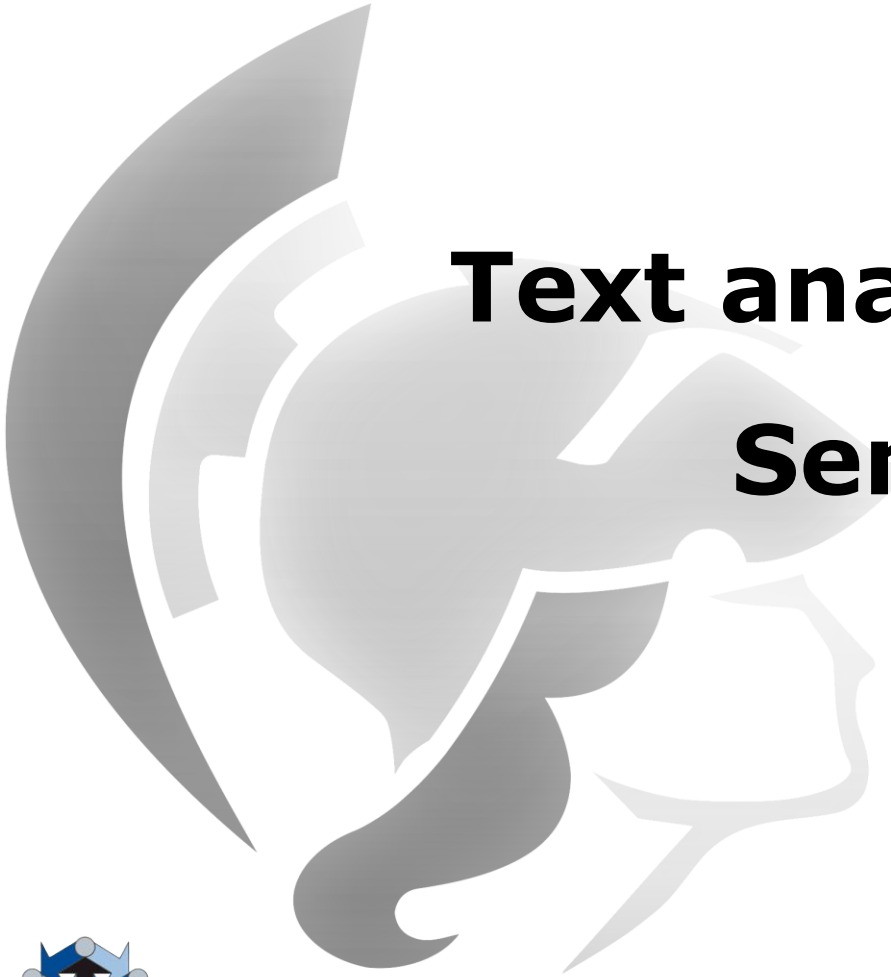
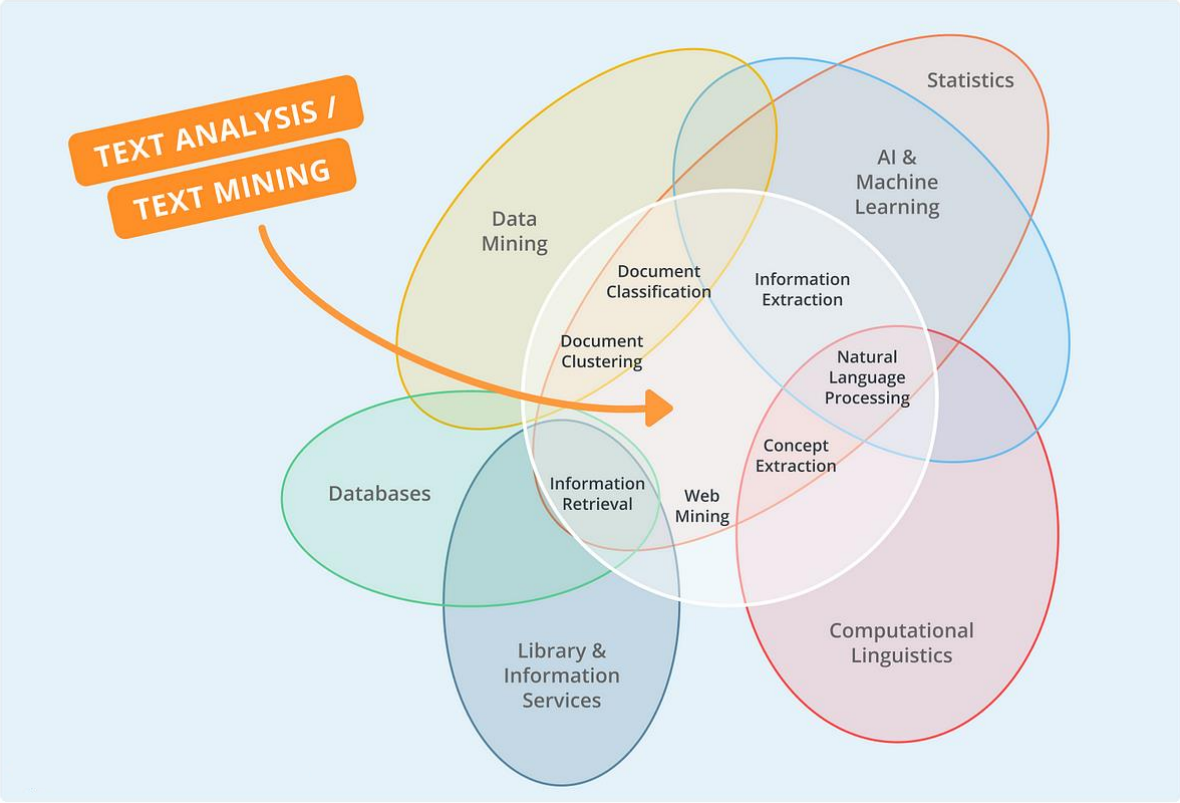


Text analytics e Aspect Based Sentiment Analysis

Text analytics e Sentiment analysis

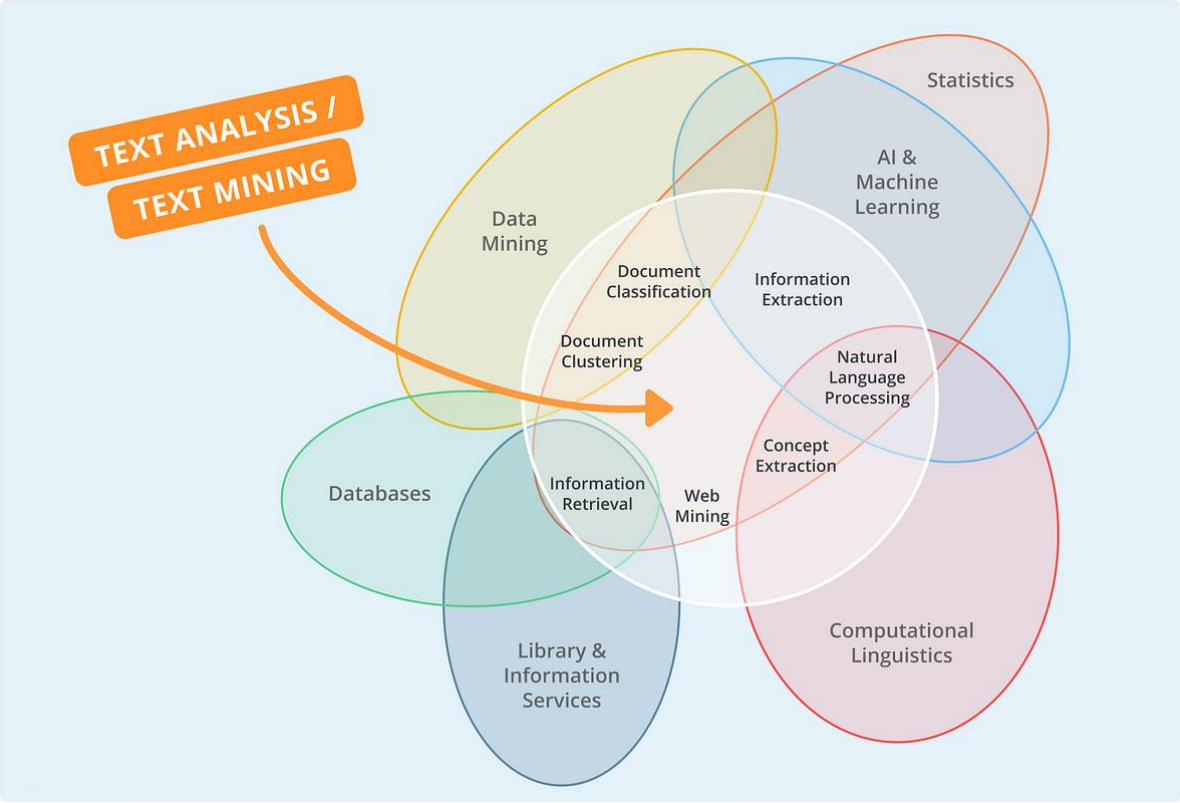
Text analytics

Consiste no uso de algoritmos de processamento de linguagem natural (NLP), aprendizado de máquina e estatística para processar grandes volumes de texto não estruturado e assim obter insights e padrões.



Text analytics

Esses insights podem se basear na identificação de padrões relacionados a palavras-chave, tópicos, sentimentos expressos, entidades nomeadas (como pessoas, locais e organizações), entre outros.



