



# Deep Learning para NLP

Aula 2 - Classificação de texto e POS Tagging com NLTK

Marlesson Santana



Marlesson Santana



<https://www.linkedin.com/in/marlesson-santana-25019358/>

- Cientista de Dados na **CQuantt**
- Pesquisador no **Deep Learning Brasil**
- **Doutorando** em Machine Learning - UFG



**CQuantt**

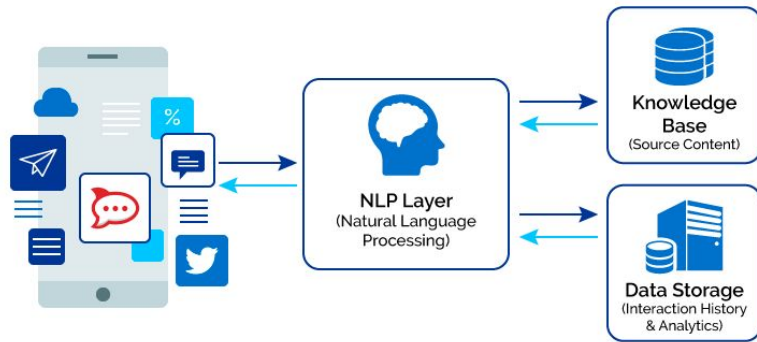
#datascience #machinelearning #deeplearning #nlp  
#python #spark #data-viz ...

# Roteiro

- Introdução NLP
- Pipeline de NLP Clássico
- Classificação de Texto
- POS Tagging
- Conclusão

# Introdução NLP

**NLP** é uma subárea da inteligência artificial que estuda a geração e **compreensão de linguagem humana natural** com objetivo de fornecer aos computadores a capacidade de “**entender**” e compor linguagem.



- Compreensão de **textos** e **áudios**
- Análise de contexto, sentimento, semântica, **extração de informação.. etc**

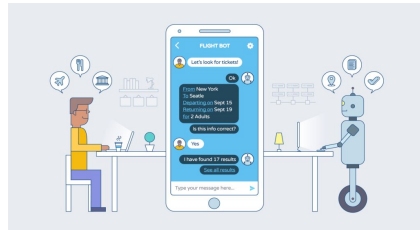
# Introdução NLP

As aplicações são nas mais diversas áreas. Ainda tem muito o que explorar pois a maior parte da informação disponível está em formato ***não estruturado***.

Assistentes Pessoais  
Coisas que pensam



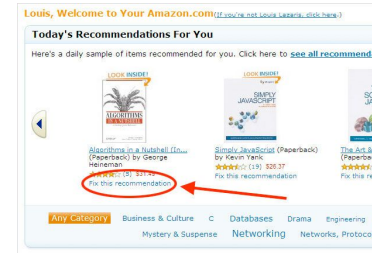
Assistentes Pessoais



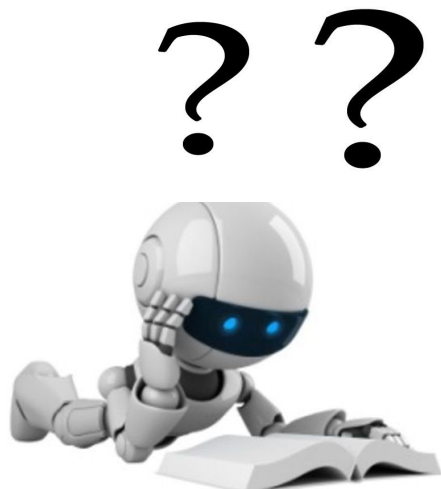
Chatbots



Análise de Sentimento



Sistemas de  
Recomendação



0,758	0,001	0,943	0,758	0,84	0,924
0,34	0,014	0,536	0,34	0,416	0,777
0,074	0,002	0,327	0,074	0,121	0,725
0,089	0,007	0,462	0,089	0,15	0,834
0,918	0,67	0,542	0,918	0,681	0,675
0,107	0,012	0,346	0,107	0,163	0,675
0,22	0,008	0,462	0,22	0,298	0,772
0,019	0,005	0,425	0,019	0,036	0,824
0,242	0,012	0,455	0,242	0,316	0,805
0,496	0,046	0,55	0,496	0,522	0,778

[illegible]

# Introdução NLP

O conceito de **semântica** bem aplicado em NLP otimiza o processo de estruturação da informação.

- sinônimos,
- conjugação verbal,
- erros de sintaxe...

