Aspect Extraction and Sentiment Analysis of Reddit discussion of RG&E

Part1: Aspect Extraction

In [43]:

```python
import nltk
from nltk.tokenize import RegexpTokenizer
from nltk.stem import WordNetLemmatizer,PorterStemmer
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.corpus import stopwords
import re
from nltk import FreqDist
lemmatizer = WordNetLemmatizer()
import pandas as pd
from textblob import TextBlob
import contractions
```

In [44]:
```python
from nltk.stem import WordNetLemmatizer
from nltk.corpus import wordnet
from nltk import word_tokenize, pos_tag
from IPython.display import display, HTML
df1 = pd.read_csv("RGE all Reddit submissions.csv")
df1.drop("Unnamed: 0",axis=1,inplace=True)
df1 = df1[df1["Content"].notna()]
df1
def get_wordnet_pos(tag):
    if tag.startswith('J'):
        return wordnet.ADJ
    elif tag.startswith('V'):
        return wordnet.VERB
    elif tag.startswith('N'):
        return wordnet.NOUN
    elif tag.startswith('R'):
        return wordnet.ADV
    else:
        return wordnet.NOUN

def lemmatize_passage(text):
    words = word_tokenize(text)
    pos_tags = pos_tag(words)
    lemmatizer = WordNetLemmatizer()
    lemmatized_words = [lemmatizer.lemmatize(word, get_wordnet_pos(tag)) for word, tag in pos_tags]
    lemmatized_sentence = ' '.join(lemmatized_words)
    return lemmatized_sentence
column_to_process = "Content"
df1[column_to_process] = df1[column_to_process].apply(lambda x:contractions.fix(str(x)))
df1[column_to_process] = df1[column_to_process].apply(lambda x: re.sub(r"\brep\b", "representative", x))
df1[column_to_process] = df1[column_to_process].apply(lambda x: re.sub(r"\bbilling\b", "bill", x))
df1[column_to_process] = df1[column_to_process].apply(lambda x: re.sub(r"\bcompany\b", "rg&e", x))
df1[column_to_process] = df1[column_to_process].apply(lambda x: re.sub(r"\bRGE\b", "rg&e", x))
df1[column_to_process] = df1[column_to_process].apply(lambda x: re.sub(r"\brge\b", "rg&e", x))



df1[column_to_process] = df1[column_to_process].apply(lambda x: lemmatize_passage(x))
replacers = {'rg & e':'RG&E','RG & E':'RG&E','Rge':'RG&E'}
df1[column_to_process] = df1[column_to_process].replace(replacers,regex=True)
display(HTML('<h1><p style="text-align:center;color:blue">Reddit Posts in tabular form</p></h1>'))
#print("Customer reviews in tabular form")
```

```
df1.reset_index(drop=True).head(10)
```

Reddit Posts in tabular form

Out[44]:

	Date	Title	Content
0	2014-01-23 00:00:00	Anyone else unable to log onto the RGE website?	I keep get this error page , I have try multip...
1	2015-07-18 00:00:00	RGE Scammers?	SW Area : I just have two people within the ho...
2	2015-10-06 00:00:00	Massive RGE bill increases?	Went from 90 to 245 buck and nothing have chan...
3	2016-10-22 00:00:00	PSA: If someone comes around and says they are...	They be lie to you to try and get you to sign ...
4	2017-03-13 00:00:00	Has anyone had power while their street is sti...	RG&E keep change my street recovery date and i...
5	2017-04-18 00:00:00	RGE budget or standard payment plans	I be move to a new apt next month . I have be ...
6	2017-08-01 00:00:00	RGE Budget Billing	Does anyone use RG & amp ; E 's budget bill ? ...
7	2017-09-20 00:00:00	Does RGE offer discounts for full time student...	I be originally from NJ , where PGE offer disc...
8	2018-05-09 00:00:00	RGE never checked my meter	[delete]
9	2018-12-29 00:00:00	High RGE bill	Anyone else 's electric bill nearly double thi...