Text analytics – tipos

Text analytics – tipos

Vs

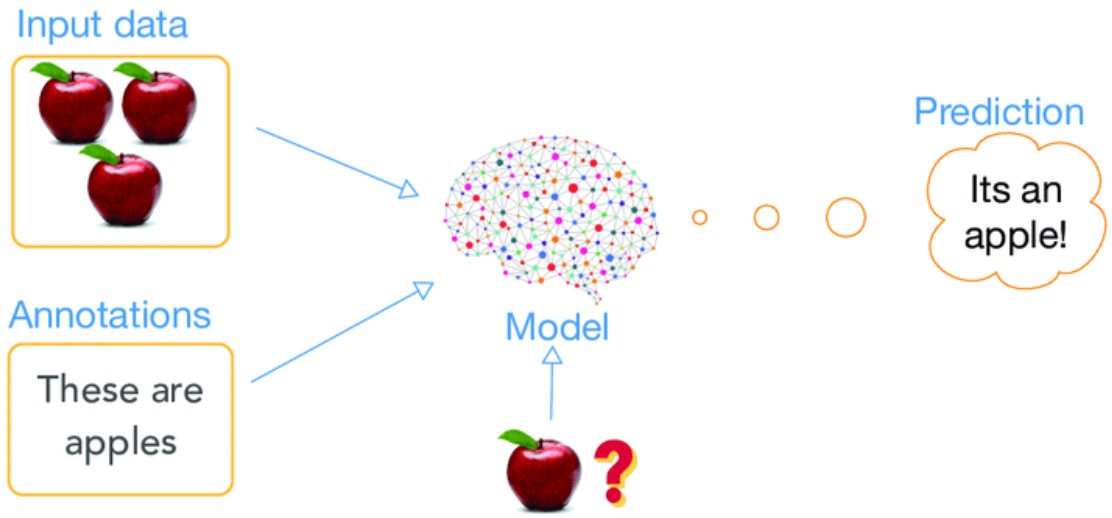
Supervised Learning

Unsupervised Learning

Text analytics – tipos

Aprendizado supervisionado

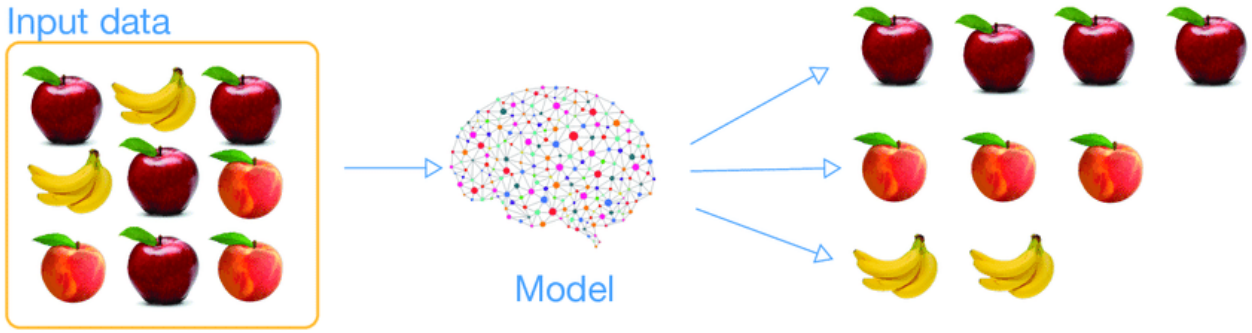
No text analytics supervisionado, um conjunto de dados é preparado com textos de entrada e as correspondentes saídas desejadas, também conhecidas como rótulos. Os rótulos indicam a classe, categoria ou resultado esperado para cada texto



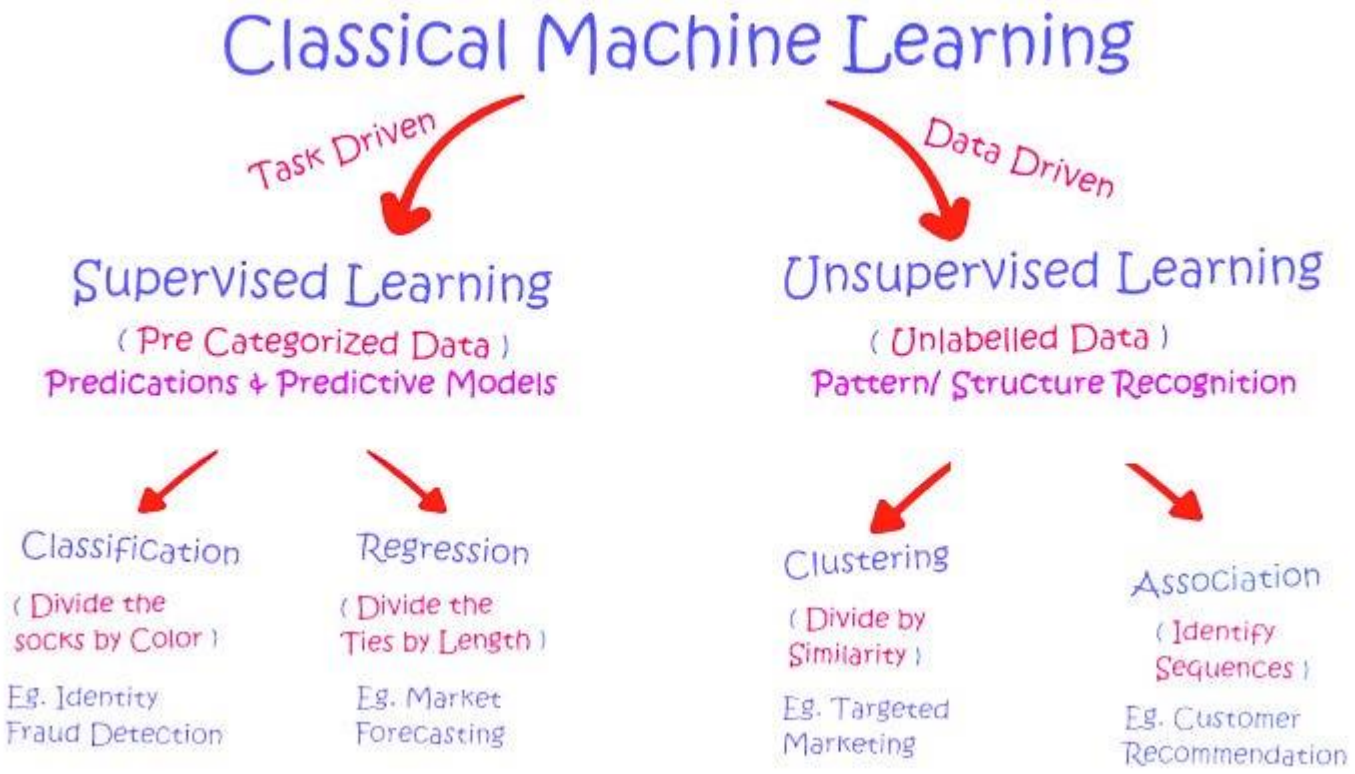
Text analytics – tipos

Aprendizado não-supervisionado

No text analytics não-supervisionado, algoritmos são aplicados a dados não rotulados, ou seja, não é necessário fornecer rótulos ou saídas desejadas durante o processo de treinamento. Nessa abordagem, o objetivo é explorar a estrutura, os padrões e as relações nos dados textuais sem a necessidade de supervisão humana.



Text analytics – tipos

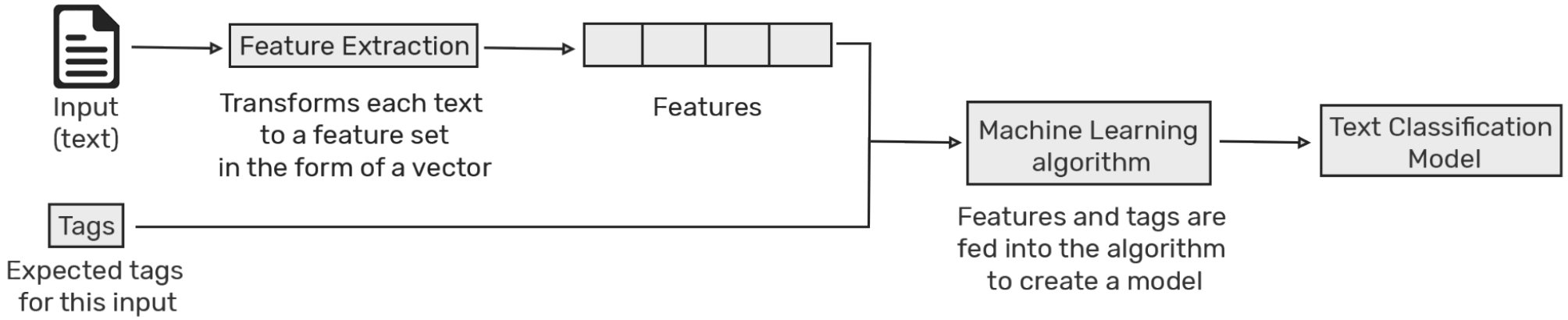


Text analytics – algumas abordagens

Text analytics – algumas abordagens

Text classification

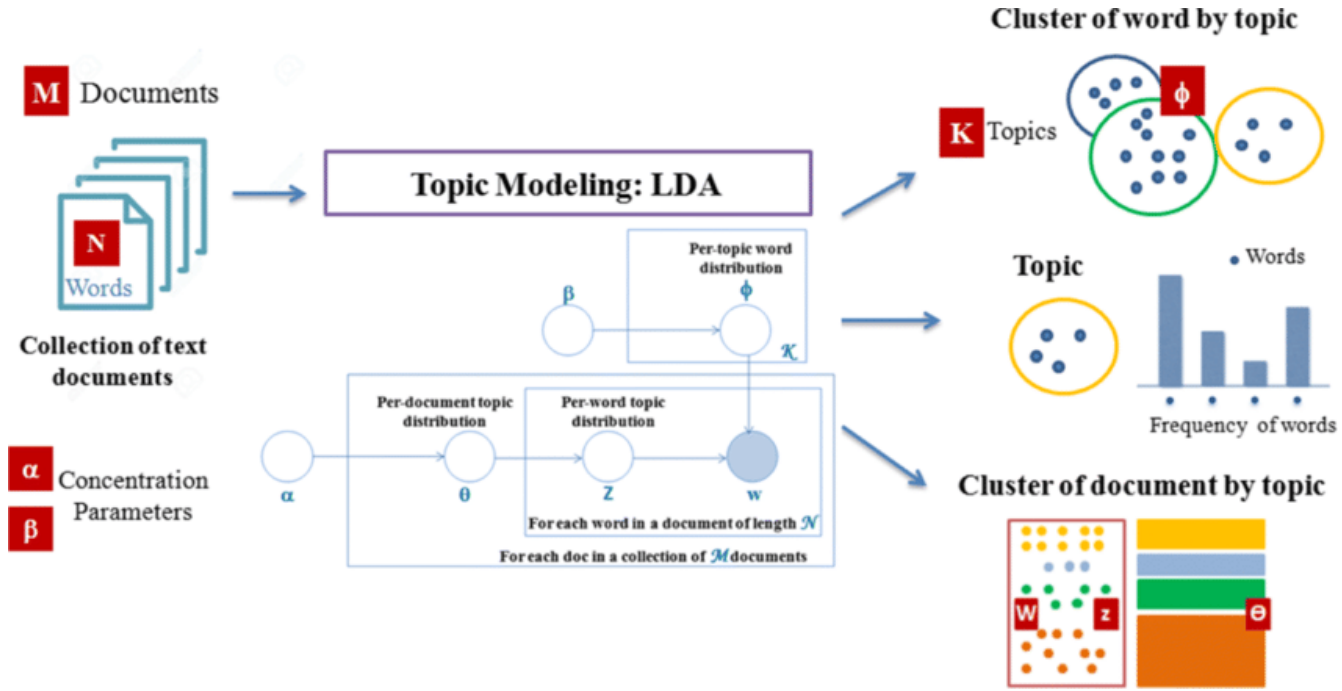
Envolve atribuir automaticamente categorias ou rótulos a um texto com base em seu conteúdo. Pode ser usada para classificar documentos em categorias pré-definidas, como notícias, esportes, entretenimento, etc., ou para realizar triagem automática de e-mails, por exemplo. *(supervisionado)*



