FACULDADE DE ENGENHARIA DE SOROCABA - FACENS
PÓS GRADUAÇÃO EM ESPECIALIZAÇÃO EM CIÊNCIA DE DADOS

ALEX COELHO ABRANTES

BRUNO ALVES COMITRE

DETECÇÃO AUTOMATIZADA DE NOTÍCIAS FALSAS:

PESQUISA COM RECONHECIMENTO DE INTEGRIDADE DAS INFORMAÇÕES

Tese apresentada ao Programa de Pós-Graduação em Ciências de Dados da Faculdade de Engenharia de Sorocaba - FACENS, como requisito parcial para a obtenção do título de Pós Graduado em Ciência de Dados.

Orientado: Prof. Matheus Mota

Coord.: Prof. Fernando Vieira da Silva

SOROCABA

2019

SUMÁRIO

- 1. Análise Explorátoria
  - 1.1 Imports
  - 1.2 Leitura do Dataset
  - 1.3 Classificação das Features
  - 1.4 Dicionário dos Dados
  - 1.5 Análise dos Dados
  - 1.6 Dados Faltantes
  - 1.7 Novos Recursos (Features)
    - 1.7.1 Recursos por Palavras (Tokens) e Entidades
  - 1.8 Dados Desiquilibrados e Random Shuffle
  - 1.9. Exportando Dataframe Refatorado CSV
- 2 Análise Gráfica
  - 2.1 WordCloud
  - 2.2 Histograma
    - 2.2.1 Histograma da Quantidade de Palavras
    - 2.2.2 Histograma do Comprimento de Palavras
    - 2.2.3 Histograma da Quantidade de Adjetivos
  - 2.3 Diagrama de Caixa
    - 2.3.1 Diagrama de Caixa da Quantidade de Palavras
  - 2.4 Gráfio de Dispersão
    - 2.4.1 Dispersão de Palavras x Comprimento
  - 2.5 Matriz de correlação
- REFERÊNCIAS

1. Análise Explorátoria:

Nesta fase do trabalho, inicialmente, faz necessário a aplicação de técnicas para manusear valores faltantes e fazer transformações de variáveis. Os dados serão ajustados e estreitando os presupostos para empregar técnicas gráficas e quantitativas, visando maximizar a obtenção de informações, tendências e detectação decomportamentos.

1.1 Imports

In [1]:

```
!pip install names

import numpy as np
import pandas as pd
import csv
import random
import pandas_profiling #conda install -c conda-forge pandas-profiling
import names #pip install names
import nltk
import matplotlib.pyplot as plt
import plotly.offline as py
import plotly.graph_objs as go
import seaborn as sns
import string
import spacy
from spacy import displacy
from nltk import pos_tag
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.tokenize import RegexpTokenizer
from nltk.tokenize import TweetTokenizer
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
from pandas.io.json import json_normalize
from plotly import tools
from tqdm import tqdm


nltk.download('averaged_perceptron_tagger')
```

```
Requirement already satisfied: names in c:\users\cliente\anaconda3\envs\un
iversity\lib\site-packages (0.3.0)

[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\cliente\AppData\Roaming\nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
```

Out[1]:

```
True
```

1.2 Leitura do Dataset

In [3]:

```
# Train Dataset
PATH_DATA_TRAIN = 'train.csv'

## arquivo disponivel em: https://drive.google.com/drive/folders/1LqNzxY8l0EgznlCD-g873VD-ys1BRN2-?fbclid=IwAR3PPLu4hgNdKfQDJzrLGwV6L42Vm3xBrrcquuCOR4ySS97bVvU46JDaR2s

train_data = pd.read_csv(PATH_DATA_TRAIN)
train_data = train_data.sample(frac = 1) # Randomly Smaple data, ratio is 100%
train_data.head()
```

Out[3]:

	id	title	author	text	label
3958	3958	Brits Fined and Prosecuted for Weeds, Weeping,...	Michael Tennant	Email \nPetty tyranny is alive and well in the...	1
7813	7813	Love and Black Lives, in Pictures Found on a B...	Annie Correal	One night six years ago, on a quiet side stree...	0
3231	3231	Dutch Elections: Geert Wilders Slams 'Hate Pre...	Oliver JJ Lane	Geert Wilders, leader of the Party for Freedom...	0
15966	15966	Bundy Brothers To Be Prosecuted in Nevada Foll...	Ryan Banister	\nAmmon and Ryan have recently been released f...	1
6146	6146	Reward Is Doubled as Authorities Seek Leads in...	Eli Rosenberg	More than five months after a mysterious subst...	0

1.3 Classificação das Features

In [4]:

```
table = [["id","Nominal Qualitativo"],["title","Nominal Qualitativo"],
         ["author","Nominal Qualitativo"],["text","Nominal Qualitativo"],
         ["label","Quantitativo Discreto"]]

filing = pd.DataFrame(table, columns=["Variável", "Classificação"])
filing
```

Out[4]:

	Variável	Classificação
0	id	Nominal Qualitativo
1	title	Nominal Qualitativo
2	author	Nominal Qualitativo
3	text	Nominal Qualitativo
4	label	Quantitativo Discreto

1.4 Dicionário dos dados

O Dataset "fake news" contém as seguintes informações:

- **ID:** id único da notícia
- **TITLE:** título da notícia
- **AUTHOR:** autor da notícia
- **TEXT:** texto da notícia
- **LABEL:** rótulo que marca se a notícia é potencialmente não confiável
  - 1: não confiável
  - 0: confiável

1.5 Análise dos dados

Ao importar os dados, é importante entender e identificar o intervalo de preditores específicos, identificar o tipo de dados de cada preditor, bem como calcular o número ou a porcentagem de valores omissos para cada preditor. Usaremos a biblioteca pandas_profiling, que fornece muitas funções extremamente úteis para a análise exploratória de dados.

Observação: Foi retitado para demonstração a aleatoriedade dos dados

In [5]:

```
train_data_profiling = pd.read_csv(PATH_DATA_TRAIN)
```

In [6]:

```
profile = pandas_profiling.ProfileReport(train_data_profiling)
display(profile)
```

Overview

Dataset info

Number of variables	5
Number of observations	20800
Missing cells	2554 (< 0.1%)
Duplicate rows	0 (0.0%)
Total size in memory	812.6 KiB
Average record size in memory	40.0 B

Variables types

Numeric	1
Categorical	3
Boolean	1
Date	0
URL	0
Text (Unique)	0
Rejected	0
Unsupported	0

Warnings

`author` has a high cardinality: 4202 distinct values	**Warning**
`author` has 1957 (9.4%) missing values	**Missing**
`text` has a high cardinality: 20387 distinct values	**Warning**
`title` has a high cardinality: 19804 distinct values	**Warning**
`title` has 558 (< 0.1%) missing values	**Missing**

1.6 Dados Faltantes

A falta de dados pode afetar a análise e o treinamento, que poderá levar a falhas no aprendizado. Então, é possível dizer se há dados ausentes no conjunto de dados? Sim, pelo relatório gerado por pandas_profiting, identificou-se:

O título do atributo tem 558 amostras (2,68%) com valores ausentes.