In [45]:

```python
df = pd.read_csv("RGE all Reddit comments.csv")
df.drop("Unnamed: 0",axis=1,inplace=True)
column_to_process = "Comment"
df[column_to_process] = df[column_to_process].apply(lambda x:contractions.fix(x))
df[column_to_process] = df[column_to_process].apply(lambda x: re.sub(r"\brep\b", "representative", x))
df[column_to_process] = df[column_to_process].apply(lambda x: re.sub(r"\bbilling\b", "bill", x))
df[column_to_process] = df[column_to_process].apply(lambda x: re.sub(r"\bcompany\b", "rg&e", x))
df[column_to_process] = df[column_to_process].apply(lambda x: re.sub(r"\bRGE\b", "rg&e", x))
df[column_to_process] = df[column_to_process].apply(lambda x: re.sub(r"\brge\b", "rg&e", x))



df[column_to_process] = df[column_to_process].apply(lambda x: lemmatize_passage(x))
replacers = {'rg & e':'RG&E','RG & E':'RG&E','Rge':'RG&E'}
df[column_to_process] = df[column_to_process].replace(replacers,regex=True)
display(HTML('<h1><p style="color:blue">Comments to Reddit Posts </p></h1>'))
df
```

Comments to Reddit Posts

Out[45]:

```
                    Date                                          Comment
  0  2014-01-23 00:00:00                                  Nope , I can get in .
  1  2014-01-23 00:00:00   For some reason , it be the web browser you be...
  2  2014-01-24 00:00:00    I encounter the same issue a month or two ago ...
  3  2014-01-24 00:00:00                      Working fine with Chrome here/now .
  4  2015-07-18 00:00:00  They be definitely scammer and swoop through t...
...                  ...                                                  ...
777  2024-06-26 21:53:18   Oh 100 % , but I can not move some of these th...
778  2024-06-26 23:02:48       Some place pay for that . First place I can th...
779  2024-06-26 23:02:48    I know , I be look for someone who want to poc...
780  2024-06-27 00:49:04          If the AC be non functional , take it to the e...
781  2024-06-27 06:17:58       I have a business card of a guy that have pick...

782 rows × 2 columns
```

In [46]:

```python
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
def get_sentiment_vader(words):
    analyzer = SentimentIntensityAnalyzer()
    #print("GGGG",words)
    vs = analyzer.polarity_scores(words)

    return vs["compound"]

def get_sentiment_blob(words):
    blob = TextBlob(words)
    return(blob.sentiment.polarity)
```

In [47]:
```python
#!python -m spacy download en_core_web_lg
import nltk
#nltk.download('vader_lexicon')

import spacy
nlp = spacy.load("en_core_web_lg")

from nltk.sentiment.vader import SentimentIntensityAnalyzer
sid = SentimentIntensityAnalyzer()


def find_sentiment(doc):
    # find roots of all entities in the text
    ner_heads = {ent.root.idx: ent for ent in doc.ents}
    #print("AA",doc)
    #print("CCC",doc.ents)
    #print("BB",ner_heads)
    rule3_pairs = []
    for token in doc:
        children = token.children
        A = "999999"
        M = "999999"
        add_neg_pfx = False
        for child in children:
            if(child.dep_ == "nsubj" and not child.is_stop): # nsubj is nominal subject
                if child.idx in ner_heads:
                    A = ner_heads[child.idx].text
                else:
                    A = child.text
            if(child.dep_ == "acomp" and not child.is_stop): # acomp is adjectival complement
                M = child.text
            # example - 'this could have been better' -> (this, not better)
            if(child.dep_ == "aux" and child.tag_ == "MD"): # MD is modal auxiliary
                neg_prefix = "not"
                add_neg_pfx = True
            if(child.dep_ == "neg"): # neg is negation
                neg_prefix = child.text
                add_neg_pfx = True
        if (add_neg_pfx and M != "999999"):
            M = neg_prefix + " " + M
        if(A != "999999" and M != "999999"):
            #print("AA",doc)
            doc = str(doc)
            phrase = doc[doc.find(A):doc.find(M)]+M
            #print("PPP",phrase)
            #print("MM1",sid.polarity_scores(phrase)['compound'],sid.polarity_scores(M)['compound'],get_s
            #print("MM2",M,sid.polarity_scores(M)['compound'],get_sentiment_vader(M))
            #print(A,M)
            #print("********************************************************************")
            if sid.polarity_scores(phrase)['compound']<0:
                sentiment = sid.polarity_scores(phrase)['compound']
            if sid.polarity_scores(M)['compound']<0:
                sentiment = sid.polarity_scores(M)['compound']
            if get_sentiment_vader(phrase)<0:
                sentiment = get_sentiment_vader(phrase)
            if get_sentiment_vader(M)<0:
                sentiment = get_sentiment_vader(M)
            else:
                sentiment = sid.polarity_scores(phrase)['compound']
            rule3_pairs.append((A, M, sentiment))
    return rule3_pairs

