referência, razão pela qual foi necessária a:

- **Escolha de Palavras e definição de polaridades, para criação de um referencial:**

Neste ponto, a equipe tratou de criar uma lista com 30 (trinta) palavras, atribuindo a cada uma destas, um peso que varia de -5 (muito ruim) a 5 (muito boa).

Trabalho.py | frases-escolhidas.txt | peso-palavra.txt

E: > Users > Lucas > Desktop > Trabalho > peso-palavra.txt

```
trair, -5
corno, -4
marmitex, 4
linda, 3
marmanjos, -2
careca, 5
autoestima, 2
criança, 1
baixa, -2
sei, 4
fazer, 2
supera, 2
sendo, 0
xingada, -2
difícil, -3
paulista, -5
ciúme, 2
você, 1
poder, 3
chamar, 1
segredo, -1
coisado, 3
desgraça, -5
noiva, 5
Deus, 3
acordar, -3
gostosa, 5
odiar, -5
é, 4
me, 1
```

Vale esclarecer que a pontuação atribuída às palavras se deu a partir da média aproximada das notas atribuídas pelos integrantes do grupo

- **Escolha de frases e definição de suas polaridades, para posterior validação:**

Neste ponto, o grupo tratou de selecionar um conjunto de 10 (dez) frases para validação do algoritmo.

As frases selecionadas foram retiradas do twitter e formatadas para evitar erros, as quais, por conseguinte, foram adicionadas a um arquivo .txt e separadas por linha, conforme exposto a seguir:

a) “Ela está namorando outro só para me fazer ciúme, me ama mesmo, supera filha.”

b) “Se me trair fica careca, além de corno, você é cabeleireiro.”
c) “Uma criança está sendo xingada por marmanjos ofendidos.”
d) “Porque o nome é autoestima se ela tá sempre baixa?”
e) “Marmitex linda igual você.”
f) “Deve ser difícil me odiar e não poder me chamar de paulista.”
g) “Sei lá, ta tudo meio coisado e eu tô tipo sei lá”
h) “Que desgraça de tema é esse meu Deus.”
i) “O segredo pra não acordar de benga dura é não acordar."
j) “Por que toda noiva é gostosa?”

Depois da escolha das frases, os integrantes do grupo passaram a realizar suas classificações na escala -5 (muito ruim) a 5 (muito boa), os resultados são os listados abaixo:

Frase	André	Aleixo	Diogo	Lucas	Média
a	-2	-4	-3	-1	-2,5
b	-3	-1	-1	-4	-2,25
c	-5	-4	-5	-3	-4,25
d	-2	-2	-1	-2	-1,75
e	4	5	5	4	4,5
f	-5	-5	-5	-5	-5
g	0	2	0	1	0,75
h	-3	-3	-3	-2	-2.2
i	1	4	-2	2	1,25
j	5	5	4	5	4,75

b) Criação da aplicação mediante a utilização das amostras previamente classificadas:

Após a definição dos parâmetros, foi dado início à etapa de codificação, através do qual se pretendeu produzir um algoritmo capaz de ler um arquivo no formato .txt com as palavras e pesos de referência e ler um arquivo no formato .txt com as frases que serão analisadas, para que ao final a aplicação salve um arquivo no formato .csv com cabeçalho, contendo os seguintes dados estruturados: Frase; Lista de palavras reconhecidas; Resultado do cálculo de sentimento; Palavras que foram aprendidas.

Em sua primeira reunião, o grupo realizou as seguintes atividades:

Inicialmente, foi necessário inicializar um dicionário, para armazenamento das palavras e pesos contidas no arquivo de texto “peso-palavra.txt” e também a inicialização de uma lista, para armazenamento das frases contidas também em um arquivo de texto “frases-escolhidas.txt”.

Após isto, foi utilizada uma estrutura de repetição para separar as palavras do peso e incluí-las no dicionário, obtendo-se assim o seguinte resultado no dicionário: {palavra: peso}. Depois disso utilizou-se de uma segunda estrutura de repetição para coletar as frases, mas antes de incluí-las na lista, utilizou-se a função “maketrans” para remover os caracteres indesejados.

Em seguida as frases foram separadas em palavras através do comando “split” e armazenadas na lista anteriormente inicializada.

Com isso, foi possível armazenar todos os dados necessários para resolução do problema proposto, dentro da memória do computador.

Muitos problemas foram encontrados nessas etapas, sobretudo para fazer a inclusão das palavras e dos pesos, mas eles acabaram sendo resolvidos eventualmente.