O autor do atributo possui 1957 amostras (9,41%) com valores ausentes.

O texto do atributo tem 39 amostras (0,19%) com valores ausentes.

Como existe dados faltantes nas 3 features do dataset (title, Author e Text) eliminar as linhas em que há dados ausentes neste caso é a melhor opção para não comprometer a análise e o treimento.

In [7]:

```
print('Antes do dropna tínhamos {} registros'.format(train_data.shape[0]))
train_data.dropna(inplace=True)
print('Depois do dropna temos {} registro'.format(train_data.shape[0]))
```

```
Antes do dropna tínhamos 20800 registros
Depois do dropna temos 18285 registro
```

1.7 Novos Recursos (Features)

Por fim, após os tratamentos dos dados, identificou-se a necessidade de criar novos recursos (features). A Análise de texto não é uma das tarefas mais fácil a se fazer, embora, seja possível por meio de extração de informações por palavras obter uma melhor compreensão da construção de um texto. O objetivo da análise foi identificar o volume de vezes que determinado texto contém: exclamação, questão, simbolo, palavras únicas e suas classes gramaticais como: substantivos, adjetivos e verbos. Entendemos ser importante identificar as entidades dentro dos contextos como: Pessoas, Grupos Politicos, Organizações, Valores monetários, Nações. Para extrair essas informações utilizou-se a biblioteca Spacy. Com esses novos recursos serão utilizados nas análises gráficas e no aprendizado de máquina.

1.7.1 Recursos por Palavras (Tokens) e Entidades

In [8]:

```
print('Função retorna a POS TAG')
def tag_part_of_speech(text):
    text_splited = text.split(' ')
    text_splited = [''.join(c for c in s if c not in string.punctuation) for s in text_
splited]
    text_splited = [s for s in text_splited if s]
    pos_list = pos_tag(text_splited)
    #palavra
    noun = [w[0] for w in pos_list if w[1] in ('NN','NNP','NNPS','NNS')]
    adjective = [w[0] for w in pos_list if w[1] in ('JJ','JJR','JJS')]
    verb = [w[0] for w in pos_list if w[1] in ('VB','VBD','VBG','VBN','VBP','VBZ')]
    #quantidade
    noun_count = len([w for w in pos_list if w[1] in ('NN','NNP','NNPS','NNS')])
    adjective_count = len([w for w in pos_list if w[1] in ('JJ','JJR','JJS')])
    verb_count = len([w for w in pos_list if w[1] in ('VB','VBD','VBG','VBN','VBP','VB
Z')])

    return[noun, adjective, verb, noun_count, adjective_count, verb_count]

tokenizer = RegexpTokenizer(r'\w+')
```

```
Função retorna a POS TAG
```

In [9]:

```
print('Função retorna tokens unicos')
def token_distint(df):
    list_words = []
    tokenizer = RegexpTokenizer(r'\w+')
    for item in df:
        for token in tokenizer.tokenize(item.lower()):
            if not token in list_words:
                list_words.append(token)
    return list_words
```

```
Função retorna tokens unicos
```

In [10]:

```
print("Geração de recursos baseados nos titulos")
tokenizer = RegexpTokenizer(r'\w+')
for df in ([train_data]):
    df['title_token'] = df['title'].apply(lambda x : tokenizer.tokenize(x.lower()))
    df['title_token_distint'] = df['title'].apply(lambda x : token_distint(x))
    df['title_comprimento'] = df['title'].apply(lambda x : len(str(x)))
    df['title_num_palavras'] = df['title'].apply(lambda x: len(str(x).split()))
    df['title_num_palavras_unicas'] = df['title'].apply(lambda x: len(set(w for w in st
r(x).split())))
    df['title_palavras_vs_unico'] = df['title_num_palavras_unicas'] / df['title_num_pal
avras']
    df['title_substantivos'], df['title_adjetivos'], df['title_verbos'], df['title_cont
agem_substantivos'], df['title_contagem_adjetivos'], df['title_contagem_verbos'] = zip(
*df['title'].apply(lambda text: tag_part_of_speech(str(text))))
```

```
Geração de recursos baseados nos titulos
```

In [11]:

```
print("Geração de recursos baseados nos textos")
tokenizer = RegexpTokenizer(r'\w+')
for df in ([train_data]):
    df['text_token'] = df['text'].apply(lambda x : tokenizer.tokenize(x.lower()))
    df['text_token_distint'] = df['text'].apply(lambda x : token_distint(x))
    df['text_comprimento'] = df['text'].apply(lambda x : len(str(x)))
    df['text_num_palavras'] = df['text'].apply(lambda x: len(str(x).split()))
    df['text_num_palavras_unicas'] = df['text'].apply(lambda x: len(set(w for w in str(
x).split())))
    df['text_palavras_vs_unico'] = df['text_num_palavras_unicas'] / df['text_num_palavr
as']
    df['text_substantivos'], df['text_adjetivos'], df['text_verbos'], df['text_contagem
_substantivos'], df['text_contagem_adjetivos'], df['text_contagem_verbos'] = zip(*df['t
ext'].apply(lambda text: tag_part_of_speech(str(text))))
```

```
Geração de recursos baseados nos textos
```

1.8 Dados Desiquilibrados e Random Shuflle

Após eliminar as linhas de dados faltantes, aplicou-se uma análise descritiva para verificar se houve desequilíbrio nos dados, constatou-se que a média de 43% da feature Label, na qual armazena valores de 0 ou 1 (não confiáveis e confiáveis) mostra que os dados estão em desequilibrio, constando uma diferença de 2437 registros com o valor 1 (confiáveis) a mais que o valor 0 (não confiáveis). Para manter o conjunto de dados equilibrado aplicou-se o Random Shuffle nos registros com valor 1 (confiáveis) considerando o valor de 7924 registros que é o número total de dados (não confiáveis).