```

In [48]:

```python
from nltk.tokenize import sent_tokenize
import numpy as np
aspect_store = []
aspect_data = {}
data = df["Comment"].tolist()+df1["Content"].tolist()
for i in data:
    sent_tok = sent_tokenize(i)
    #print(sent_tok)
    for j in sent_tok:
        aspect = find_sentiment(nlp(j))
        if len(aspect)>0:
            #print(aspect)
            aspect_store.append(aspect)
            for i in range(len(aspect)):
                if aspect[i][0].lower() in aspect_data:
                    aspect_data[aspect[i][0].lower()] +=[aspect[i][1].lower()]
                else:
                    aspect_data[aspect[i][0].lower()] =[aspect[i][1].lower()]


#aspect_data
```

In [49]:

```python
aspect_datas = sorted(aspect_data.items(), key= lambda x: len(x[1]), reverse=True)
#print(aspect_datas)
data = []
aspect_term =[]
from collections import Counter
for i in aspect_datas:
    #print(i[0])
    aspect_term.append(i[0])
    data.append(Counter(i[1]))
    #print(Counter(i[1]))
    #print(list(Counter(i[1]).keys()))
    #print('*****************************')
```

Top Aspect phrases used by customers in the review:

1. Bill
2. RG&E
3. Charge
4. Number

In [50]:
```python
config = {
  'toImageButtonOptions': {
    'format': 'svg', # one of png, svg, jpeg, webp
    'filename': 'custom_image',
    'height': 500,
    'width': 700,
    'scale': 5 # Multiply title/legend/axis/canvas sizes by this factor
  },'modeBarButtonsToAdd': ['drawline',
                                        'drawopenpath',
                                        'drawclosedpath',
                                        'drawcircle',
                                        'drawrect',
                                        'eraseshape'
                                       ]}