Text analytics – algumas abordagens

Topic modelling

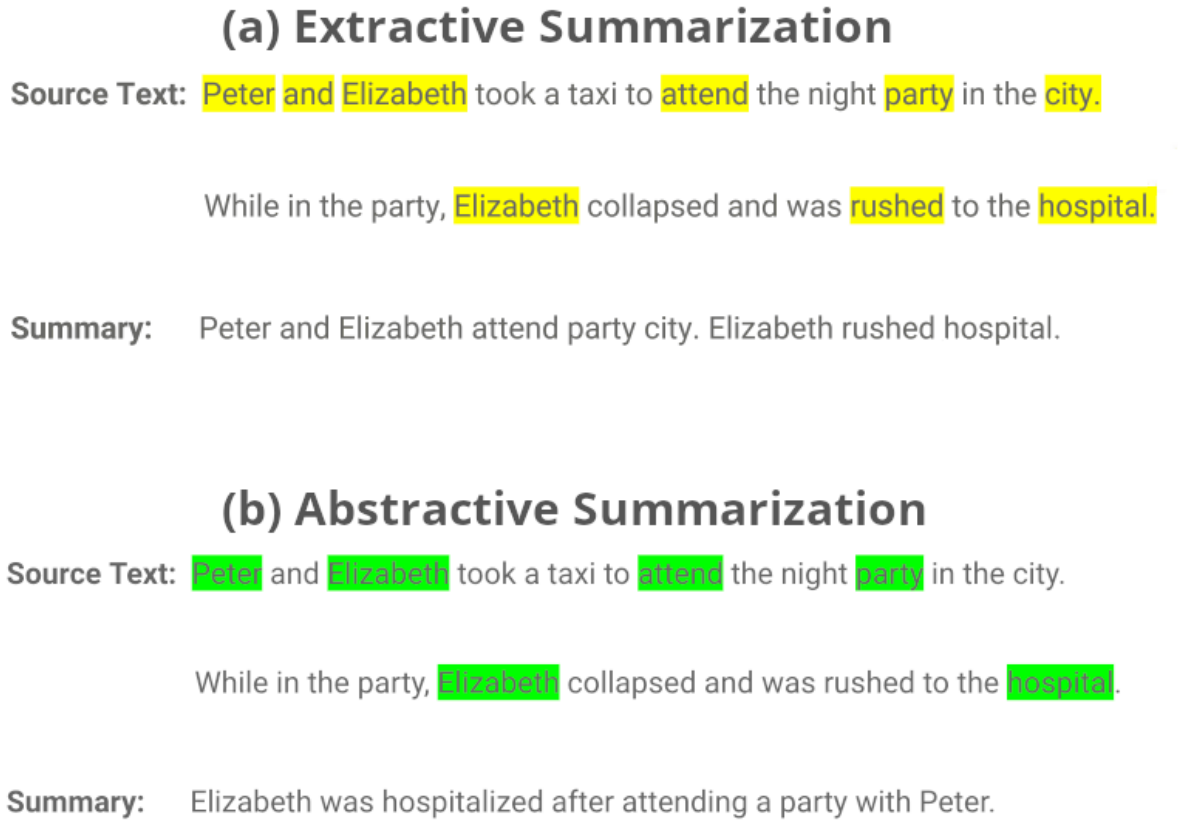
Envolve identificar e agrupar automaticamente os textos em tópicos ou temas semelhantes. Pode ser realizada por meio de algoritmos que identificam as palavras e os padrões mais relevantes em um conjunto de documentos para inferir os tópicos discutidos. *(não supervisionado)*



Text analytics – algumas abordagens

Text summarization

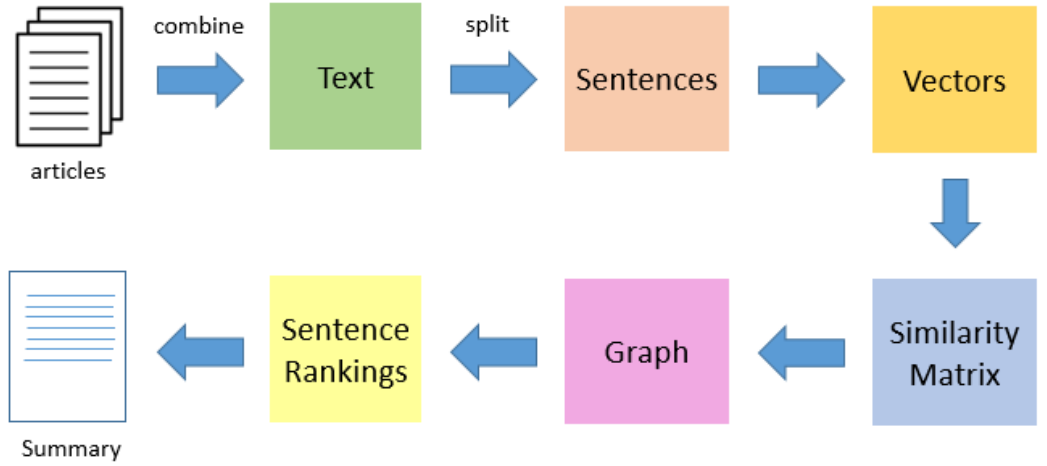
Visa criar um resumo conciso e relevante de um texto longo ou de vários documentos. A sumarização automática pode ser feita de forma extrativa (selecionando as frases mais importantes do texto original) ou abstrativa (gerando frases novas que capturam o conteúdo principal). *(supervisionado)*



Text analytics – algumas abordagens

Text summarization

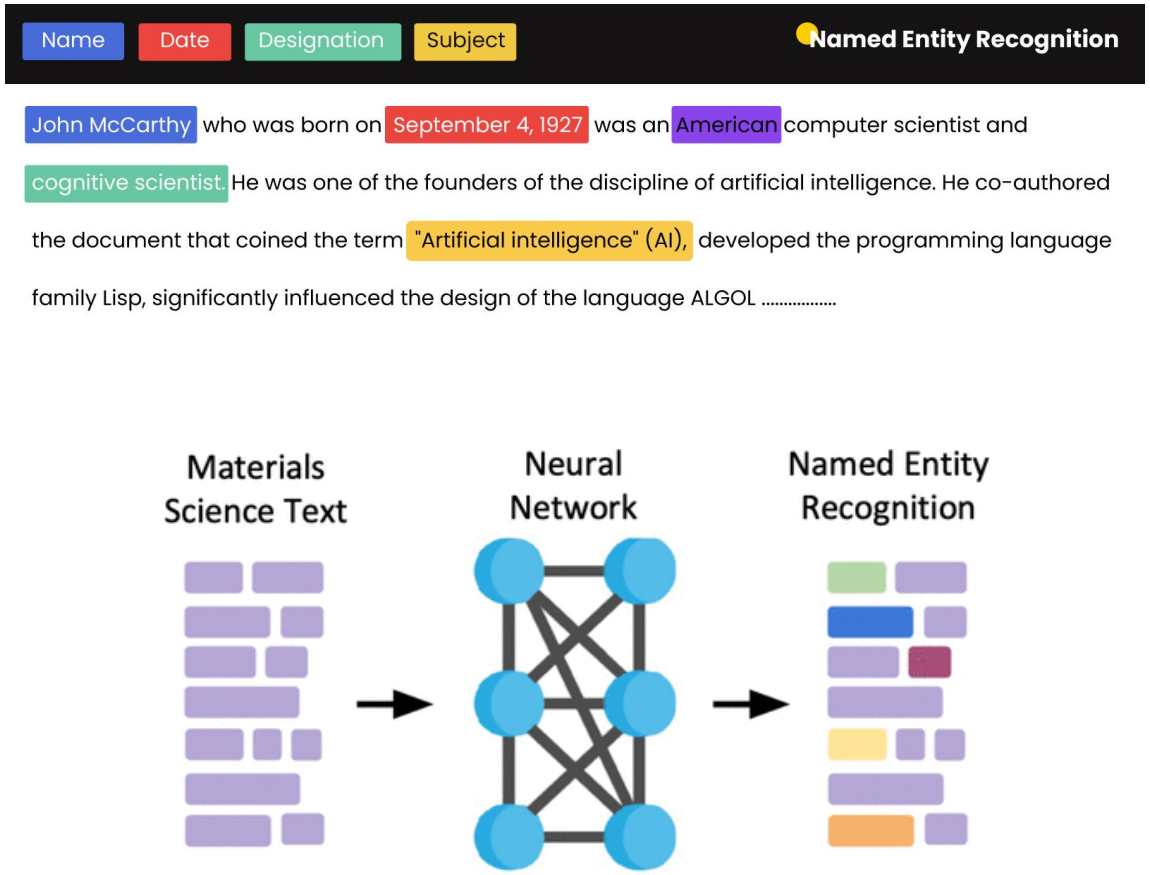
Visa criar um resumo conciso e relevante de um texto longo ou de vários documentos. A sumarização automática pode ser feita de forma extrativa (selecionando as frases mais importantes do texto original) ou abstrativa (gerando frases novas que capturam o conteúdo principal). *(supervisionado)*



Text analytics – algumas abordagens

Entity recognition

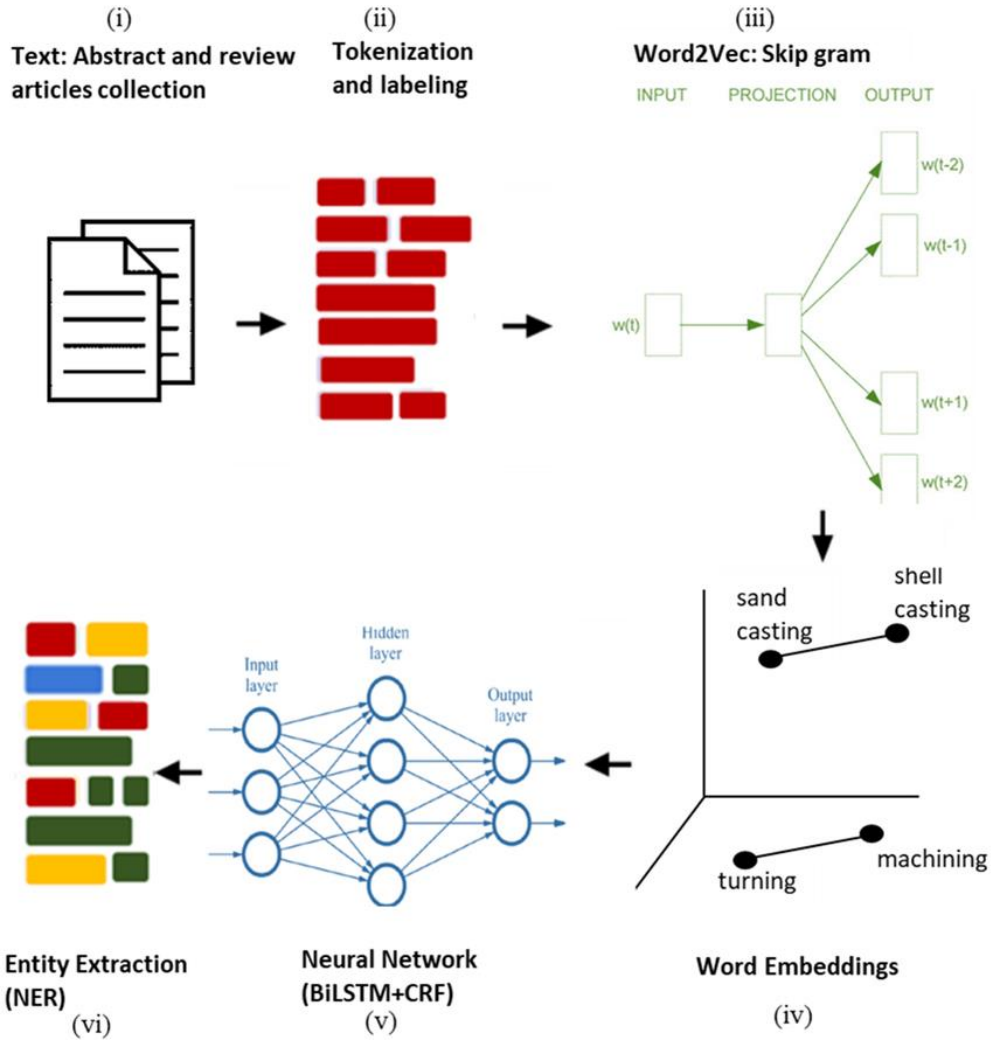
Visa identificar e extrair informações específicas de um texto, como entidades nomeadas (nomes de pessoas, locais, organizações), datas, números, relações entre entidades, entre outros. É útil para obter informações estruturadas a partir de textos não estruturados. *(supervisionado)*