In [12]:

```
train_data.label.describe()
```

Out[12]:

```
count    18285.000000
mean         0.433361
std          0.495553
min          0.000000
25%          0.000000
50%          0.000000
75%          1.000000
max          1.000000
Name: label, dtype: float64
```

In [13]:

```
unreliable = train_data[train_data['label'] == 1]
print('Não confiável: ', len(unreliable))

reliable = train_data[train_data['label'] == 0]
print('Confiável: ', len(reliable))

print('Desequilibrio nos dados de {} registros confiáveis'.format(len(reliable) - len(u
nreliable)))
```

```
Não confiável:  7924
Confiável:  10361
Desequilibrio nos dados de 2437 registros confiáveis
```

In [14]:

```
print('aplicando Random Shuffle')
mean = min(len(unreliable), len(reliable))

un_data = unreliable.sample(n = mean)
print('Não confiável: ', len(un_data))
r_data = reliable.sample(n = mean)
print('Confiável: ', len(r_data))

train_data = pd.concat([un_data, r_data])
```

```
aplicando Random Shuffle
Não confiável:  7924
Confiável:  7924
```

1.9. Exportando Dataframe Refatorado CSV:

A saída de todos os tratamentos dos dados será utilizada na próxima etapa que será o desenvolvimento de um aprendizado de máquina com deep leraning.

In [15]:

```
train_data.to_csv ('train_refatorado.csv', index = None, header=True)
```

2 Análise Gráfica

- Gerar um WordCloud com todos os títulos e textos e analisar as palavras mais utilizadas.
- Gerar um histograma referente ao tamanho dos títulos e textos. Será que existe diferença de tamanho (caracteres) para os títulos e textos confiáveis e não confiáveis?
- Gerar um boxplot referente a quantidade de palavras dos títulos e textos e analisar os valores mínimos, máximos, primeiro e terceiro quartil, mediana e existência de outliers. Será que existe diferença de entre confiáveis e não confiáveis?
- Podemos verificar alguma correlação entre os novos recursos?
- A forma de escrita do texto (exemplos: educado, rude, gírias, etc…) tem influência no sentimento Confiável e Não Confiável?

Utilizamos o arquivo de saída da análise explorátoria, train_refatorado.csv para utilizar os novos recursos criados. Primeiramente, repetimos o processo de importação do dataset e separamos os textos confiáveis e não confiáveis.

2.1 WordCloud

A ideia central ao plotar o gráfico de nuvem de palavras, é encontrar o volume de vezes que determinadas palavra se repetem dentro do corpus do texto. Este tipo de gráfico se trata de um método heurístico a fim de encontrar respostas viáveis, ainda que imperfeitas, pois mesmo observando as palavras de destaques não há muito resultado para nosso problema, porém nos guiam para alguns questionamentos.

In [16]:

```
print('Função para listar as palavras')
def list_words(data):
    l_words = []
    tokenizer = RegexpTokenizer(r'\w+')
    for tokens in data:
        for token in tokens:
            l_words.append(token)
    return l_words
```

```
Função para listar as palavras
```

In [17]:

```
print('Função que plota o gráfico de Words Cloud')
def words_cloud(confiavel, nao_confiavel, title):


    ## Palavras dentro do texto
    words = list_words(confiavel)
    text_confiavel = " ".join(review for review in words)

    words = list_words(nao_confiavel)
    text_nao_confiavel = " ".join(review for review in words)

    ## Carregando a lista de stopwords
    stopwords = nltk.corpus.stopwords.words('english')

    # Generate a word cloud image
    wordcloud_confiavel = WordCloud(stopwords=stopwords, background_color="white", widt
h=1000, height=800, margin=0, collocations=False).generate(text_confiavel)
    wordcloud_nao_confiavel = WordCloud(stopwords=stopwords, background_color="white",
width=1000, height=800, margin=0, collocations=False).generate(text_nao_confiavel)

    ## plotagem
    #plt.figure(1)
    plt.figure(figsize=(12,8))

    plt.subplot(121)
    plt.imshow(wordcloud_confiavel, interpolation='bilinear')
    plt.axis('off')
    plt.margins(x=0,y=0)
    plt.title(title + ' Confiáveis')
    plt.legend()

    plt.subplot(122)
    plt.imshow(wordcloud_nao_confiavel, interpolation='bilinear')
    plt.axis('off')
    plt.margins(x=0,y=0)
    plt.title(title + ' Não Confiáveis')
    plt.legend()

    plt.show()
```

```
Função que plota o gráfico de Words Cloud
```

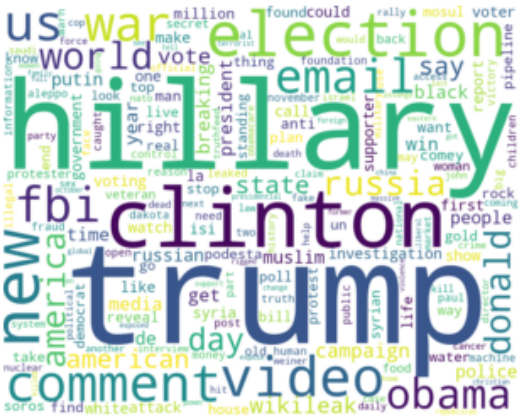
In [18]:

```
words_cloud(reliable['title_token'], unreliable['title_token'], 'Nuvem de Palavras dos
 Títulos')
words_cloud(reliable['text_token'], unreliable['text_token'], 'Nuvem de Palavras dos Te
xtos')
words_cloud(reliable['title_substantivos'], unreliable['title_substantivos'], 'Nuvem de
Palavras Substantivos nos Textos')
words_cloud(reliable['title_verbos'], unreliable['title_verbos'], 'Nuvem de Palavras Ve
rbos nos Textos')
words_cloud(reliable['title_adjetivos'], unreliable['title_adjetivos'], 'Nuvem de Palav
ras Adjetivos nos Textos')
```

No handles with labels found to put in legend.
No handles with labels found to put in legend.

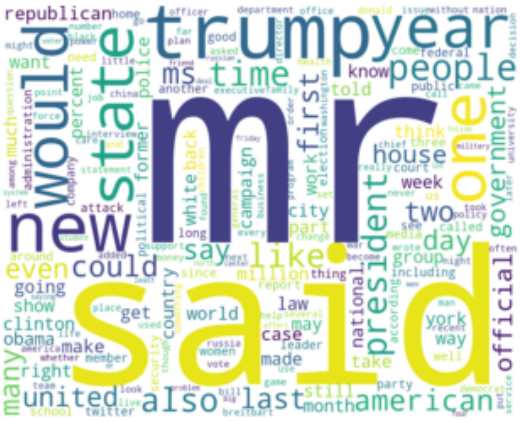
Nuvem de Palavras dos Títulos Confiáveis

Nuvem de Palavras dos Títulos Não Confiáveis



No handles with labels found to put in legend.
No handles with labels found to put in legend.