Fizemos mais alterações seguindo o passo a passo do problema, para que a aplicação fosse capaz de: reconhecer as palavras de referências contidas na frase;

realizar o cálculo do sentimento e armazenar as palavras ainda não conhecidas, tendo inicialmente sido desenvolvido o seguinte código:

```python
getWords = {}
unwantedSymbols = {'?': '', '!': '', ',': '', '.': '', ':': '', ';': '', '0': '', '1': '', '2': '',
                   '3': '', '4': '', '5': '', '6': '', '7': '', '8': '', '9': ''}
setPhraseValue = {}
fileWords = open("peso-palavra", 'r+', encoding="utf-8")
valuePhrase = []
addWords = []
newWord = False
phraseAverage = 0

for line in fileWords.readlines():
    line = line.replace('\n', '')
    line = line.split(', ')
    getWords[line[0]] = line[1]

filePhrases = open('frases-escolhidas', 'r', encoding="utf-8")

for line in filePhrases.readlines():
    line = line.casefold()
    makeLineTrans = line.maketrans(unwantedSymbols)
    line = line.translate(makeLineTrans)
    valuePhrase.append(line.splitlines())

for i in range(len(valuePhrase)):
    phrase = str(valuePhrase[i])
    phrase = phrase.split(' ')
    for j in range(len(phrase)):
        if str(phrase[j]) in getWords:
            phraseAverage += int(getWords[str(phrase[j])])

        else:
            newWord = True
            addWords.append(str(phrase[j]))

    phraseAverage = (phraseAverage/len(phrase))
    addWords.append(phraseAverage)
    setPhraseValue[str(valuePhrase[i])] = phraseAverage

    if newWord:
        for j in range(len(addWords)):
            fileWords.write('{}, {}'.format(addWords[i], addWords[-1::]))

    addWords.clear()

print(setPhraseValue)
```

O código em questão ainda não funcionava perfeitamente, apresentava alguns erros, pois não conseguia ler corretamente as palavras contidas no dicionário de palavras conhecidas e não conhecidas, para poder atribuir o valor do cálculo de sentimento da frase a cada uma dessas palavras.

Na segunda reunião, a equipe tratou de aprimorar o código anteriormente desenvolvido a partir da criação de funções, dentre as quais se pode citar: a função "*open_files*" para abrir o arquivo; a "*word_files*" função para adicionar as palavras e pesos ao dicionário; a função "phrase_files" para separar as frases, retirar o caracteres indesejados e incluí-las em uma lista e uma função para assimilar novas palavras.

Além destas funções supracitadas, a equipe tratou de criar mais uma função, a chamada "treat_words", para solucionar o problema anteriormente descrito, através do qual foi realizado o tratamento das novas palavras conhecidas e para finalizar.

Por fim, a equipe se concentrou na última etapa, a de salvar os resultados obtidos em um arquivo .CSV estruturado.

Nesta fase, foram enfrentadas diversas dificuldades, sobretudo para os resultados fossem estruturados nas colunas dos seus respectivos títulos, o que foi solucionado através da criação da função "*grade_phrases*".

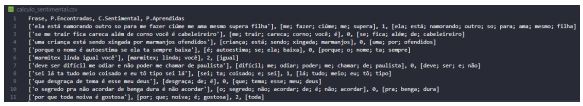
As funções anteriormente descritas podem ser verificadas nas imagens colacionadas a seguir:

```python
def open_files(file_name, type_f):
    file = open(file_name, type_f, encoding='UTF-8')

    return file


def treat_words(word):
    if word.startswith("["):
        word = str(word)[2:]
        return word
    elif word.endswith("']"):
        word = str(word)[:-2]
        return word
    else:
        return word


def word_file(dictio):
    fileWord = open_files('peso-palavra.txt', 'r+')
    for line in fileWord.readlines():
        line = line.replace('\n', '')
        line = line.split(', ')
        dictio[line[0]] = line[1]

    return dictio


def phrase_file(list_phrase, symbols):
    filePhrase = open_files('frases-escolhidas.txt', 'r')
    for line in filePhrase.readlines():
        line = line.casefold()
        makeLineTrans = line.maketrans(symbols)
        line = line.translate(makeLineTrans)
        list_phrase.append(line.splitlines())

    return list_phrase


def add_words(list_words, value):
    global dictWords
    file = open_files('peso-palavra.txt', 'a')
    for word in list_words:
        wordAdd = treat_words(word)
        if wordAdd != ' ':
            file.write('\n{}, {}'.format(wordAdd, value))
            dictWords[wordAdd] = str(value)

    return None
```

```python
def grade_phrases(list_phrase):
    pValue = 0
    notKnownWord = []
    knownWords = []

    knownWords_tratada = ""
    notKnownWord_tratada = ""
    phrase_tratada = ""

    file_csv = open('calculo_sentimental.csv', 'w+', encoding='UTF-8')
    file_csv.write("Frase, P.Encontradas, C.Sentimental, P.Aprendidas")
    file_csv.write('\n')
    for i in range(len(list_phrase)):
        phrase = str(list_phrase[i])
        phrase = phrase.split(' ')
        for word in phrase:
            wordCheck = treat_words(word)
            if wordCheck in dictWords:
                pValue += int(dictWords.get(wordCheck))
                knownWords.append(wordCheck)
            else:
                if wordCheck not in notKnownWord:
                    notKnownWord.append(wordCheck)

        pValue = round(pValue / len(phrase))

        notKnownWord_tratada = '; '.join(notKnownWord)
        phrase_tratada = ' '.join(phrase)
        knownWords_tratada = '; '.join(knownWords)
        file_csv.write("{}, [{}], {}, [{}]".format(phrase_tratada, knownWords_tratada,
                                                   pValue, notKnownWord_tratada))
        file_csv.write('\n')
        add_words(notKnownWord, pValue)
        notKnownWord.clear()
        knownWords.clear()

    return None


unwantedSymbols = {'?': '', '!': '', ',': '', '.': '', ':': '', ';': '', '0': '', '1': '', '2': ''
                   '3': '', '4': '', '5': '', '6': '', '7': '', '8': '', '9': ''}

dictWords = {}
listPhrase = []
dictWords = word_file(dictWords)
listPhrase = phrase_file(listPhrase, unwantedSymbols)
grade_phrases(listPhrase)
```