Text analytics – algumas abordagens

Entity recognition

Visa identificar e extrair informações específicas de um texto, como entidades nomeadas (nomes de pessoas, locais, organizações), datas, números, relações entre entidades, entre outros. É útil para obter informações estruturadas a partir de textos não estruturados. *(supervisionado)*

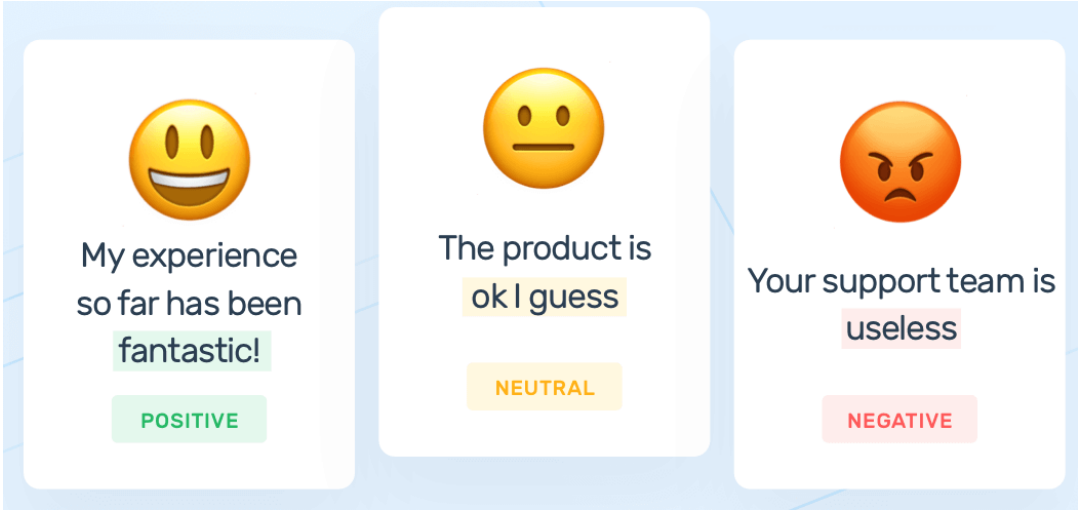


Text analytics – algumas abordagens

Sentiment analysis

Visa determinar a polaridade emocional de um texto, classificando-o como positivo, negativo ou neutro. Pode ser aplicada em avaliações de produtos, feedback de clientes, mídias sociais e outras fontes para medir a opinião geral ou o sentimento associado a uma determinada entidade ou tópico.

(pode ser supervisionado ou não supervisionado)



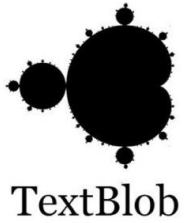
Sentiment analysis - Python

Sentiment analysis - Python

TextBlob é uma biblioteca Python para processamento de dados textuais.

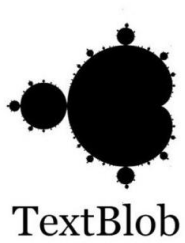
Ele não apenas permite a análise de sentimentos, mas também fornece uma API simples para executarmos tarefas comuns de processamento de linguagem natural.

Sentiment analysis - Python



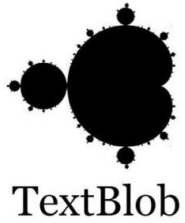
- Viabiliza a execução de ABSA não supervisionada.
- Funciona perfeitamente com dados de mídias sociais.
- Rápido suficiente para processar streaming data.
- Retorna polaridade e subjetividade.

Sentiment analysis - Python



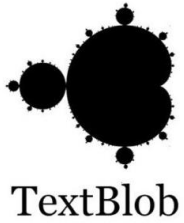
- Por ser uma abordagem baseada em léxico, um sentimento é definido por sua orientação semântica e a intensidade de cada palavra na frase.
- Isso requer um dicionário predefinido classificando palavras negativas e positivas.
- Geralmente, uma mensagem de texto será representada por uma bag-of-words.
- Depois de atribuir pontuações individuais a todas as palavras, o sentimento final é calculado pela média de todos os sentimentos identificados.

Sentiment analysis - Python



- TextBlob retorna a polaridade e a subjetividade de uma frase.
- A polaridade está entre [-1,1]
- -1 define um sentimento negativo e 1 define um sentimento positivo.
- As palavras de negação invertem a polaridade.

Sentiment analysis - Python



- A subjetividade situa-se entre [0,1].
- Quantifica a quantidade de opinião pessoal e informações factuais contidas no texto.
- Quanto maior o valor da subjetividade, mais o texto contém opinião pessoal em vez de informações factuais.

ATIVIDADE

1. Gere frases no chatGPT relacionadas ao contexto que desejarem, com um prompt semelhante a esse:

"please, provide me a list of 200 sentences considering the following:

- each sentence should have a maximum of 280 characters

- the sentences should reflect the opinion of a customer regarding an aspect of the service of an airline (such as waiting time, cost, delay, service, food, confort, etc)

- the sentences should be diverse, meaning some of positive opinions, others of negative opinions"

2. Implementem e rodem o código de Sentiment Analysis não-supervisionada com Textblob.

3. Analisem os resultados (visualização e estatística descritiva)

Sentiment analysis Zanon.ipynb

Arquivo Editar Ver Inserir Ambiente de execução Ferramentas Ajuda Última edição em 29 de out. de 2022

+ Código + Texto

```
[ ] # Importing TextBlob
    from textblob import TextBlob
    import pandas as pd
```

```
sentences = [
  'The food we had yesterday was incredible',
  'My time in Italy was very enjoyable',
  'I found the meal very tasty',
  'The internet was slow.',
  'The Sound Quality is great.',
  'The hotel structure is great.',
  'Our experience was terrible',
  'The battery has not a good performance.',
  'The camera and resolution are perfect.'
  'the most impressive features of the apple iphone 13 is the extended battery life',
  'no question, quality control is a disaster',
  'launched the new with a few new changes and a muchimproved battery life',
  'the camera is awesome',
  'this camera is insane',
  'definitely got the beat for me this camera!',
  'i love this iphone13, the camera is so bomb',
  'the quality of this camera is breathtaking',
  'the camera is pretty ok',
  'iphone 14 pro the best camera in an iphone and than ever',
  'this camera is amazing i got right up to the window he looks huge'
]

print(sentences)
```

```
['The food we had yesterday was incredible', 'My time in Italy was very enjoyable', 'I found the meal very tasty', 'The internet was slow.',
```

Sentiment analysis Zanon.ipynb ☆

Arquivo Editar Ver Inserir Ambiente de execução Ferramentas Ajuda Última edição em 29 de out. de 2022

+ Código + Texto

```
[ ] # Importing TextBlob
    from textblob import TextBlob
    import pandas as pd
```

```
sentences = [
  'The food we had yesterday was incredible',
  'My time in Italy was very enjoyable',
  'I found the meal very tasty',
  'The internet was slow.',
  'The Sound Quality is great.',
  'The hotel structure is great.',
  'Our experience was terrible',
  'The battery has not a good performance.',
  'The camera and resolution are perfect.'
  'the most impressive features of the apple iphone 13 is the extended battery life',
  'no question, quality control is a disaster',
  'launched the new with a few new changes and a muchimproved battery life',
  'the camera is awesome',
  'this camera is insane',
  'definitely got the beat for me this camera!',
  'i love this iphone13, the camera is so bomb',
  'the quality of this camera is breathtaking',
  'the camera is pretty ok',
  'iphone 14 pro the best camera in an iphone and than ever',
  'this camera is amazing i got right up to the window he looks huge'
]

print(sentences)
```

```
['The food we had yesterday was incredible', 'My time in Italy was very enjoyable', 'I found the meal very tasty', 'The internet was slow.',
```

Carregar sentenças. Aqui estamos fazendo dentro do código, numa lista. Cada frase é representada como uma string delimitada por aspas simples ou duplas. A operação pode ser substituída pela leitura de um arquivo .csv, por exemplo.

Sentiment analysis Zanon.ipynb

Arquivo Editar Ver Inserir Ambiente de execução Ferramentas Ajuda Última edição em 29 de out. de 2022

+ Código + Texto

```
[ ] def getSubjectivity(text):
      return TextBlob(text).sentiment.subjectivity


    def getPolarity(text):
      return TextBlob(text).sentiment.polarity

    listaSub = []
    listaPolaridade = []
    for i in sentences:
      try:
        listaSub.append(getSubjectivity(i))
        listaPolaridade.append(getPolarity(i))
      except:
        listaSub.append(0.0)
        listaPolaridade.append(0.0)


    dfFinal = pd.DataFrame(listaPolaridade, columns = ['Polaridade'])
    dfFinal['Subjetividade'] = listaSub
    dfFinal["Sentença"] = sentences
    dfFinal
```

	Polaridade	Subjetividade	Sentença
0	0.900000	0.900000	The food we had yesterday was incredible
1	0.650000	0.780000	My time in Italy was very enjoyable
2	0.200000	0.300000	I found the meal very tasty
3	-0.300000	0.400000	The internet was slow.
4	0.600000	0.575000	The Sound Quality is great.
5	0.800000	0.750000	The hotel structure is great.
6	-1.000000	1.000000	Our experience was terrible
7	-0.350000	0.600000	The battery has not a good performance.
8	0.750000	0.750000	The camera and resolution are perfect.the most...
9	0.000000	0.000000	no question, quality control is a disaster
10	0.024242	0.336364	launched the new with a few new changes and a ...