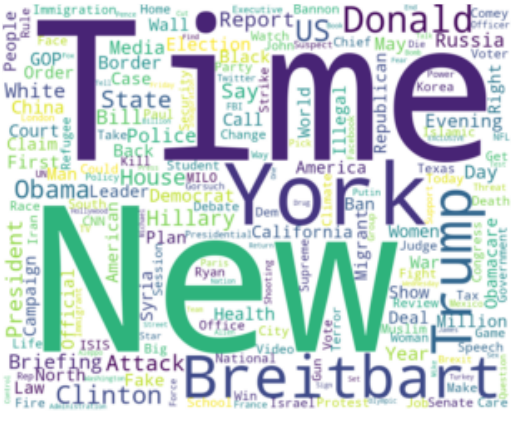
Nuvem de Palavras dos Textos Confiáveis



Nuvem de Palavras dos Textos Não Confiáveis

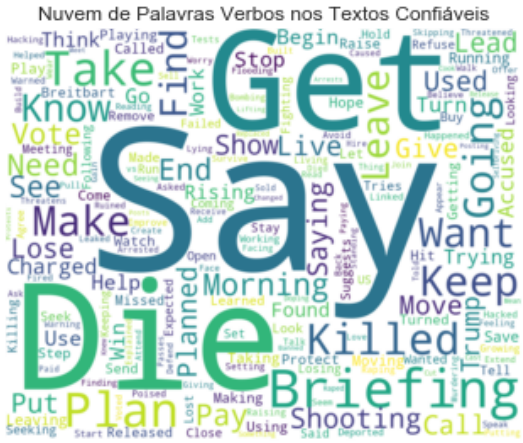
No handles with labels found to put in legend.
No handles with labels found to put in legend.

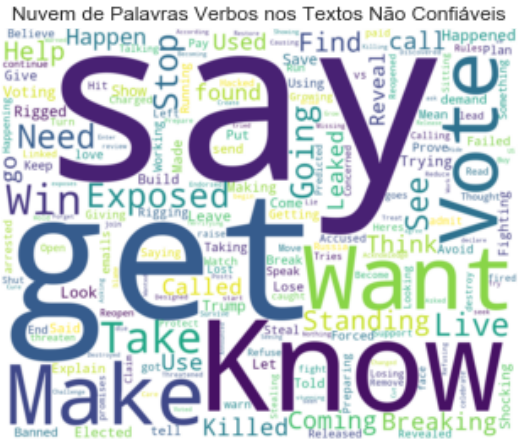
Nuvem de Palavras Substantivos nos Textos Confiáveis



Nuvem de Palavras Substantivos nos Textos Não Confiáveis

No handles with labels found to put in legend.
No handles with labels found to put in legend.






```
No handles with labels found to put in legend.
No handles with labels found to put in legend.
```

2.2 Histograma

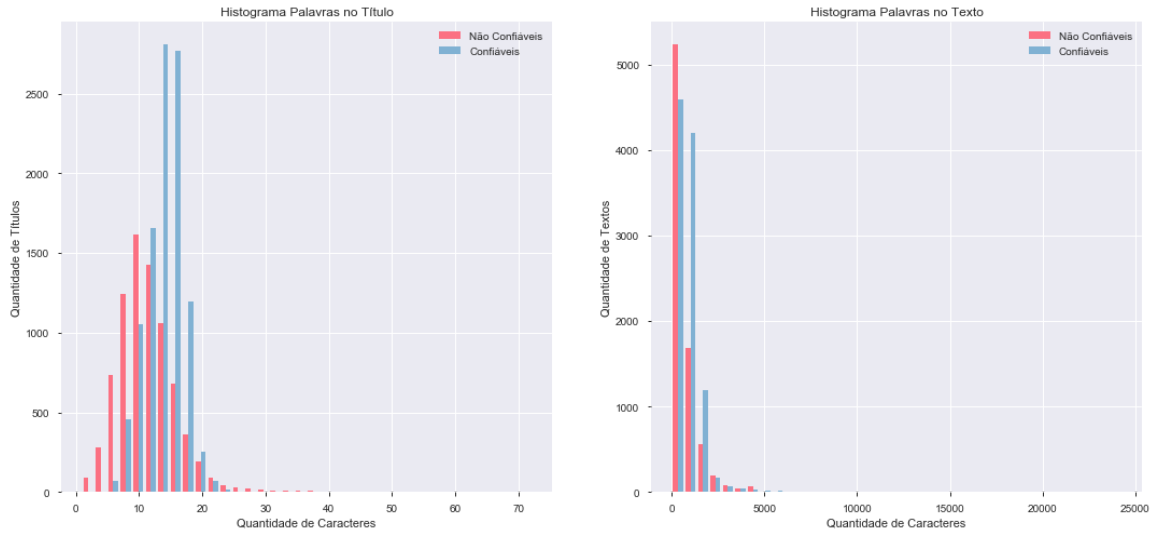
A construção de histogramas tem caráter preliminar em qualquer estudo e é um importante indicador da distribuição de dados. Neste estudo utilizamos a frequência absoluta, que é o número que representa a quantidade de dados em uma determinada amostra ou o intervalo de classe especifico, indicando a frequência (absoluta) com que uma classe aparece no conjunto de dados.

2.2.1 Histograma do Quantidade de Palavras

In [19]:

```
labels = ['Não Confiáveis','Confiáveis']
colors = ['#fb7082','#80b1d3']

plt.figure(figsize=(18,8))
plt.subplot(121)
plt.hist([unreliable.title_num_palavras,reliable.title_num_palavras], bins=int(180/5),
normed=False, color = colors, label=labels)
plt.xlabel('Quantidade de Caracteres')
plt.ylabel('Quantidade de Títulos')
plt.title('Histograma Palavras no Título')
plt.legend()


plt.subplot(122)
plt.hist([unreliable.text_num_palavras,reliable.text_num_palavras], bins=int(180/5), no
rmed=False, color = colors, label=labels)
plt.xlabel('Quantidade de Caracteres')
plt.ylabel('Quantidade de Textos')
plt.title('Histograma Palavras no Texto')
plt.legend()

plt.show()
```

```
C:\Users\cliente\Anaconda3\envs\University\lib\site-packages\ipykernel_lau
ncher.py:6: MatplotlibDeprecationWarning:


The 'normed' kwarg was deprecated in Matplotlib 2.1 and will be removed in
3.1. Use 'density' instead.

C:\Users\cliente\Anaconda3\envs\University\lib\site-packages\ipykernel_lau
ncher.py:14: MatplotlibDeprecationWarning:


The 'normed' kwarg was deprecated in Matplotlib 2.1 and will be removed in
3.1. Use 'density' instead.
```