```

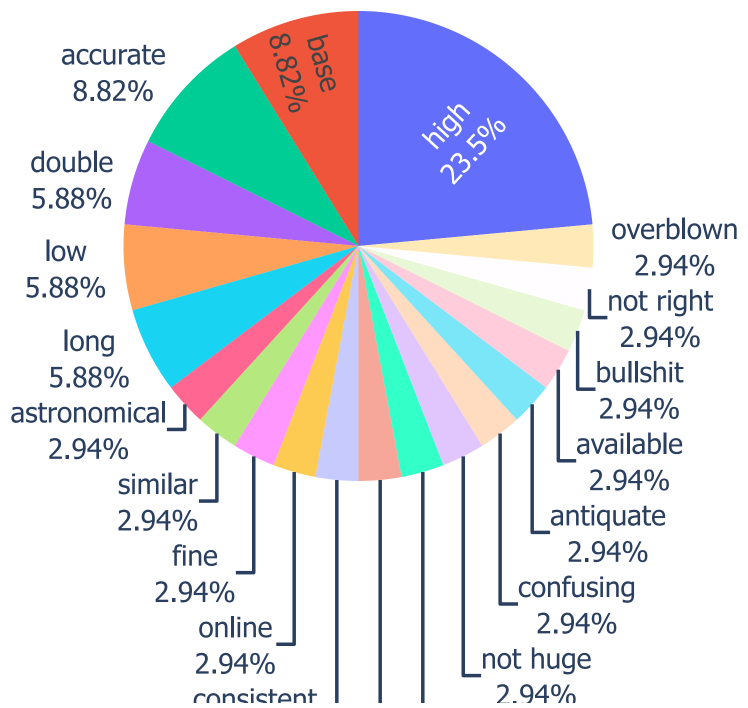
In [51]:

```
import plotly.express as px
import plotly.graph_objects as go
import matplotlib.pyplot as plt

k = 0 #defines which index in the list to plot
label = list(data[k].keys())
#rep = representative = {'friendly': 2, 'unprofessional': 2, 'able': 1, 'rude': 3, 'not helpful': 2, 'hum
val = list(data[k].values())
#label = list(rep.keys())
#val = list(rep.values())

#print(label)
val
fig = go.Figure(data=[go.Pie(labels=label, values=val, textinfo='label+percent',insidetextorientation='ra
fig.update_layout(legend=dict({'traceorder': 'normal'}),
                  legend_title_text="Description of term "+'"'+aspect_term[k].capitalize()+'"',
                  title ="Descriptions of the term "+'"'+aspect_term[k].capitalize()+'"')
fig.update_layout(title_x=0.5)
fig.update_layout(showlegend=False)
fig.update_layout(title_text="Descriptions used for the term <span style='color:orangered'>%s </span>"%as
fig.update_layout(
    font_family="tahoma",
    font_size=14,
    legend_title_font_color="green"
)
#fig.write_image("name.svg")
fig.show(config=config)
```

Descriptions used for the term BILL



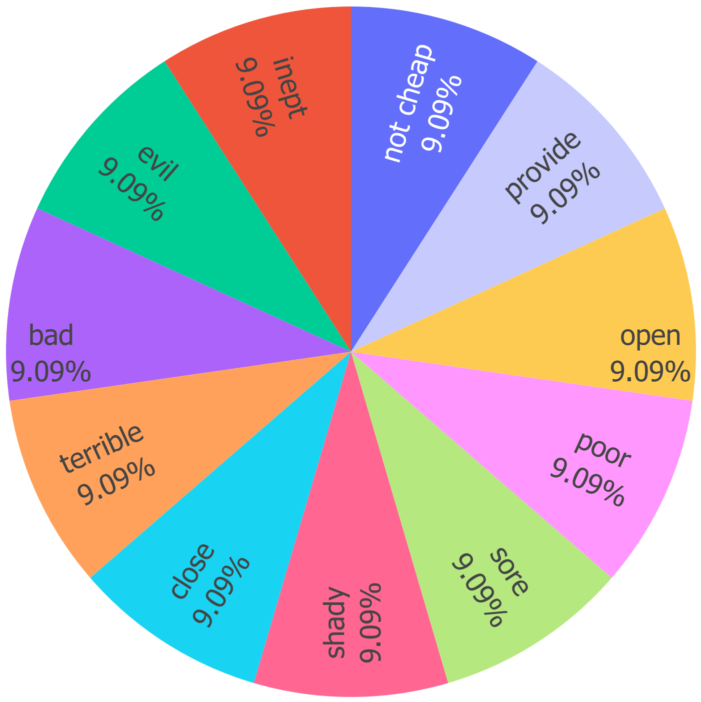
In [52]:

```
import plotly.express as px
import plotly.graph_objects as go
import matplotlib.pyplot as plt

k = 1 #defines which index in the list to plot
label = list(data[k].keys())
#rep = representative = {'friendly': 2, 'unprofessional': 2, 'able': 1, 'rude': 3, 'not helpful': 2, 'hum
val = list(data[k].values())
#label = list(rep.keys())
#val = list(rep.values())

#print(label)
val
fig = go.Figure(data=[go.Pie(labels=label, values=val, textinfo='label+percent',insidetextorientation='ra
fig.update_layout(legend=dict({'traceorder': 'normal'}),
                  legend_title_text="Description of term "+'"'+aspect_term[k].capitalize()+'"',
                  title ="Descriptions of the term "+'"'+aspect_term[k].capitalize()+'"')
fig.update_layout(title_x=0.5)
fig.update_layout(showlegend=False)
fig.update_layout(title_text="Descriptions used for the term <span style='color:orangered'>%s </span>"%as
fig.update_layout(
    font_family="tahoma",
    font_size=14,
    legend_title_font_color="green"
)
#fig.write_image("name.svg")
fig.show(config=config)
```