Sentiment analysis Zanon.ipynb

Arquivo Editar Ver Inserir Ambiente de execução Ferramentas Ajuda Última edição em 29 de out. de 2022

+ Código + Texto

```
[ ]  def getSubjectivity(text):
       return TextBlob(text).sentiment.subjectivity


     def getPolarity(text):
       return TextBlob(text).sentiment.polarity

     listaSub = []
     listaPolaridade = []
     for i in sentences:
       try:
         listaSub.append(getSubjectivity(i))
         listaPolaridade.append(getPolarity(i))
       except:
         listaSub.append(0.0)
         listaPolaridade.append(0.0)


     dfFinal = pd.DataFrame(listaPolaridade, columns = ['Polaridade'])
     dfFinal['Subjetividade'] = listaSub
     dfFinal["Sentença"] = sentences
     dfFinal
```

	Polaridade	Subjetividade	Sentença
0	0.900000	0.900000	The food we had yesterday was incredible
1	0.650000	0.780000	My time in Italy was very enjoyable
2	0.200000	0.300000	I found the meal very tasty
3	-0.300000	0.400000	The internet was slow.
4	0.600000	0.575000	The Sound Quality is great.
5	0.800000	0.750000	The hotel structure is great.
6	-1.000000	1.000000	Our experience was terrible
7	-0.350000	0.600000	The battery has not a good performance.
8	0.750000	0.750000	The camera and resolution are perfect.the most...
9	0.000000	0.000000	no question, quality control is a disaster
10	0.024242	0.336364	launched the new with a few new changes and a ...

A função **getSubjectivity** recebe um texto como entrada, cria um objeto **TextBlob** a partir desse texto e, em seguida, retorna a subjetividade desse objeto. A subjetividade é um valor entre 0 e 1, onde 0 indica objetividade (fato) e 1 indica subjetividade (opinião).

A função **getPolarity** funciona de maneira semelhante, recebendo um texto, criando um objeto **TextBlob** a partir dele e retornando a polaridade desse objeto. A polaridade é um valor entre -1 e 1, onde -1 indica uma opinião negativa, 0 indica neutralidade e 1 indica uma opinião positiva.

No loop **for**, cada frase da lista **sentences** é processada pelas funções **getSubjectivity** e **getPolarity**. Se uma exceção ocorrer durante o processamento, o que pode acontecer se uma frase não puder ser analisada corretamente, um valor padrão de 0.0 é adicionado às listas **listaSub** e **listaPolaridade**.

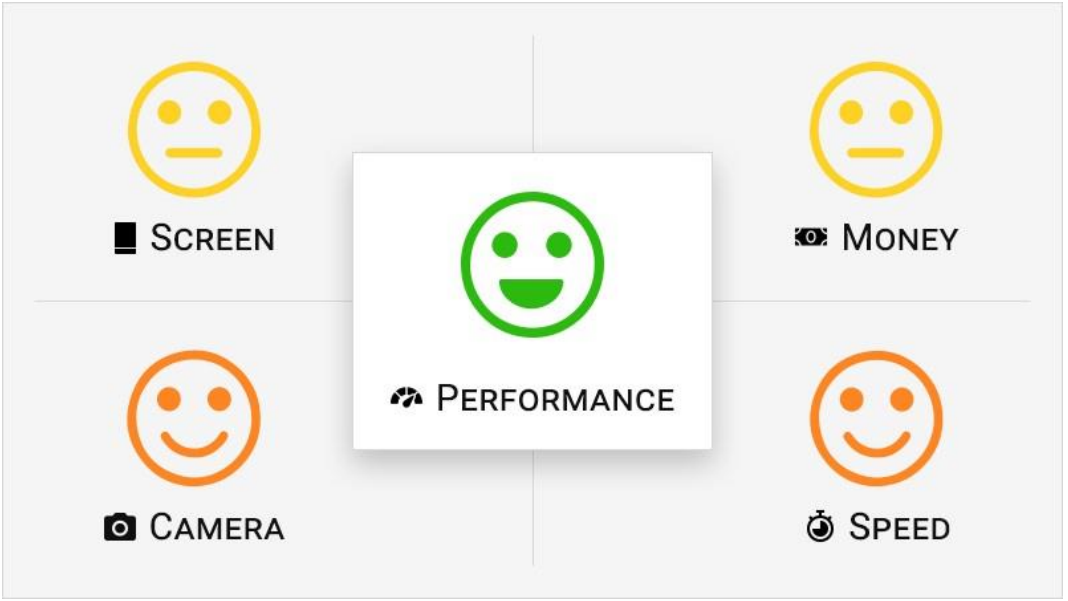
Em seguida, os dados são organizados em um DataFrame do pandas chamado **dfFinal**. As listas **listaPolaridade**, **listaSub** e **sentences** são atribuídas como colunas do DataFrame.

Aspect Based Sentiment Analysis

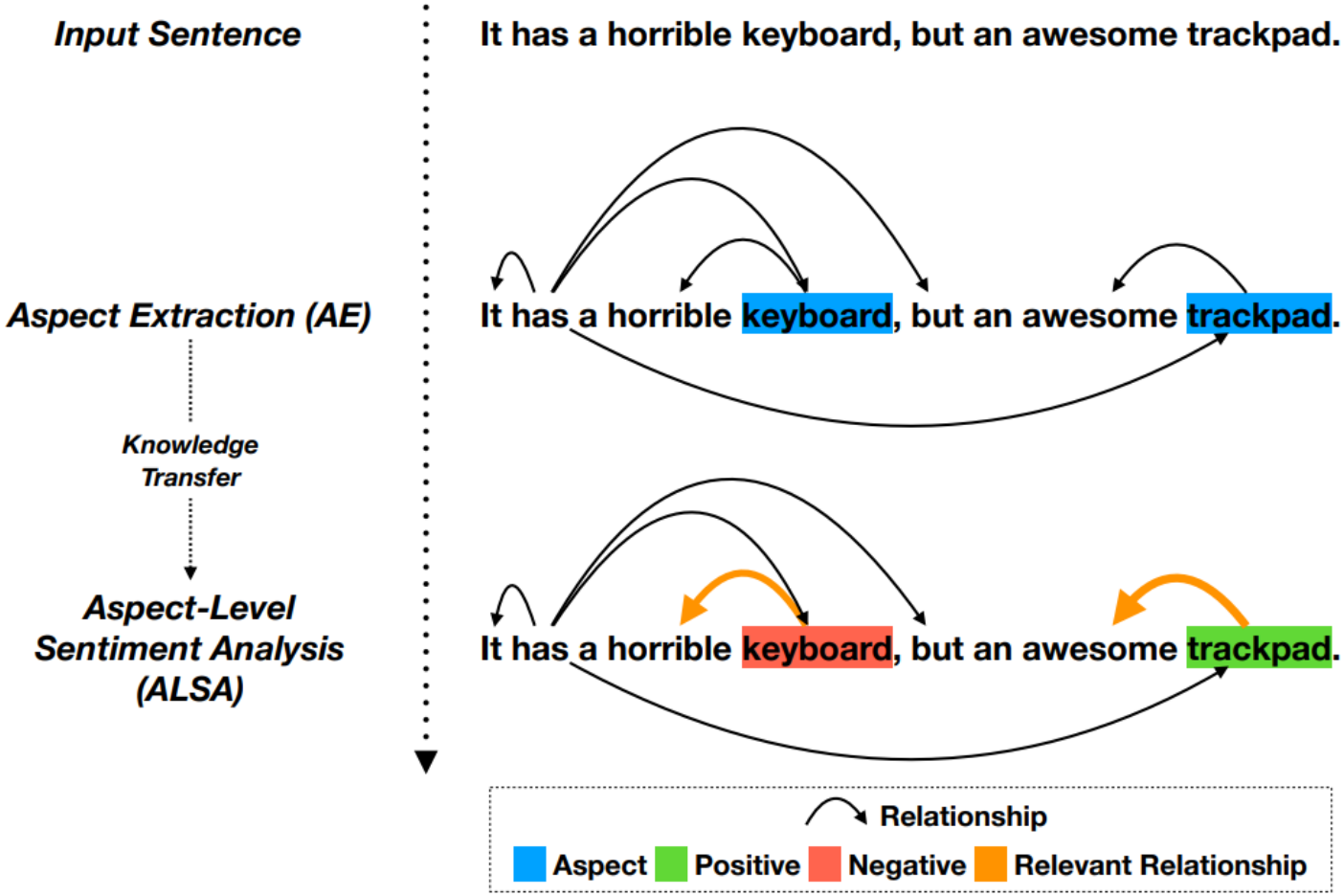
Aspect Based Sentiment analysis

- Identificar sentimentos associados a aspectos específicos em um texto é uma tarefa mais complexa conhecida como Aspect Based Sentiment Analysis (ABSA).

- A principal vantagem do ABSA é que não só é possível monitorar o conteúdo das frases em larga escala, mas sobre qual aspecto aquela frase trata e qual sentimento ela expressa em relação a esse aspecto especificamente.



Aspect Based Sentiment analysis



Aspect Based Sentiment analysis

1) Pré-processamento de texto:

- Converter todas as letras para minúsculas.
- Remover caracteres especiais e números, mantendo apenas letras.
- Tokenizar o texto em palavras.
- Remover palavras irrelevantes, como stopwords (palavras comuns que não carregam muito significado, como "is", "the", etc.).
- Lemmatizar as palavras (reduzir as palavras para sua forma base, por exemplo, "running" para "run").

Aspect Based Sentiment analysis

2) Extração de aspectos:

- Receber um texto e extrair os aspectos (substantivos) e adjetivos mencionados no texto.

- Para isso, o texto é dividido em sentenças, e em cada sentença, as palavras são marcadas com as classes gramaticais (part-of-speech tags).

- As palavras marcadas como substantivos (que começam com 'NN') são consideradas aspectos, enquanto as palavras marcadas como adjetivos (que começam com 'JJ') são consideradas adjetivos.

Aspect Based Sentiment analysis

3) Análise de sentimentos:

- São identificadas todas as associações substantivo-adjetivo.

- Em seguida, é utilizada a biblioteca TextBlob para realizar a análise de sentimento em cada uma dessas associações.

- A polaridade do sentimento (valor entre -1 e 1) é extraída e armazenada para o aspecto, bem como sua polaridade (valor entre 0 e 1).

Aspect Based Sentiment analysis

4) Visualização e interpretação:

- Frequência dos aspectos.
- Seleção dos principais aspectos.
- Cálculo dos valores médios de polaridade e subjetividade.
- Estratificação dos adjetivos relacionados a cada aspecto principal.
- Plotagem de gráficos.
- Etc.

Aspect Based Sentiment analysis

1. Gere frases no chatGPT relacionadas ao contexto que desejarem, com um prompt semelhante a esse:

"please, provide me a list of 200 sentences considering the following:

- each sentence should have a maximum of 280 characters

- the sentences should reflect the opinion of a customer regarding an aspect of the service of an airline (such as waiting time, cost, delay, service, food, confort, etc)

- the sentences should contain positive and negative opinions regarding different aspects of the airline service (such as "good service, but bad at flight confort").