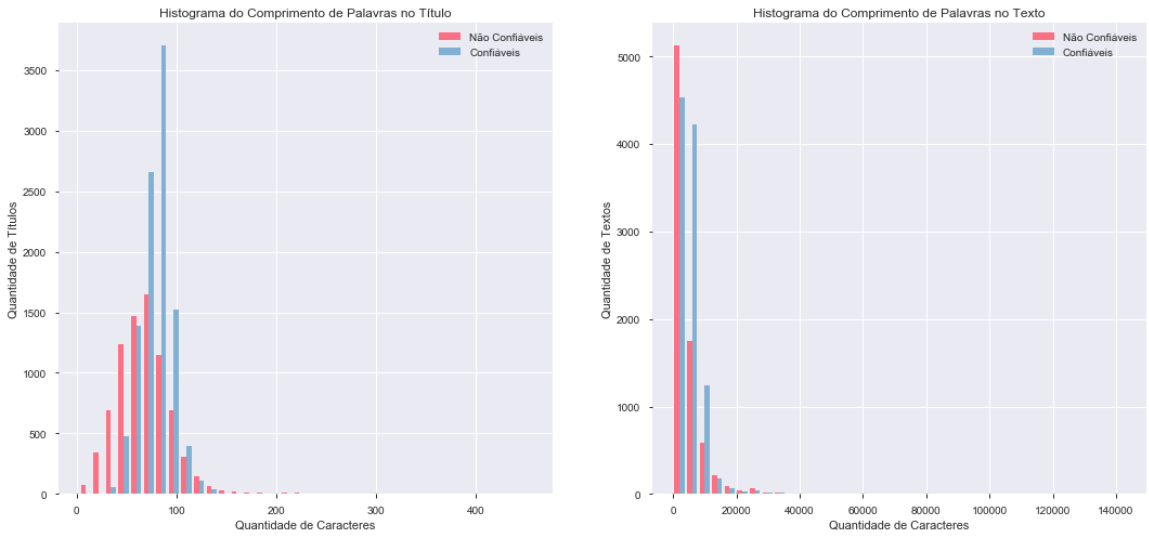


2.2.2 Histograma do Comprimento de Palavras

In [20]:

```
labels = ['Não Confiáveis','Confiáveis']
colors = ['#fb7082','#80b1d3']

plt.figure(figsize=(18,8))
plt.subplot(121)
plt.hist([unreliable.title_comprimento,reliable.title_comprimento], bins=int(180/5), no
rmed=False, color = colors, label=labels)
plt.xlabel('Quantidade de Caracteres')
plt.ylabel('Quantidade de Títulos')
plt.title('Histograma do Comprimento de Palavras no Título')
plt.legend()

plt.subplot(122)
plt.hist([unreliable.text_comprimento,reliable.text_comprimento], bins=int(180/5), norm
ed=False, color = colors, label=labels)
plt.xlabel('Quantidade de Caracteres')
plt.ylabel('Quantidade de Textos')
plt.title('Histograma do Comprimento de Palavras no Texto')
plt.legend()

plt.show()
```

```
C:\Users\cliente\Anaconda3\envs\University\lib\site-packages\ipykernel_lau
ncher.py:6: MatplotlibDeprecationWarning:


The 'normed' kwarg was deprecated in Matplotlib 2.1 and will be removed in
3.1. Use 'density' instead.

C:\Users\cliente\Anaconda3\envs\University\lib\site-packages\ipykernel_lau
ncher.py:13: MatplotlibDeprecationWarning:


The 'normed' kwarg was deprecated in Matplotlib 2.1 and will be removed in
3.1. Use 'density' instead.
```

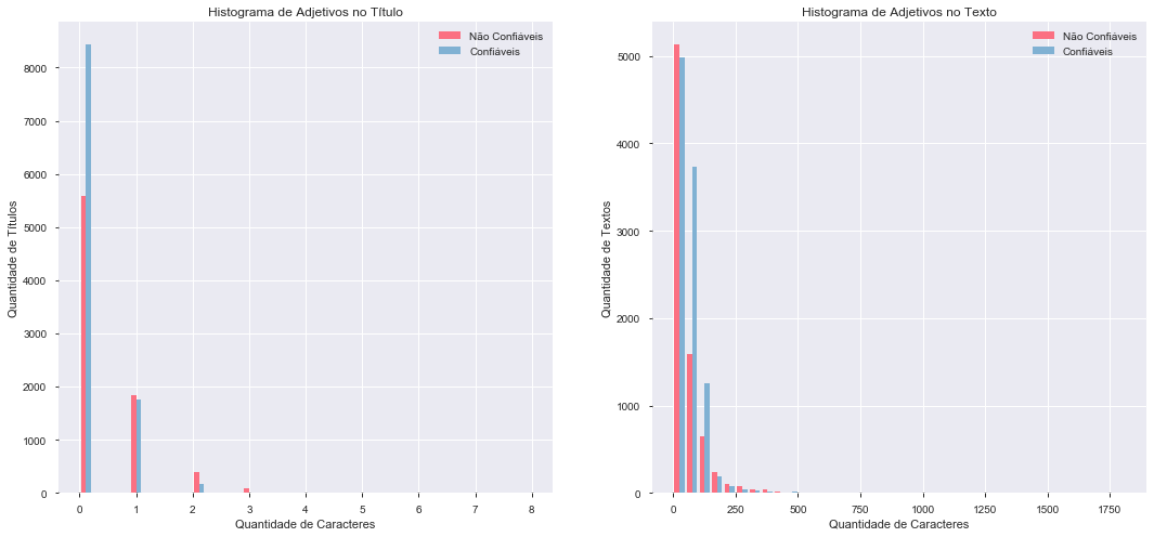


2.2.3 Histograma da Quantidade de Adjetivos

In [21]:

```
labels = ['Não Confiáveis','Confiáveis']
colors = ['#fb7082','#80b1d3']

plt.figure(figsize=(18,8))
plt.subplot(121)
plt.hist([unreliable.title_contagem_adjetivos,reliable.title_contagem_adjetivos], bins=
int(180/5), normed=False, color = colors, label=labels)
plt.xlabel('Quantidade de Caracteres')
plt.ylabel('Quantidade de Títulos')
plt.title('Histograma de Adjetivos no Título')
plt.legend()

plt.subplot(122)
plt.hist([unreliable.text_contagem_adjetivos,reliable.text_contagem_adjetivos], bins=in
t(180/5), normed=False, color = colors, label=labels)
plt.xlabel('Quantidade de Caracteres')
plt.ylabel('Quantidade de Textos')
plt.title('Histograma de Adjetivos no Texto')
plt.legend()

plt.show()
```

```
C:\Users\cliente\Anaconda3\envs\University\lib\site-packages\ipykernel_lau
ncher.py:6: MatplotlibDeprecationWarning:


The 'normed' kwarg was deprecated in Matplotlib 2.1 and will be removed in
3.1. Use 'density' instead.

C:\Users\cliente\Anaconda3\envs\University\lib\site-packages\ipykernel_lau
ncher.py:13: MatplotlibDeprecationWarning:


The 'normed' kwarg was deprecated in Matplotlib 2.1 and will be removed in
3.1. Use 'density' instead.
```