Descriptions used for the term RG&E

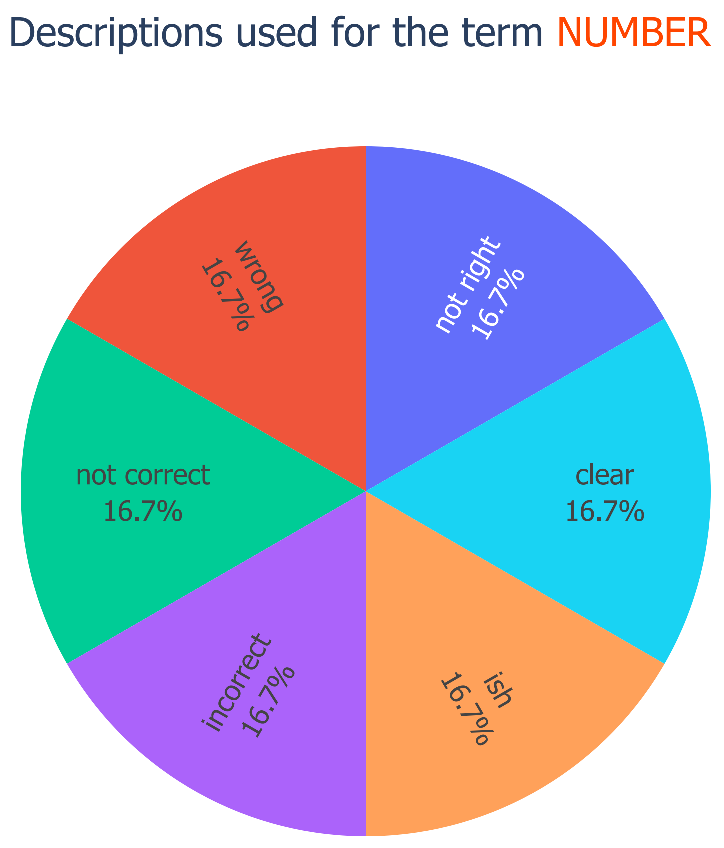


In [53]:

```
import plotly.express as px
import plotly.graph_objects as go
import matplotlib.pyplot as plt

k = 3 #defines which index in the list to plot
label = list(data[k].keys())
#rep = representative = {'friendly': 2, 'unprofessional': 2, 'able': 1, 'rude': 3, 'not helpful': 2, 'hum
val = list(data[k].values())
#label = list(rep.keys())
#val = list(rep.values())

#print(label)
val
fig = go.Figure(data=[go.Pie(labels=label, values=val, textinfo='label+percent',insidetextorientation='ra
fig.update_layout(legend=dict({'traceorder': 'normal'}),
                  legend_title_text="Description of term "+'"'+aspect_term[k].capitalize()+'"',
                  title ="Descriptions of the term "+'"'+aspect_term[k].capitalize()+'"')
fig.update_layout(title_x=0.5)
fig.update_layout(showlegend=False)
fig.update_layout(title_text="Descriptions used for the term <span style='color:orangered'>%s </span>"%as
fig.update_layout(
    font_family="tahoma",
    font_size=14,
    legend_title_font_color="green"
)
#fig.write_image("name.svg")
fig.show(config=config)
```



```
In [ ]:  1
```

Summary: the descriptions used for each term shows that customers are unhappy with the Copany and would

love to have some changes made in some areas.