c) Apresentação dos resultados:

Os resultados obtidos ao executar a aplicação foi o seguinte:



E a representação do arquivo CSV na forma de tabela foi o seguinte:

Frase	P.Encontradas	C.Sentimental	P.Aprendidas
[ela está namorando outro so para me fazer ciúme me ama mesmo sapera filha']	[me; fazer; ciúme; me; sapera]	1	[ela; está; namorando; outro; so; para; ama; mesmo; filha]
['ue me trair fica careca além do corno você é cabeleireiro']	[me; trair; careca; corno; você]	0	[ue; fica; além; de; cabeleireiro]
['uma criança está sendo xingada por marmanjos ofendidos']	[criança; está; sendo; xingada; marmanjos]	0	[uma; por; ofendidos]
['porque o nome é autocntimo so ela ta sempre baixa']	[é; autocntimo; so; ela; baixa]	0	[porque; o; nome; ta; sempre]
['marmitex linda igual você']	[marmitex; linda; você]	2	[igual]
['deve ser difícil eu odiar e não poder me chamar de paulista']	[difícil; me; odiar; poder; me; chamar; de; paulista]	0	[deve; ser; e; não]
['sei lá ta tudo meio colado e eu tô tipo sei lá']	[sei; ta; colado; e; sei]	1	[lá; tudo; meio; eu; tô; tipo]
['que desgraça de tema é esse meu deus']	[desgraça; de; é]	0	[que; tema; esse; meu; deus]
['o negrodo pra não acordar de benga dura é não acordar']	[o; negrodo; não; acordar; de; é; não; acordar]	0	[pra; benga; dura]
['por que toda noiva é gostosa']	[por; que; noiva; é; gostosa]	2	[toda]

Além disso, as novas palavras conhecidas, foram inscritas ao arquivos de texto, da seguinte forma:

```
ela, 1
está, 1
namorando, 1
outro, 1
so, 1
para, 1
ama, 1
mesmo, 1
filha, 1
se, 0
fica, 0
além, 0
de, 0
cabeleireiro, 0
uma, 0
por, 0
ofendidos, 0
porque, 0
o, 0
nome, 0
ta, 0
sempre, 0
igual, 2
deve, 0
ser, 0
e, 0
não, 0
lá, 1
tudo, 1
meio, 1
eu, 1
tô, 1
tipo, 1
que, 0
tema, 0
esse, 0
meu, 0
deus, 0
pra, 0
benga, 0
dura, 0
toda, 2
```

Nota-se, portanto, que o algoritmo atendeu todos os requisitos estabelecidos na atividade, já salvou um arquivos .CSV com as frases, as palavras encontradas, o resultado do

cálculo sentimental e as palavras aprendidas a cada frase.

3. CONCLUSÃO.

Neste ponto, cumpre apontar que caso fosse realizada uma comparação entre o valor do cálculo de sentimento das frases obtido pelo algoritmo e o valor das notas anteriormente atribuídas pelos integrantes da equipe, verificou-se que a aplicação calculou pesos mais próximos à zero, portanto, neutros.

Enquanto as notas apresentadas pelos integrantes do grupo eram mais extravagantes, seja em positividade, seja em negatividade.

Tal situação decorre do fato de que as frases selecionadas possuíam uma grande quantidade de elementos ainda não classificados, os quais integravam o divisor na equação do cálculo de sentimentos, além do fato de que a maior parte das frases selecionadas estavam permeadas por sarcasmo, fato este que dificulta a aferição do resultado, uma vez que certas palavras quando isoladas possuem pesos diferentes daqueles atribuídos às frases.

Ademais, em atenção a todos os passos narrados e aos resultados obtidos, é, forçoso concluir que o grupo foi capaz de desenvolver de forma satisfatória uma aplicação capaz de realizar o cálculo do sentimento, em atendimento aos requisitos da atividade proposta.