
SHIFT
FIAP

Ana Raquel

Carreira

- Tecnólogo em banco de dados pela faculdade FIAP.
- MBA em inteligência artificial pela FIAP.
- Mais de 8 anos de experiência como profissional na área de dados tendo atuado em diversos projetos de Banco de Dados, BI, Analytics e Data Science.
- Cientista de dados na FIAP e professora de Machine Learning , Deep Learning, Processamento de Linguagem Natural e Data Viz na FIAP.

Manipulação de texto

N-grama

Basicamente, podemos considerar que o “N-grama” **separa a quantidade de elementos que compõe um texto**. O “N” representa muitos elementos e esses elementos podem ser definidos como:

- Unigramas (N = 1) podem ser chamados de Tokens.
- Bigramas (N = 2)
- Trigramas (N = 3).

Vamos considerar a seguinte frase e ver o comportamento do N-grama para cada uma delas:

Eu amo café com leite.
Eu amo chocolate ao leite com café.

Vamos verificar o funcionamento no notebook “Técnicas de Manipulação de texto com NLP.ipynb”

Tokenização

O **word_tokenizer** é uma função que separa em “tokens” uma sentença.

```
exemplo = 'Eu gosto muito de música, e na minha opnião as melhores são dos anos 80 e 70! Você não acha?'
words = word_tokenize(exemplo)
words
```

```
['Eu',
 'gosto',
 'muito',
 'de',
 'música',
 ',',
 'e',
 'na',
 'minha',
 'opnião',
 'as',
 'melhores',
 'são',
 'dos',
 'anos',
 '80',
 'e',
 '70',
 '!',
 'Você',
 'não',
 'acha',
 '?']
```

Regex

Pode ser definida como uma **forma flexível de identificar determinada cadeia de caractere** para nosso interesse. Uma cadeia pode ser um caractere específico, uma palavra ou um padrão.

Como eu posso pesquisar por cada uma dessas palavras? Será que eu posso criar padrões de pesquisa?

Caneta
caneta
Canetas
canetas

Vamos fazer alguns exemplos em python e você também pode conferir mais opções em:

https://docs.python.org/3/library/re.html

Regex

Principais caracteres:

- Ponto (.): Em modo padrão, significa qualquer caractere, menos o de nova linha.
- Circunflexo (^): Em modo padrão, significa inicio da *string*.
- Cifrão ($): Em modo padrão, significa fim da *string*.
- Contra-barra (\): Caractere de escape, permite usar caracteres especiais como se fossem comuns.
- Colchetes ([]): Qualquer caractere dos listados entre os colchetes.
- Asterisco (*): Zero ou mais ocorrências da expressão anterior.
- Mais (+): Uma ou mais ocorrências da expressão anterior.
- Interrogação (?): Zero ou uma ocorrência da expressão anterior.
- Chaves ({n}): n ocorrências da expressão anterior.
- Barra vertical (|): "ou" lógico.
- Parenteses (()): Delimitam um grupo de expressões.
- \d: Dígito. Equivale a [0-9].
- \D: Não dígito. Equivale a [^0-9].
- \s: Qualquer caractere de espaçamento ([\t\n\r\f\v]).
- \S: Qualquer caractere que não seja de espaçamento.([^ \t\n\r\f\v]).
- \w: Caractere alfanumérico ou sublinhado ([a-zA-Z0-9_]).
- \W: Caractere que não seja alfanumérico ou sublinhado ([^a-zA-Z0-9_]).

Stop Words

Quando estamos analisando texto, algumas palavras podem ser irrelevantes na extração do contexto. A técnica **stop words basicamente mapeia um conjunto de palavras irrelevantes ou que não contribuem para o significado da fase**. Essa técnica consiste na remoção de ruídos do texto que são menos evidentes que pontuações, como os conectivos “que”, “o”, “a”, “de”, entre outros. Normalmente é um conjunto composto por artigos, advérbios, preposições e alguns verbos.

Exemplo:

Antes de utilizar stop words:
“Que dia lindo!”

Depois de utilizar stop words:
“dia lindo!”

Normalização de texto

Imagine a seguinte situação: Você precisa **extrair comentários positivos de uma pesquisa de satisfação referente ao produto XPTO** e ao analisar o dataset com as respostas você obteve o seguinte resultado:

ID	Comentario
453	Só produtos bonitos!!
888	Essa camiseta é bonita! Amei.
156	Bonito e útil.

Observe que a palavra **bonito** possui algumas variações nas respostas dos clientes: bonitos, bonita e bonitos.