Part2: Aspect Sentiment analysis

In [55]:
```
def get_sentiment(x):
    if x>0:
        return "positive"
    elif x<0:
        return "negative"
    else:
        return "neutral"
```

In [56]:
```
from collections import defaultdict
sentiments = defaultdict(list)
for i in aspect_store:
    sentiments[i[0][0]] += [get_sentiment(i[0][2])]
```

In [57]:

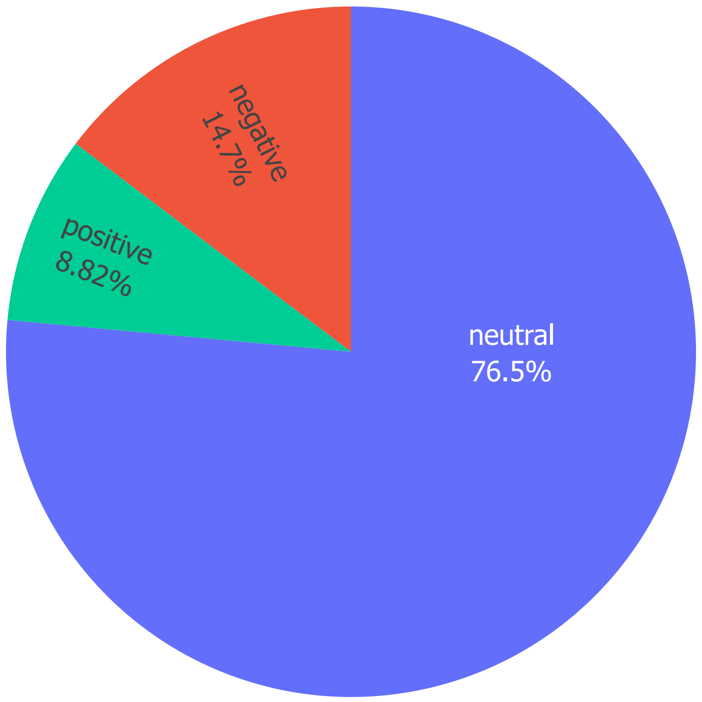
```python
aspect_datas = sorted(sentiments.items(), key= lambda x: len(x[1]), reverse=True)
#print(aspect_datas)
data = []
aspect_term =[]
from collections import Counter
for i in aspect_datas:
    #print(i[0])
    aspect_term.append(i[0])
    data.append(Counter(i[1]))
    #print(Counter(i[1]))
    #print(list(Counter(i[1]).keys()))
    #print('*****************************')
```

In [58]:

```python
import plotly.express as px
import plotly.graph_objects as go
import matplotlib.pyplot as plt

k = 0 #defines which index in the list to plot
label = list(data[k].keys())
val = list(data[k].values())
#print(label)
val
fig = go.Figure(data=[go.Pie(labels=label, values=val, textinfo='label+percent',insidetextorientation='ra
fig.update_layout(legend=dict({'traceorder': 'normal'}),legend_title_text='"'+aspect_term[k].capitalize()
fig.update_layout(title_x=0.5)
fig.update_layout(showlegend=False)
fig.update_layout(title_text="Sentiments associated with the term <span style='color:orangered'>%s </span
fig.update_layout(
    font_family="tahoma",
    font_size=14,
    legend_title_font_color="green"
)
#fig.write_image("name.svg")
fig.show(config=config)
```