2. Implementem e rodem o código de ABSA não-supervisionada com Textblob.

3. Analisem os resultados (visualização e estatística descritiva)

ABSA_textblob_SEP5836.ipynb

Arquivo Editar Ver Inserir Ambiente de execução Ferramentas Ajuda

+ Código + Texto

```
import re
import nltk
from nltk.corpus import stopwords
from textblob import TextBlob
import pandas as pd
import matplotlib.pyplot as plt

nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('averaged_perceptron_tagger')
```

A primeira linha importa o módulo **re**, que é usado manipular strings de texto.

A segunda linha importa o pacote **nltk** (Natural Language Toolkit), que é uma biblioteca amplamente usada para processamento de linguagem natural em Python.

A terceira linha importa o módulo **stopwords** do pacote nltk. As stopwords são palavras comuns (por exemplo, "o", "é", "e", "para", etc.) que geralmente não contêm informações úteis para a análise de texto e são frequentemente removidas para reduzir o ruído nos dados de texto.

A terceira linha importa o módulo **stopwords** do pacote nltk. As stopwords são palavras comuns (por exemplo, "o", "é", "e", "para", etc.) que geralmente não contêm informações úteis para a análise de texto e são frequentemente removidas para reduzir o ruído nos dados de texto.

A quarta linha importa a classe **TextBlob** do módulo **textblob**

A quinta linha importa o pacote **pandas**, que é uma biblioteca popular para análise e manipulação de dados em Python.

ABSA_textblob_SEP5836.ipynb

Arquivo Editar Ver Inserir Ambiente de execução Ferramentas Ajuda

+ Código + Texto

```
import re
import nltk
from nltk.corpus import stopwords
from textblob import TextBlob
import pandas as pd
import matplotlib.pyplot as plt

nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('averaged_perceptron_tagger')
```

A sexta linha importa o módulo **pyplot** do pacote **matplotlib**, que é uma biblioteca de visualização de dados em Python. O **pyplot** fornece funções para criar gráficos e visualizações a partir de dados.

As últimas quatro linhas realizam o download de recursos adicionais do pacote **nltk** que são necessários para algumas funcionalidades específicas.

Esses recursos incluem modelos de tokenização (**punkt**), stopwords (**stopwords**), um lematizador (**wordnet**) e um etiquetador POS (**averaged_perceptron_tagger**).

Esses recursos são baixados na primeira execução do código para garantir que estejam disponíveis localmente para uso posterior.

ABSA_textblob_SEP5836.ipynb

Arquivo Editar Ver Inserir Ambiente de execução Ferramentas Ajuda

+ Código + Texto

```
import re
import nltk
from nltk.corpus import stopwords
from textblob import TextBlob
import pandas as pd
import matplotlib.pyplot as plt

nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('averaged_perceptron_tagger')


def preprocess_text(text):
    # Convert to lowercase
    text = text.lower()
    # Remove special characters and numbers
    text = re.sub(r'[^a-zA-Z]', ' ', text)
    # Tokenize the text
    tokens = nltk.word_tokenize(text)
    # Remove stopwords
    stop_words = set(stopwords.words('english'))
    tokens = [word for word in tokens if word not in stop_words]
    # Lemmatize the tokens
    lemmatizer = nltk.WordNetLemmatizer()
    tokens = [lemmatizer.lemmatize(word) for word in tokens]
    # Join tokens into a single string
    preprocessed_text = ' '.join(tokens)
    return preprocessed_text
```

Definimos uma função chamada **preprocess_text** que recebe um parâmetro **text**, que representa o texto a ser pré-processado.

text = text.lower()
Essa linha converte todo o texto para letras minúsculas. Isso é feito para garantir consistência na análise de texto, considerando que letras maiúsculas e minúsculas são tratadas de forma diferente.

text = re.sub(r'[^a-zA-Z]', ' ', text)
Essa linha usa expressões regulares para remover caracteres especiais e números do texto. A função **re.sub()** substitui qualquer caractere que não seja uma letra (maiúscula ou minúscula) por um espaço em branco.

tokens = nltk.word_tokenize(text)
Essa linha utiliza a função **word_tokenize()** do pacote **nltk** para tokenizar o texto em palavras individuais. Isso divide o texto em uma lista de tokens (palavras).

ABSA_textblob_SEP5836.ipynb

Arquivo Editar Ver Inserir Ambiente de execução Ferramentas Ajuda

+ Código + Texto

```
import re
import nltk
from nltk.corpus import stopwords
from textblob import TextBlob
import pandas as pd
import matplotlib.pyplot as plt

nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('averaged_perceptron_tagger')


def preprocess_text(text):
    # Convert to lowercase
    text = text.lower()
    # Remove special characters and numbers
    text = re.sub(r'[^a-zA-Z]', ' ', text)
    # Tokenize the text
    tokens = nltk.word_tokenize(text)
    # Remove stopwords
    stop_words = set(stopwords.words('english'))
    tokens = [word for word in tokens if word not in stop_words]
    # Lemmatize the tokens
    lemmatizer = nltk.WordNetLemmatizer()
    tokens = [lemmatizer.lemmatize(word) for word in tokens]
    # Join tokens into a single string
    preprocessed_text = ' '.join(tokens)
    return preprocessed_text
```

```
stop_words = set(stopwords.words('english'))
tokens = [word for word in tokens if word not in stop_words]
```

Essas linhas definem uma lista de stopwords para o idioma inglês usando o pacote **nltk**. Em seguida, é criada uma nova lista de tokens onde as stopwords são removidas. Isso é feito usando uma compreensão de lista, onde cada palavra da lista original (**tokens**) é verificada para ver se não está presente nas stopwords antes de ser incluída na nova lista.

```
lemmatizer = nltk.WordNetLemmatizer()
tokens = [lemmatizer.lemmatize(word) for word in tokens]
```

Essas linhas utilizam o lematizador do **nltk** para lematizar os tokens. O lematizador reduz cada palavra aos seus lemas, ou seja, a forma básica da palavra. Por exemplo, os verbos no infinitivo, os substantivos no singular, etc. O lematizador usado é o **WordNetLemmatizer** do pacote **nltk**.

```
preprocessed_text = ' '.join(tokens)
```

Essa linha junta os tokens pré-processados de volta em uma única string, separados por espaços. Isso é feito usando o método **join()** do tipo de string.

```
return preprocessed_text
```

A função retorna o texto pré-processado.

ABSA_textblob_SEP5836.ipynb

Arquivo Editar Ver Inserir Ambiente de execução Ferramentas Ajuda

+ Código + Texto

```python
def extract_aspects(text):
    # Extract aspects from the text
    sentences = nltk.sent_tokenize(text)
    aspects = []
    adjectives = []
    for sentence in sentences:
        # Tokenize the sentence
        tokens = nltk.word_tokenize(sentence)
        # Tag the tokens with part-of-speech (POS) tags
        tagged_tokens = nltk.pos_tag(tokens)
        # Extract nouns and adjectives as aspects
        for word, tag in tagged_tokens:
            if tag.startswith('NN'):
                aspects.append(word.lower())
            elif tag.startswith('JJ'):
                adjectives.append(word.lower())
    return aspects, adjectives
```

A função `extract_aspects` realiza a extração de aspectos de um texto. Vamos analisar cada linha em detalhes:

python Copy code

```python
def extract_aspects(text):
```

Essa linha define uma função chamada `extract_aspects` que recebe um parâmetro `text`, que representa o texto a partir do qual os aspectos serão extraídos. A função irá retornar duas listas: uma lista de aspectos e uma lista de adjetivos.

python Copy code

```python
sentences = nltk.sent_tokenize(text)
```

Essa linha utiliza a função `sent_tokenize()` do pacote `nltk` para tokenizar o texto em frases individuais. Isso divide o texto em uma lista de frases (sentenças).

ABSA_textblob_SEP5836.ipynb

Arquivo Editar Ver Inserir Ambiente de execução Ferramentas Ajuda

+ Código + Texto

```python
def extract_aspects(text):
    # Extract aspects from the text
    sentences = nltk.sent_tokenize(text)
    aspects = []
    adjectives = []
    for sentence in sentences:
        # Tokenize the sentence
        tokens = nltk.word_tokenize(sentence)
        # Tag the tokens with part-of-speech (POS) tags
        tagged_tokens = nltk.pos_tag(tokens)
        # Extract nouns and adjectives as aspects
        for word, tag in tagged_tokens:
            if tag.startswith('NN'):
                aspects.append(word.lower())
            elif tag.startswith('JJ'):
                adjectives.append(word.lower())
    return aspects, adjectives
```

python Copy code

```python
aspects = []
adjectives = []
```

Essas linhas criam duas listas vazias, `aspects` e `adjectives`, que serão usadas para armazenar os aspectos extraídos e os adjetivos relacionados.

python Copy code

```python
for sentence in sentences:
```

Essa linha inicia um loop que irá percorrer cada sentença no texto.

python Copy code

```python
tokens = nltk.word_tokenize(sentence)
```

Essa linha utiliza a função `word_tokenize()` do pacote `nltk` para tokenizar cada sentença em palavras individuais. Isso divide a sentença em uma lista de tokens (palavras).

ABSA_textblob_SEP5836.ipynb

Arquivo Editar Ver Inserir Ambiente de execução Ferramentas Ajuda

+ Código + Texto

```python
def extract_aspects(text):
    # Extract aspects from the text
    sentences = nltk.sent_tokenize(text)
    aspects = []
    adjectives = []
    for sentence in sentences:
        # Tokenize the sentence
        tokens = nltk.word_tokenize(sentence)
        # Tag the tokens with part-of-speech (POS) tags
        tagged_tokens = nltk.pos_tag(tokens)
        # Extract nouns and adjectives as aspects
        for word, tag in tagged_tokens:
            if tag.startswith('NN'):
                aspects.append(word.lower())
            elif tag.startswith('JJ'):
                adjectives.append(word.lower())
    return aspects, adjectives
```

```
tagged_tokens = nltk.pos_tag(tokens)
```

Essa linha utiliza a função **`pos_tag()`** do pacote **`nltk`** para atribuir uma etiqueta de parte do discurso (POS) a cada token na sentença. Isso identifica a classe gramatical de cada palavra, como substantivo (NN), adjetivo (JJ), verbo (VB), etc.

python Copy code

```
for word, tag in tagged_tokens:
```

Essa linha inicia um loop que itera sobre cada par de palavra e etiqueta na sentença.

python Copy code

```
if tag.startswith('NN'):
    aspects.append(word.lower())
elif tag.startswith('JJ'):
    adjectives.append(word.lower())
```

Essas linhas verificam se a etiqueta de parte do discurso começa com 'NN' (substantivo) ou 'JJ' (adjetivo). Se a palavra for um substantivo, ela é adicionada à lista de **`aspects`** (aspectos) após ser convertida para minúsculas. Se a palavra for um adjetivo, ela é adicionada à lista de **`adjectives`** (adjetivos) após ser convertida para minúsculas.

python Copy code

```
return aspects, adjectives
```

A função retorna as duas listas de aspectos e adjetivos extraídos.