Como podemos **extrair** esse comentário de **forma padrão** entre todas as instancias de dados?

Normalização de texto

Podemos normalizar a coluna “comentario” e extrair todas as variações da palavra bonito utilizando algumas técnicas**. Normalização de texto é a tarefa de ajustar palavras/tokens num formato padrão**.

Nessa técnica temos dois métodos diferentes: a lematização e a stemização.

Lematização

Basicamente é um processo que determina uma única “raiz” para a palavra, independente de suas diferenças superficiais.

ID	Comentario
453	Só produtos **bonit**os!!
888	Essa camiseta é **bonit**a! Amei.
156	**Bonit**o e útil.

Entretanto, NLTK não oferece suporte para lematização em português. Assim, vamos usar **Spacy**.

Normalização de texto

Stematização

Stemização é uma **versão simples da lematização** na qual realizamos **a remoção do sufixo de uma palavra**.

Exemplo:

Connection, **connect**ions, **connect**ive, **connect**ing, **connect**ed

Vale ressaltar que a stemização nem sempre irá funcionar bem com a língua portuguesa...

```
from nltk.stem import PorterStemmer

examples = ["conecta","conectado","conectamos","desconectados","conectividade"]

ps = PorterStemmer()

for word in examples:
  print(ps.stem(word))
```

```
conecta
conectado
conectamo
desconectado
conectividad
```

Normalização de texto

Felizmente, temos um em português que apresenta bons resultados: **RSLPStemmer.**

conecta, **conectado**, **conectamos**, **desconectados**, **conectividade**

```
from nltk.stem.rslp import RSLPStemmer

examples = ["conecta","conectado","conectamos","desconectados","conectividade"]

rslp = RSLPStemmer()

for word in examples:
  print(rslp.stem(word))
```

```
conect
conect
conect
desconect
conect
```

POS-Tagger

Na escola primária, aprendemos a diferença entre substantivo, verbos, advérbios e adjetivos. Tais classes gramaticais são categorias úteis para muitas tarefas de processamento de linguagem natural.

O processo de classificar palavras em classes gramaticais é chamado de Part-of-speech tagging, ou **POS-Tagging**.

Com o POS-tagging, é possível encontrar palavras que pertençam à mesma classe gramatical.

```
text = nltk.word_tokenize("A beautiful day for learn")
nltk.pos_tag(text) #tag em cada token
```

```
[('A', 'DT'),
 ('beautiful', 'JJ'),
 ('day', 'NN'),
 ('for', 'IN'),
 ('learn', 'NN')]
```

TextBlob

TextBlob é uma biblioteca Python voltada para o processamento de dados textuais. TextBlob foi criado com base nas libs NLTK e Pattern e tem por objetivo prover uma interface simples para as funções do NLTK. Vale lembrar que o TextBlob só possui suporte para a língua inglesa.

Podemos utilizar o TextBlob para classificar textos por exemplo:

```
from textblob import TextBlob
from textblob.sentiments import NaiveBayesAnalyzer

nltk.download('movie_reviews')
opinion = TextBlob("This movie was horrible!", analyzer=NaiveBayesAnalyzer())
opinion.sentiment
```

```
[nltk_data] Downloading package movie_reviews to
[nltk_data]     C:\Users\cl1476\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\movie_reviews.zip.
```

```
Sentiment(classification='neg', p_pos=0.3205526272161921, p_neg=0.6794473727838077)
```

Spacy

Uma das grandes vantagens do SpaCy é que ele tem modelos treinados em português disponíveis para download. Com o modelo da língua portuguesa carregado, acessar o POS Tag dos tokens é tão simples quanto acessar um atributo.

```
import spacy
nlp = spacy.load('pt_core_news_sm')
doc = nlp(u'Os Beatkes foi uma banda inglesa que revolucionou.')
print([token.orth_ for token in doc])
```

```
['Os', 'Beatkes', 'foi', 'uma', 'banda', 'inglesa', 'que', 'revolucionou', '.']
```

```
[(token.orth_, token.pos_) for token in doc]
```

```
[('Os', 'DET'),
 ('Beatkes', 'PROPN'),
 ('foi', 'AUX'),
 ('uma', 'DET'),
 ('banda', 'NOUN'),
 ('inglesa', 'ADJ'),
 ('que', 'PRON'),
 ('revolucionou', 'VERB'),
 ('.', 'PUNCT')]
```

Obrigada!

Ana Raquel

linkedin.com/ana-raquel-fernandes-cunha