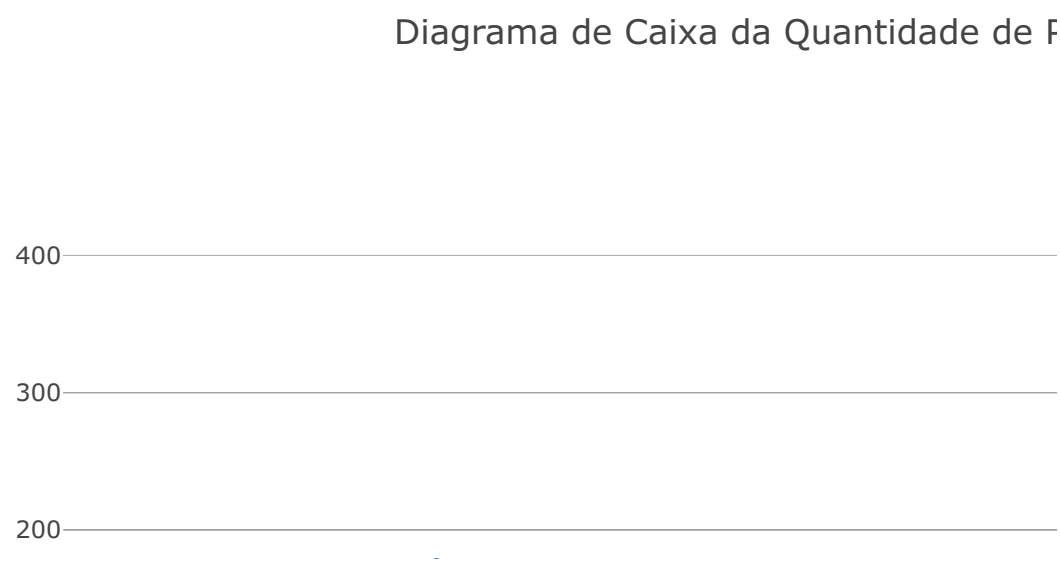
2.3 Diagrama de Caixa

O diagrama de caixa é construído utilizando as referências de valores mínimos e máximos, primeiro e terceiro quartil, mediana e os outliers da base de dados. Diferentemente do histograma que possibilita ter uma melhor visualização das médias e desvio padrão, o diagrama de caixa têm como objetivo estudar as medidas estatística e identificar os valores atípicos dentro do conjunto de dados.

2.3.1 Diagrama de Caixa da Quantidade de Palavras

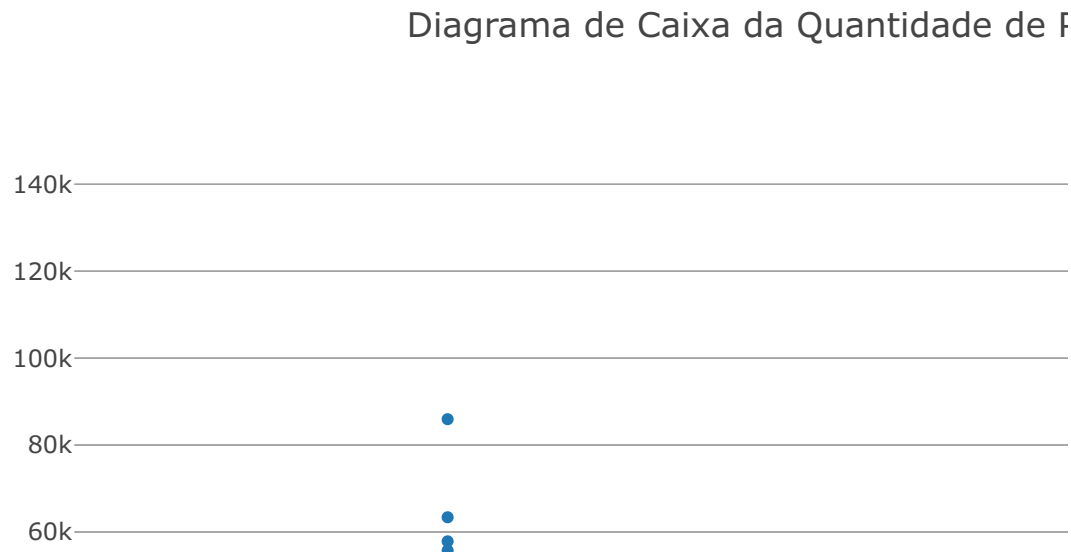
In [30]:

```
confiavel = go.Box(y=reliable.title_comprimento, name = 'confiável', boxmean=True)
nao_confiavel = go.Box(y=unreliable.title_comprimento, name = 'não confiáveis', boxmean
=True)
data = [confiavel, nao_confiavel]
layout = go.Layout(title = "Diagrama de Caixa da Quantidade de Palavras dos Títulos")
fig = go.Figure(data=data,layout=layout)
py.iplot(fig)
```



In [31]:

```
confiavel = go.Box(y=reliable.text_comprimento, name = 'confiável', boxmean=True)
nao_confiavel = go.Box(y=unreliable.text_comprimento, name = 'não confiáveis', boxmean=
True)
data = [confiavel, nao_confiavel]
layout = go.Layout(title = "Diagrama de Caixa da Quantidade de Palavras dos Textos")
fig = go.Figure(data=data,layout=layout)
py.iplot(fig)
```



2.4 Gráfico de Dispersão

Gráfico de Dispersão é utilizado para pontuar dados em um eixo vertical e horizontal com a intenção de exibir quanto uma variável é afetada por outra. Existem vários tipos de associações entre parâmetros que podem ser demonstradas pelo gráfico de dispersão. A relação pode ser positiva ou negativa (quando um cresce o outro decresce), fraca ou forte, linear ou não linear.