ABSA_textblob_SEP5836.ipynb

Arquivo Editar Ver Inserir Ambiente de execução Ferramentas Ajuda

+ Código + Texto

```python
def aspect_sentiment_analysis(text, aspects):
    # Preprocess the text
    preprocessed_text = preprocess_text(text)
    sentiment_scores = {}
    for aspect in aspects:
        aspect_sentiment = []
        # Identify sentences related to the aspect
        sentences = nltk.sent_tokenize(preprocessed_text)
        aspect_sentences = [sentence for sentence in sentences if aspect in sentence]
        for sentence in aspect_sentences:
            # Perform sentiment analysis on each aspect sentence
            blob = TextBlob(sentence)
            sentiment_score = blob.sentiment.polarity
            aspect_sentiment.append(sentiment_score)
        # Calculate average sentiment score for the aspect
        if aspect_sentiment:
            aspect_sentiment_score = sum(aspect_sentiment) / len(aspect_sentiment)
        else:
            aspect_sentiment_score = 0
        sentiment_scores[aspect] = aspect_sentiment_score
    return sentiment_scores
```

python Copy code

```python
def aspect_sentiment_analysis(text, aspects):
```

Essa linha define uma função chamada `aspect_sentiment_analysis` que recebe dois parâmetros: `text`, que representa o texto a ser analisado, e `aspects`, que é uma lista dos aspectos específicos que serão avaliados em termos de sentimento. A função irá retornar um dicionário de pontuações de sentimento para cada aspecto.

python Copy code

```python
preprocessed_text = preprocess_text(text)
```

Essa linha chama a função `preprocess_text` para pré-processar o texto fornecido. O texto é convertido para minúsculas, caracteres especiais e números são removidos, stopwords são removidas, os tokens são lematizados e, por fim, os tokens são unidos novamente em uma única string. O texto pré-processado é atribuído à variável `preprocessed_text`.

python Copy code

```python
sentiment_scores = {}
```

Essa linha cria um dicionário vazio chamado `sentiment_scores`, que será usado para armazenar as pontuações de sentimento para cada aspecto.

ABSA_textblob_SEP5836.ipynb

Arquivo Editar Ver Inserir Ambiente de execução Ferramentas Ajuda

+ Código + Texto

```python
def aspect_sentiment_analysis(text, aspects):
    # Preprocess the text
    preprocessed_text = preprocess_text(text)
    sentiment_scores = {}
    for aspect in aspects:
        aspect_sentiment = []
        # Identify sentences related to the aspect
        sentences = nltk.sent_tokenize(preprocessed_text)
        aspect_sentences = [sentence for sentence in sentences if aspect in sentence]
        for sentence in aspect_sentences:
            # Perform sentiment analysis on each aspect sentence
            blob = TextBlob(sentence)
            sentiment_score = blob.sentiment.polarity
            aspect_sentiment.append(sentiment_score)
        # Calculate average sentiment score for the aspect
        if aspect_sentiment:
            aspect_sentiment_score = sum(aspect_sentiment) / len(aspect_sentiment)
        else:
            aspect_sentiment_score = 0
        sentiment_scores[aspect] = aspect_sentiment_score
    return sentiment_scores
```

python Copy code

```python
for aspect in aspects:
```

Essa linha inicia um loop que itera sobre cada aspecto na lista de `aspects`.

python Copy code

```python
aspect_sentiment = []
```

Essa linha cria uma lista vazia chamada `aspect_sentiment`, que será usada para armazenar as pontuações de sentimento para as sentenças relacionadas ao aspecto atual.

python Copy code

```python
sentences = nltk.sent_tokenize(preprocessed_text)
aspect_sentences = [sentence for sentence in sentences if aspect in sentence
```

Essas linhas dividem o texto pré-processado em sentenças usando a função `sent_tokenize` do pacote `nltk`. Em seguida, é criada uma nova lista chamada `aspect_sentences` que contém apenas as sentenças que contêm o aspecto atual.

ABSA_textblob_SEP5836.ipynb

Arquivo Editar Ver Inserir Ambiente de execução Ferramentas Ajuda

+ Código + Texto

```python
def aspect_sentiment_analysis(text, aspects):
    # Preprocess the text
    preprocessed_text = preprocess_text(text)
    sentiment_scores = {}
    for aspect in aspects:
        aspect_sentiment = []
        # Identify sentences related to the aspect
        sentences = nltk.sent_tokenize(preprocessed_text)
        aspect_sentences = [sentence for sentence in sentences if aspect in sentence]
        for sentence in aspect_sentences:
            # Perform sentiment analysis on each aspect sentence
            blob = TextBlob(sentence)
            sentiment_score = blob.sentiment.polarity
            aspect_sentiment.append(sentiment_score)
        # Calculate average sentiment score for the aspect
        if aspect_sentiment:
            aspect_sentiment_score = sum(aspect_sentiment) / len(aspect_sentiment)
        else:
            aspect_sentiment_score = 0
        sentiment_scores[aspect] = aspect_sentiment_score
    return sentiment_scores
```

python Copy code

```python
for sentence in aspect_sentences:
```

Essa linha inicia um loop que itera sobre cada sentença nas `aspect_sentences`.

python Copy code

```python
blob = TextBlob(sentence)
sentiment_score = blob.sentiment.polarity
aspect_sentiment.append(sentiment_score)
```

Essas linhas usam a classe `TextBlob` para realizar a análise de sentimento em cada sentença relacionada ao aspecto atual. A sentença é passada para o `TextBlob`, que atribui uma pontuação de polaridade ao sentimento da sentença. A pontuação de sentimento é obtida usando a propriedade `polarity` do `TextBlob`. A pontuação é adicionada à lista `aspect_sentiment`.

ABSA_textblob_SEP5836.ipynb

Arquivo Editar Ver Inserir Ambiente de execução Ferramentas Ajuda

+ Código + Texto

```python
def aspect_sentiment_analysis(text, aspects):
    # Preprocess the text
    preprocessed_text = preprocess_text(text)
    sentiment_scores = {}
    for aspect in aspects:
        aspect_sentiment = []
        # Identify sentences related to the aspect
        sentences = nltk.sent_tokenize(preprocessed_text)
        aspect_sentences = [sentence for sentence in sentences if aspect in sentence]
        for sentence in aspect_sentences:
            # Perform sentiment analysis on each aspect sentence
            blob = TextBlob(sentence)
            sentiment_score = blob.sentiment.polarity
            aspect_sentiment.append(sentiment_score)
        # Calculate average sentiment score for the aspect
        if aspect_sentiment:
            aspect_sentiment_score = sum(aspect_sentiment) / len(aspect_sentiment)
        else:
            aspect_sentiment_score = 0
        sentiment_scores[aspect] = aspect_sentiment_score
    return sentiment_scores
```

python Copy code

```python
if aspect_sentiment:
    aspect_sentiment_score = sum(aspect_sentiment) / len(aspect_sentiment)
else:
    aspect_sentiment_score = 0
sentiment_scores[aspect] = aspect_sentiment_score
```

Essas linhas calculam a pontuação média de sentimento para o aspecto atual. Se houver pontuações de sentimento na lista `aspect_sentiment`, a pontuação média é calculada dividindo a soma das pontuações pelo número de pontuações. Caso contrário, se a lista estiver vazia, a pontuação de sentimento é definida como zero. A pontuação de sentimento é atribuída ao aspecto atual no dicionário `sentiment_scores`.

python Copy code

```python
return sentiment_scores
```

A função retorna o dicionário `sentiment_scores`, que contém as pontuações de sentimento para cada aspecto.

ABSA_textblob_SEP5836.ipynb

Arquivo Editar Ver Inserir Ambiente de execução Ferramentas Ajuda

+ Código + Texto

```
def getSubjectivity(text):
    return TextBlob(text).sentiment.subjectivity


def getPolarity(text):
    return TextBlob(text).sentiment.polarity


sentences = [
    'The service on this flight was outstanding',
'The legroom in economy class is cramped',
'The food quality was excellent, but the portion size was too small',
'The in-flight entertainment system was outdated and had limited options',
'The flight attendants were polite and attentive',
'The check-in process was slow and inefficient',
'The Wi-Fi on board was unreliable and kept disconnecting',
"The airline's loyalty program offers great rewards and benefits",
```

Essas funções e a lista de sentenças podem ser usadas para obter a subjetividade e polaridade de um texto em particular ou de uma lista de sentenças. Por exemplo, pode-se chamar a função **getSubjectivity()** passando uma frase como argumento para obter a subjetividade desse texto. Da mesma forma, pode-se chamar a função **getPolarity()** para obter a polaridade. Isso pode ser útil em tarefas de análise de sentimentos, onde se deseja medir o grau de subjetividade e o sentimento expresso em um texto.

ABSA_textblob_SEP5836.ipynb

Arquivo Editar Ver Inserir Ambiente de execução Ferramentas Ajuda

+ Código + Texto

```
# Extract aspects and adjectives mentioned in the sentences
aspects = set()
adjectives = set()
for sentence in sentences:
    extracted_aspects, extracted_adjectives = extract_aspects(sentence)
    aspects.update(extracted_aspects)
    adjectives.update(extracted_adjectives)

# Dictionary to store sentiment scores for each aspect
aspect_sentiment_scores = {aspect: [] for aspect in aspects}

for sentence in sentences:
    sentiment_scores = aspect_sentiment_analysis(sentence, aspects)
    for aspect, score in sentiment_scores.items():
        aspect_sentiment_scores[aspect].append(score)
```

- `aspects = set()`: Cria um conjunto vazio chamado `aspects` para armazenar os aspectos mencionados nas sentenças.
- `adjectives = set()`: Cria um conjunto vazio chamado `adjectives` para armazenar os adjetivos mencionados nas sentenças.
- `for sentence in sentences:`: Itera sobre cada sentença na lista de sentenças.
- `extracted_aspects, extracted_adjectives = extract_aspects(sentence)`: Chama a função `extract_aspects` passando a sentença atual como argumento. Essa função retorna uma tupla contendo os aspectos e adjetivos extraídos da sentença. Esses valores são atribuídos às variáveis `extracted_aspects` e `extracted_adjectives`, respectivamente.
- `aspects.update(extracted_aspects)`: Adiciona os aspectos extraídos à variável `aspects` usando o método `update()` do conjunto. Isso garante que os aspectos sejam adicionados ao conjunto, eliminando duplicatas.
- `adjectives.update(extracted_adjectives)`: Adiciona os adjetivos extraídos à variável `adjectives` usando o método `update()` do conjunto. Isso garante que os adjetivos sejam adicionados ao conjunto, eliminando duplicatas.