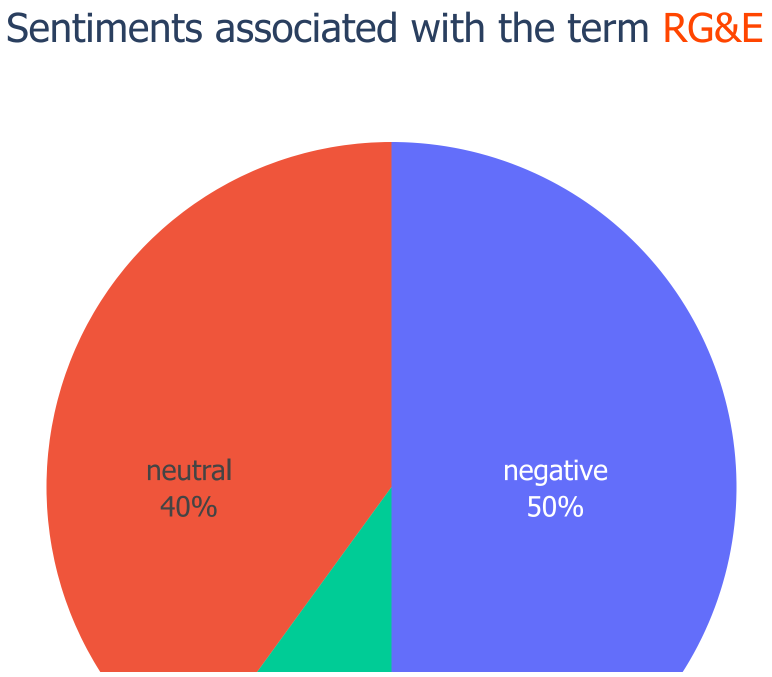
Sentiments associated with the term BILL



In [64]:

```
import plotly.express as px
import plotly.graph_objects as go
import matplotlib.pyplot as plt

k = 1 #defines which index in the list to plot
label = list(data[k].keys())
val = list(data[k].values())
#print(label)
val
fig = go.Figure(data=[go.Pie(labels=label, values=val, textinfo='label+percent',insidetextorientation='ra
fig.update_layout(legend=dict({'traceorder': 'normal'}),legend_title_text='"'+aspect_term[k].capitalize()
fig.update_layout(title_x=0.5)
fig.update_layout(showlegend=False)
fig.update_layout(title_text="Sentiments associated with the term <span style='color:orangered'>%s </span
fig.update_layout(
    font_family="tahoma",
    font_size=14,
    legend_title_font_color="green"
)
#fig.write_image("name.svg")
fig.show(config=config)
```