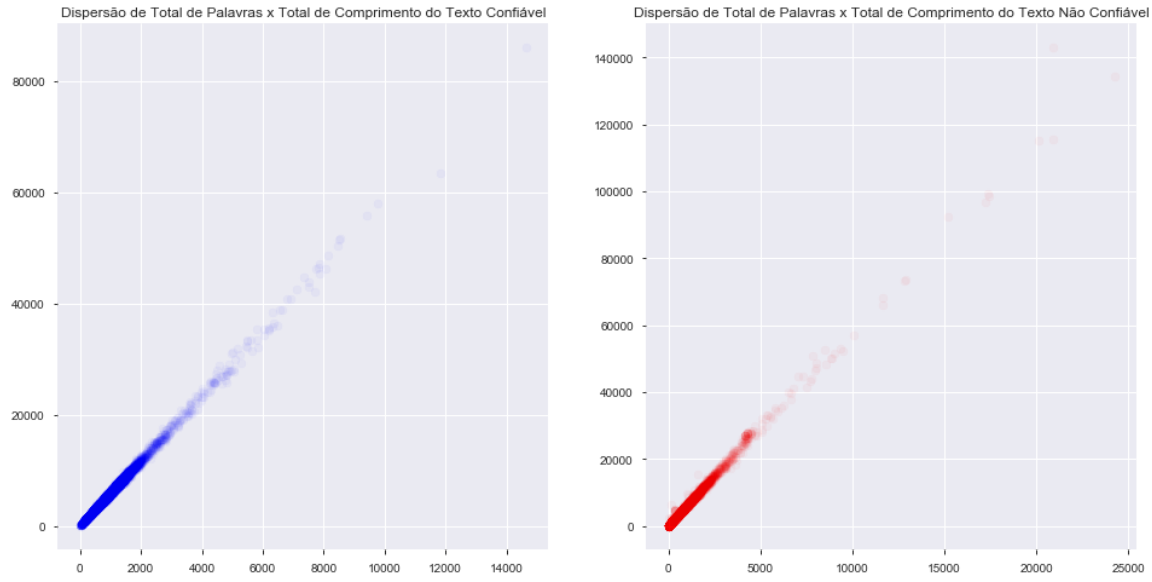
2.4.1 Dispersão Palavras x Comprimento

In [24]:

```
alpha = 0.03
plt.figure(figsize=(16,8))

# loc_x and loc_y
plt.subplot(121)
plt.scatter(reliable.text_num_palavras, reliable.text_comprimento, color='blue', alpha=
alpha)
plt.title('Dispersão de Total de Palavras x Total de Comprimento do Texto Confiável')

# lat and lon
plt.subplot(122)
plt.scatter(unreliable.text_num_palavras, unreliable.text_comprimento, color='red', alp
ha=alpha)
plt.title('Dispersão de Total de Palavras x Total de Comprimento do Texto Não Confiáve
l')
```

Out[24]:

```
Text(0.5, 1.0, 'Dispersão de Total de Palavras x Total de Comprimento do T
exto Não Confiável')
```



2.5 Matriz de correlação

A matriz de correlação mostra os valores de correlação de Pearson, que medem o grau de relação linear entre cada par de itens ou variáveis. Os valores de correlação podem cair entre -1 e +1. Uma vez feita a segmentação das variáveis, podemos efetuar uma verificação de correlação mais concisa entre as variáveis de cada segmento, assim obtendo uma compreensção dos dados positivamente e negativamente relacionadas.

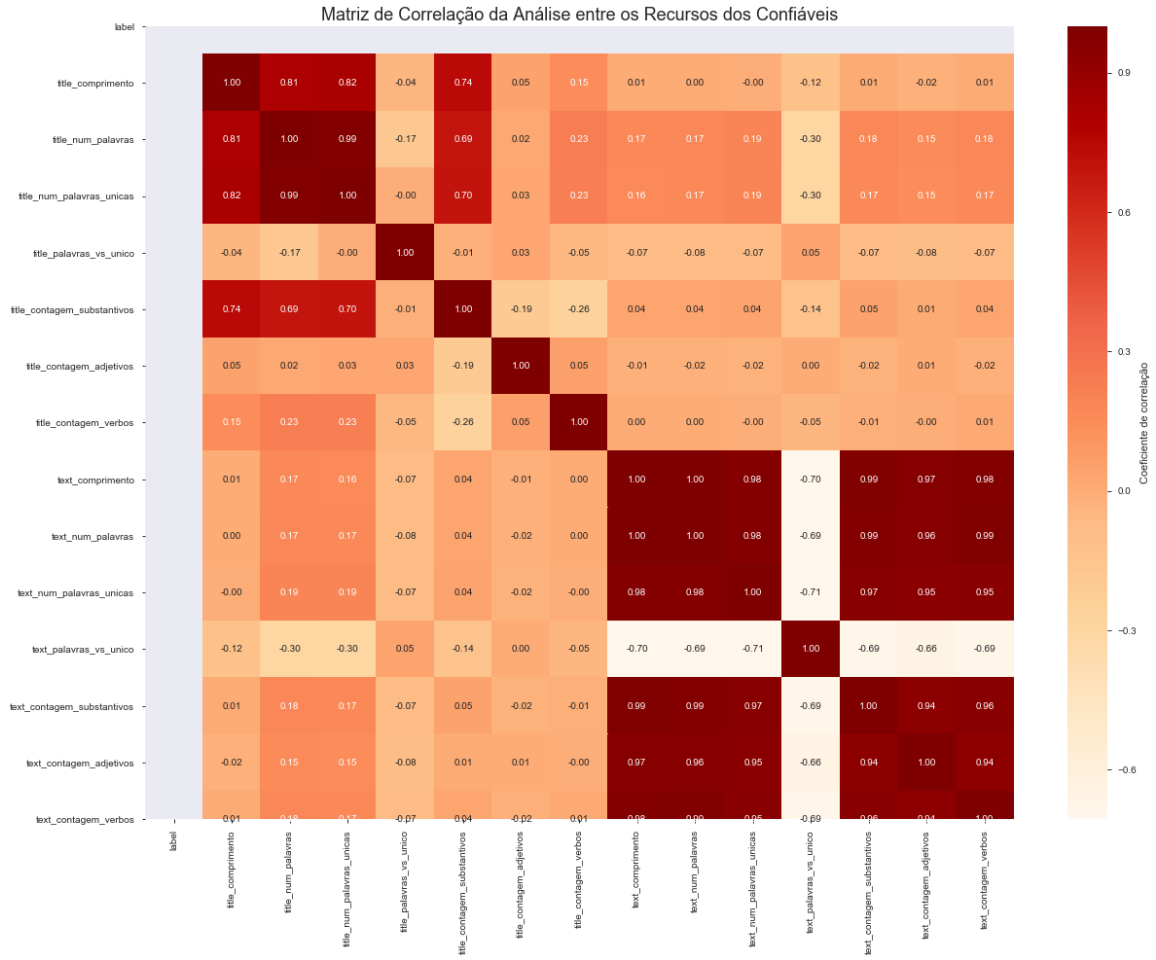
In [25]:

```
py.init_notebook_mode(connected=True)
pd.options.mode.chained_assignment = None
np.random.seed(13)

color = sns.color_palette()
%matplotlib inline
```

Com as variáveis extraídas por meio da análise por palavra, se tornou possível criar relacionamentos e identicar quais desses relacionamento se correlacionam entre si.