ABSA_textblob_SEP5836.ipynb

Arquivo Editar Ver Inserir Ambiente de execução Ferramentas Ajuda

+ Código + Texto

```
# Dictionary to store sentiment scores for each aspect
aspect_sentiment_scores = {aspect: [] for aspect in aspects}

for sentence in sentences:
    sentiment_scores = aspect_sentiment_analysis(sentence, aspects)
    for aspect, score in sentiment_scores.items():
        aspect_sentiment_scores[aspect].append(score)

# Calculate the frequency of each aspect
aspect_frequency = {aspect: len(scores) for aspect, scores in aspect_sentiment_scores.items()}

# Sort the aspects by frequency in descending order
sorted_aspects = sorted(aspect_frequency, key=aspect_frequency.get, reverse=True)

# Select the top n most frequent aspects to display in the graph (e.g., top 10)
n = 10
selected_aspects = sorted_aspects[:n]
```

- `aspect_sentiment_scores = {aspect: [] for aspect in aspects}`: Cria um dicionário chamado `aspect_sentiment_scores` que irá armazenar as pontuações de sentimento para cada aspecto. Cada aspecto é inicializado com uma lista vazia.
- `for sentence in sentences:`: Itera sobre cada sentença na lista de sentenças.
- `sentiment_scores = aspect_sentiment_analysis(sentence, aspects)`: Chama a função `aspect_sentiment_analysis` passando a sentença atual e os aspectos como argumentos. Essa função retorna um dicionário de pontuações de sentimento para cada aspecto na sentença.
- `for aspect, score in sentiment_scores.items():`: Itera sobre cada par aspecto-pontuação no dicionário de pontuações de sentimento.
- `aspect_sentiment_scores[aspect].append(score)`: Adiciona a pontuação de sentimento à lista correspondente ao aspecto no dicionário `aspect_sentiment_scores`.
- `aspect_frequency = {aspect: len(scores) for aspect, scores in aspect_sentiment_scores.items()}`: Cria um dicionário chamado `aspect_frequency` que armazena a frequência de cada aspecto, ou seja, o número de vezes que o aspecto foi mencionado nas sentenças.
- `sorted_aspects = sorted(aspect_frequency, key=aspect_frequency.get, reverse=True)`: Classifica os aspectos com base em sua frequência em ordem descendente. Os aspectos mais frequentes serão listados primeiro.
- `n = 10`: Define o número de aspectos mais frequentes que serão selecionados para exibição no gráfico.
- `selected_aspects = sorted_aspects[:n]`: Seleciona os `n` aspectos mais frequentes da lista classificada.

ABSA_textblob_SEP5836.ipynb

Arquivo Editar Ver Inserir Ambiente de execução Ferramentas Ajuda

+ Código + Texto

```
# Calculate average sentiment score for selected aspects
aspect_avg_sentiment_scores = {
    aspect: sum(scores) / len(scores) if scores else 0
    for aspect, scores in aspect_sentiment_scores.items()
    if aspect in selected_aspects
}

# Create a DataFrame for the aspects, adjectives, polarity, and subjectivity
data = []
for aspect in selected_aspects:
    for adjective in adjectives:
        polarity = getPolarity(adjective)
        subjectivity = getSubjectivity(adjective)
        data.append([aspect, adjective, polarity, subjectivity])

df = pd.DataFrame(data, columns=['Aspect', 'Adjetivo', 'Polaridade', 'Subjetividade'])

# Print the DataFrame
print(df)
```

- **`aspect_avg_sentiment_scores = { ... }`**: Inicia a criação do dicionário **`aspect_avg_sentiment_scores`**.
- **`aspect: sum(scores) / len(scores) if scores else 0`**: Define o valor para cada chave **`aspect`** no dicionário. Ele calcula a média das pontuações (**`sum(scores) / len(scores)`**) para o aspecto atual. Se não houver pontuações para o aspecto (lista vazia), o valor padrão será 0.
- **`for aspect, scores in aspect_sentiment_scores.items()`**: Itera sobre os itens do dicionário **`aspect_sentiment_scores`**, onde cada item consiste em um aspecto e uma lista de pontuações associadas a esse aspecto.
- **`if aspect in selected_aspects`**: Verifica se o aspecto atual está presente na lista **`selected_aspects`**. Apenas os aspectos presentes nessa lista serão incluídos no dicionário final.

Portanto, o resultado dessa expressão de dicionário é um novo dicionário, **`aspect_avg_sentiment_scores`**, que contém as pontuações médias de sentimento para os aspectos selecionados.

ABSA_textblob_SEP5836.ipynb

Arquivo Editar Ver Inserir Ambiente de execução Ferramentas Ajuda

+ Código + Texto

```
# Calculate average sentiment score for selected aspects
aspect_avg_sentiment_scores = {
    aspect: sum(scores) / len(scores) if scores else 0
    for aspect, scores in aspect_sentiment_scores.items()
    if aspect in selected_aspects
}

# Create a DataFrame for the aspects, adjectives, polarity, and subjectivity
data = []
for aspect in selected_aspects:
    for adjective in adjectives:
        polarity = getPolarity(adjective)
        subjectivity = getSubjectivity(adjective)
        data.append([aspect, adjective, polarity, subjectivity])

df = pd.DataFrame(data, columns=['Aspect', 'Adjetivo', 'Polaridade', 'Subjetividade'])

# Print the DataFrame
print(df)
```

- `data = []`: Cria uma lista vazia chamada `data` que será usada para armazenar as informações sobre aspectos, adjetivos, polaridade e subjetividade.
- `for aspect in selected_aspects:`: Itera sobre cada aspecto na lista `selected_aspects`.
- `for adjective in adjectives:`: Itera sobre cada adjetivo na lista `adjectives`.
- `polarity = getPolarity(adjective)`: Chama a função `getPolarity` para obter a polaridade do adjetivo atual.
- `subjectivity = getSubjectivity(adjective)`: Chama a função `getSubjectivity` para obter a subjetividade do adjetivo atual.
- `data.append([aspect, adjective, polarity, subjectivity])`: Adiciona uma lista contendo o aspecto, adjetivo, polaridade e subjetividade à lista `data`.
- `df = pd.DataFrame(data, columns=['Aspect', 'Adjetivo', 'Polaridade', 'Subjetividade'])`: Cria um DataFrame chamado `df` usando a lista `data` e especificando os nomes das colunas como 'Aspect', 'Adjetivo', 'Polaridade' e 'Subjetividade'.
- `print(df)`: Imprime o DataFrame.

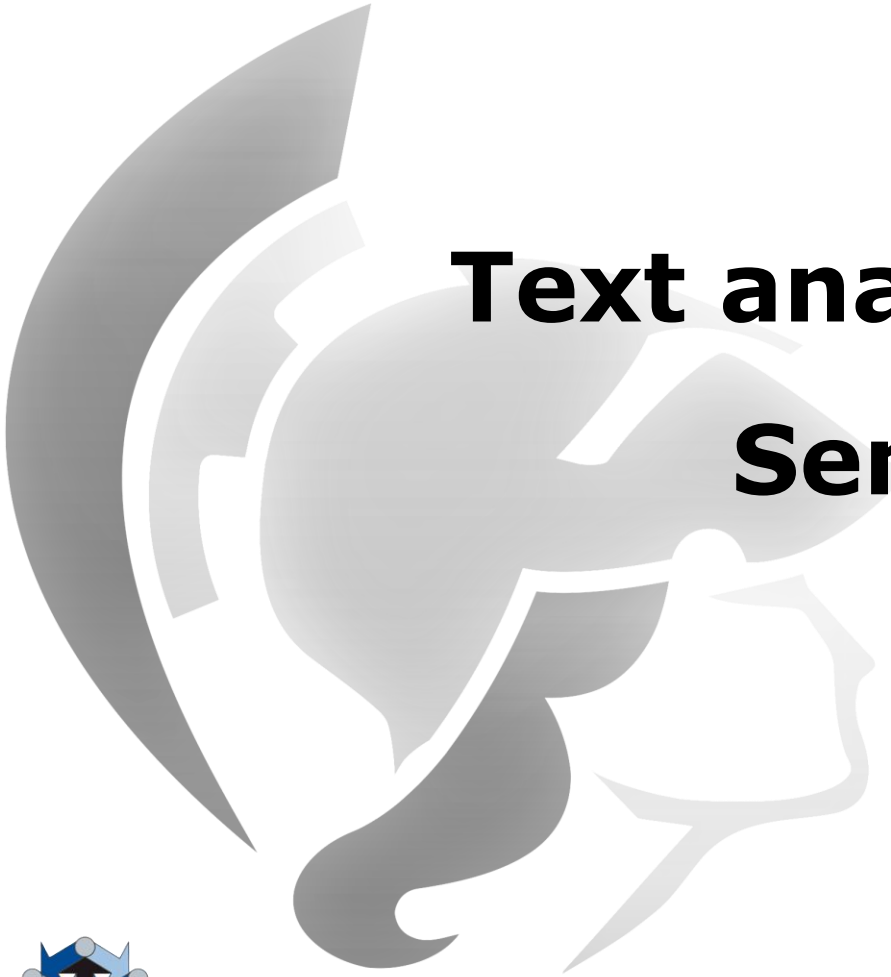
ABSA_textblob_SEP5836.ipynb

Arquivo Editar Ver Inserir Ambiente de execução Ferramentas Ajuda

+ Código + Texto

```
# Plotting the sentiment scores
plt.figure(figsize=(10, 6))
plt.bar(range(len(aspect_avg_sentiment_scores)), aspect_avg_sentiment_scores.values())
plt.xlabel('Aspect')
plt.ylabel('Average Sentiment Score')
plt.title('Aspect-Based Sentiment Analysis')
plt.xticks(range(len(aspect_avg_sentiment_scores)), aspect_avg_sentiment_scores.keys(), rotation=45)
plt.show()
```

- `plt.figure(figsize=(10, 6))`: Cria uma nova figura do matplotlib com o tamanho especificado (largura=10, altura=6). Essa linha define as dimensões da figura que irá conter o gráfico de barras.
- `plt.bar(range(len(aspect_avg_sentiment_scores)), aspect_avg_sentiment_scores.values())`: Cria um gráfico de barras utilizando os valores do dicionário `aspect_avg_sentiment_scores`. A função `plt.bar()` recebe como argumento a sequência de posições das barras no eixo x e a sequência de alturas das barras no eixo y.
- `plt.xlabel('Aspect')`: Define um rótulo para o eixo x do gráfico, que representa os aspectos.
- `plt.ylabel('Average Sentiment Score')`: Define um rótulo para o eixo y do gráfico, que representa a pontuação média de sentimento.
- `plt.title('Aspect-Based Sentiment Analysis')`: Define um título para o gráfico.
- `plt.xticks(range(len(aspect_avg_sentiment_scores)), aspect_avg_sentiment_scores.keys(), rotation=45)`: Define as marcações no eixo x do gráfico. Os rótulos são extraídos das chaves do dicionário `aspect_avg_sentiment_scores`, e a função `range(len(aspect_avg_sentiment_scores))` determina as posições das marcações no eixo x. O argumento `rotation=45` rotaciona os rótulos em 45 graus para facilitar a leitura.
- `plt.show()`: Exibe o gráfico.



Text analytics e Aspect Based Sentiment Analysis