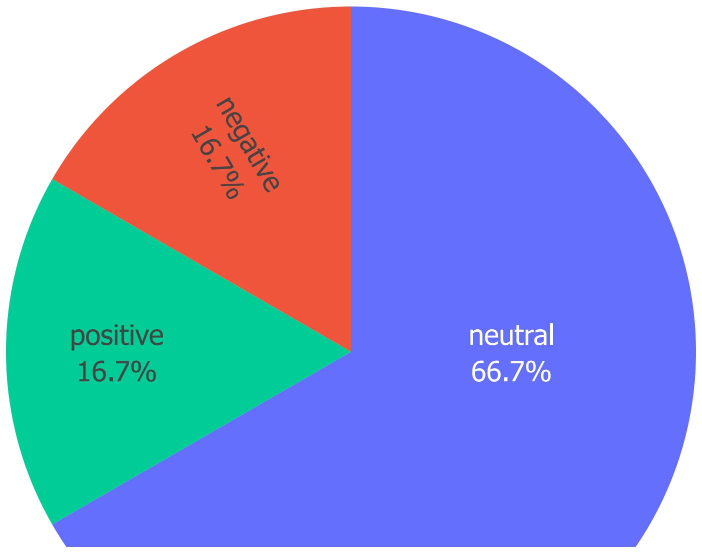


In [60]:

```python
import plotly.express as px
import plotly.graph_objects as go
import matplotlib.pyplot as plt

k = 2 #defines which index in the list to plot
label = list(data[k].keys())
val = list(data[k].values())
#print(label)
val
fig = go.Figure(data=[go.Pie(labels=label, values=val, textinfo='label+percent',insidetextorientation='ra
fig.update_layout(legend=dict({'traceorder': 'normal'}),
                  legend_title_text="Description of term "+'"'+aspect_term[k].capitalize()+'"',
                  title ="Descriptions of the term "+'"'+aspect_term[k].capitalize()+'"')
fig.update_layout(title_x=0.5)
fig.update_layout(showlegend=False)
fig.update_layout(title_text="Distribution of the sentiments used for the term  <span style='color:orange
fig.update_layout(
    font_family="tahoma",
    font_size=14,
    legend_title_font_color="green"
)
#fig.write_image("name.svg")
fig.show(config=config)
```

Distribution of the sentiments used for the term PEOPLE

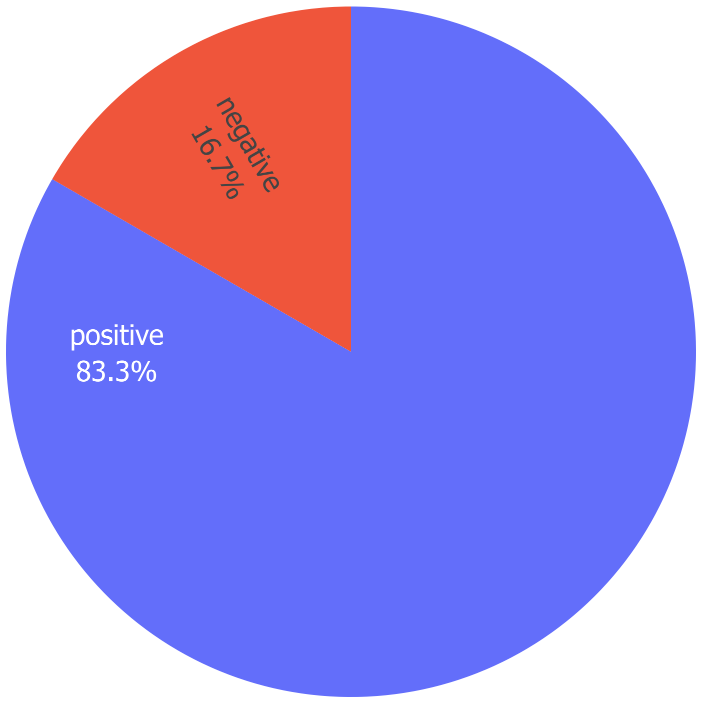


In [61]:

```
import plotly.express as px
import plotly.graph_objects as go
import matplotlib.pyplot as plt

k = 3 #defines which index in the list to plot
label = list(data[k].keys())
val = list(data[k].values())
#print(label)
val
fig = go.Figure(data=[go.Pie(labels=label, values=val, textinfo='label+percent',insidetextorientation='ra
fig.update_layout(legend=dict({'traceorder': 'normal'}),legend_title_text='"'+aspect_term[k].capitalize()
fig.update_layout(title_x=0.5)
fig.update_layout(showlegend=False)
fig.update_layout(title_text="Sentiments associated with the term <span style='color:orangered'>%s </span
fig.update_layout(
    font_family="tahoma",
    font_size=14,
    legend_title_font_color="green"
)
#fig.write_image("name.svg")
fig.show(config=config)
```

Sentiments associated with the term NUMBER



Summary:

Sentiments associated with each aspect shows that there are mostly more negative sentiments than positive

In [62]:
```
!jupyter nbconvert --to slides  --no-input Aspect_based_sentiment_analyzer_using_multi_approach.ipynb
```

```
[NbConvertApp] Converting notebook Aspect_based_sentiment_analyzer_using_multi_approach.ipynb to slides
[NbConvertApp] Writing 657964 bytes to Aspect_based_sentiment_analyzer_using_multi_approach.slides.html
```