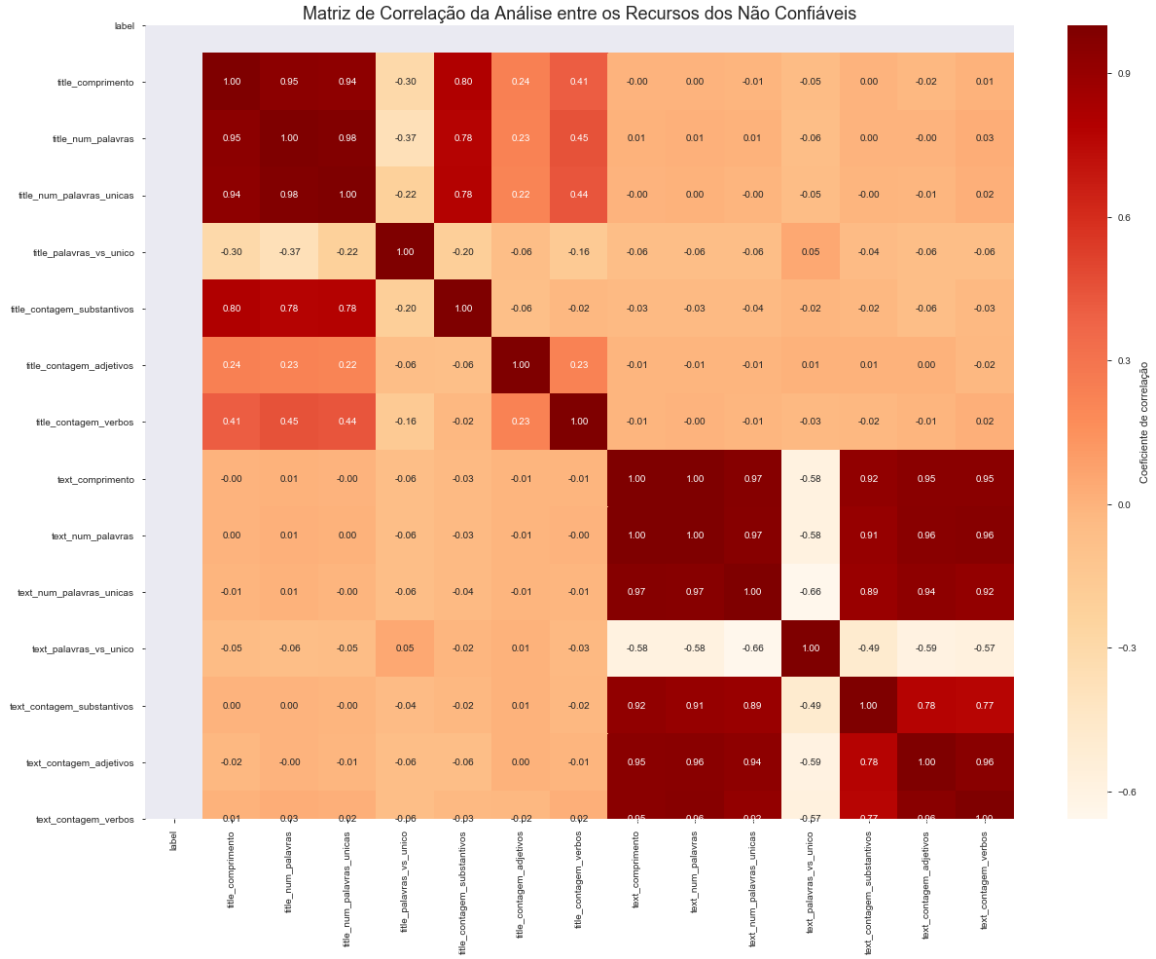
In [26]:

```
corr_reliable = reliable.drop(columns=['id', 'title', 'author', 'text', 'title_token',
'text_token','title_substantivos','title_adjetivos','title_verbos','text_substantivos',
                                'text_adjetivos','text_verbos'])

f, ax = plt.subplots(figsize= [20,15])
sns.heatmap(corr_reliable.corr(), annot=True, fmt=".2f", ax=ax,
            cbar_kws={'label': 'Coeficiente de correlação'}, cmap='OrRd')
ax.set_title("Matriz de Correlação da Análise entre os Recursos dos Confiáveis", fontsi
ze=18)
plt.show()
```



In [27]:

```
corr_unreliable = unreliable.drop(columns=['id', 'title', 'author', 'text', 'title_token','text_token','title_substantivos','title_adjetivos','title_verbos','text_substantivos',
                                           'text_adjetivos','text_verbos'])

f, ax = plt.subplots(figsize= [20,15])
sns.heatmap(corr_unreliable.corr(), annot=True, fmt=".2f", ax=ax,
            cbar_kws={'label': 'Coeficiente de correlação'}, cmap='OrRd')
ax.set_title("Matriz de Correlação da Análise entre os Recursos dos Não Confiáveis", fontsize=18)
plt.show()
```



In [28]:

```
train_data.columns
```

Out[28]:

```
Index(['id', 'title', 'author', 'text', 'label', 'title_token',
       'title_token_distint', 'title_comprimento', 'title_num_palavras',
       'title_num_palavras_unicas', 'title_palavras_vs_unico',
       'title_substantivos', 'title_adjetivos', 'title_verbos',
       'title_contagem_substantivos', 'title_contagem_adjetivos',
       'title_contagem_verbos', 'text_token', 'text_token_distint',
       'text_comprimento', 'text_num_palavras', 'text_num_palavras_unica
s',
       'text_palavras_vs_unico', 'text_substantivos', 'text_adjetivos',
       'text_verbos', 'text_contagem_substantivos', 'text_contagem_adjetiv
os',
       'text_contagem_verbos'],
      dtype='object')
```

REFERÊNCIAS

[1] SHAHEBAZ, Mohammad. Hand Crafted Feature Engineering For Insincerity. Disponível em: <https://www.kaggle.com/shaz13/feature-engineering-for-nlp-classification/notebook#Tagging-Parts-Of-Speech-And-More-Feature-Engineering;. Acesso em: 14 set. 2019.

INSTALLED VERSIONS

In [29]:

```
pd.show_versions()
```

```
INSTALLED VERSIONS
------------------
commit: None
python: 3.7.1.final.0
python-bits: 64
OS: Windows
OS-release: 10
machine: AMD64
processor: AMD64 Family 23 Model 8 Stepping 2, AuthenticAMD
byteorder: little
LC_ALL: None
LANG: None
LOCALE: None.None

pandas: 0.24.2
pytest: 5.0.1
pip: 19.1.1
setuptools: 41.0.1
Cython: None
numpy: 1.16.4
scipy: 1.2.1
pyarrow: None
xarray: None
IPython: 7.5.0
sphinx: None
patsy: None
dateutil: 2.8.0
pytz: 2019.1
blosc: None
bottleneck: None
tables: None
numexpr: None
feather: None
matplotlib: 3.1.1
openpyxl: None
xlrd: None
xlwt: None
xlsxwriter: None
lxml.etree: 4.3.4
bs4: None
html5lib: None
sqlalchemy: None
pymysql: None
psycopg2: None
jinja2: 2.10.1
s3fs: None
fastparquet: None
pandas_gbq: None
pandas_datareader: None
gcsfs: None
```