“*Goiânia* está  **muito quente...**”

“*Goiânia* está **fazendo muinto calor....**”

“**GYN** está quente **pakas**”



# Introdução NLP

O conceito de **contexto** é importante para o entendimento correto da informação. Exemplo:

*“... estamos sem **ar** ...”*

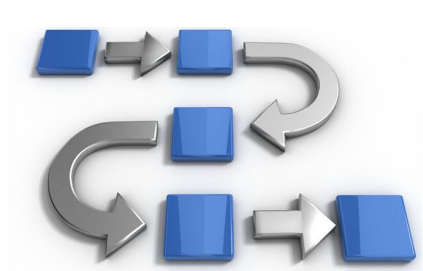
- a) “... para respirar.”*
- b) “... condicionado.”*
- c) “... puro.”*

# Introdução NLP

O conceito de **contexto** é importante para o entendimento correto da informação. Exemplo:

*“Hoje fez muito **calor**, infelizmente estamos sem **ar** ...”*

- a) *“... para respirar.”*
- b) *“... **condicionado**.”*
- c) *“... puro.”*


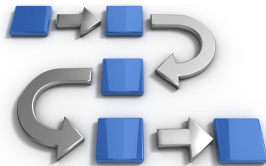


## Pipeline de NLP Clássico

# Pipeline de NLP

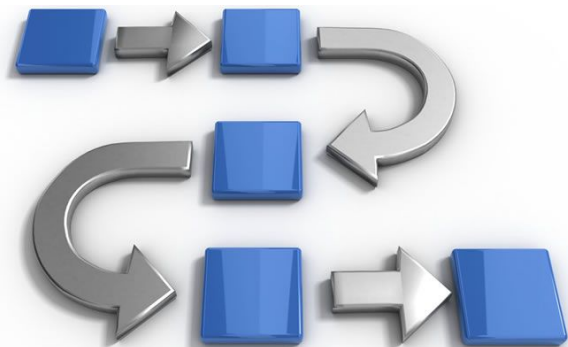
Em um processo de **NLP** o dado original passa por um pipeline de transformações e é estruturado para que as **features** de cada documento sejam observadas.

Sejam observadas.

Dado Original (Texto)	NLP	Dado Estruturado Features						
		A	B	C	D	E	F	
		D1	0,758	0,001	0,943	0,758	0,84	0,924
		D2	0,34	0,014	0,536	0,34	0,416	0,777
		D3	0,074	0,002	0,327	0,074	0,121	0,725
		D4	0,089	0,007	0,462	0,089	0,15	0,834
		D5	0,918	0,67	0,542	0,918	0,681	0,675
		D6	0,107	0,012	0,346	0,107	0,163	0,675
		D7	0,22	0,008	0,462	0,22	0,298	0,772
		D8	0,019	0,005	0,425	0,019	0,036	0,824
		D9	0,242	0,012	0,455	0,242	0,316	0,805
		D10	0,496	0,046	0,55	0,496	0,522	0,778

UNIVERSIDADE  
FEDERAL DE GOIÁS

# Pipeline de NLP



O Pipeline de NLP **executa diferentes tarefas** para que seja possível transformar o dado. São algumas fases:

- Tokenização
- Limpeza
- Normalização
- Vetorização

# 1 Tokenização

Em um **texto**, o que pode ser uma **característica**? Quais as unidades **semânticas**?

```
documents = ["jazz tem um ritmo de swing",  
             "swing é difícil de explicar",  
             "ritmo de swing é um ritmo natural"]
```

# 1 Tokenização

Em um **texto**, o que pode ser uma **característica**? Quais as unidades **semânticas**?

```
documents = ["jazz tem um ritmo de swing",  
             "swing é difícil de explicar",  
             "ritmo de swing é um ritmo natural"]
```

**Sentenças** carregam muita informação, mas as **Palavras** carregam a menor unidade de **informação semântica** em um texto.

# 1 Tokenização - Bag-of-words

**BoW** é o processo **mais básico** para transformar texto em vetor de características. É uma representação simples e rápida

```
documents = ["jazz tem um ritmo de swing",  
             "swing é difícil de explicar",  
             "ritmo de swing é um ritmo natural"]
```



```
features = ['de', 'difícil', 'explicar',  
            'jazz', 'natural', 'ritmo',  
            'swing', 'tem', 'um']
```



# 1 Tokenização - Bag-of-words

```
documents = ["jazz tem um ritmo de swing",  
            "swing é difícil de explicar",  
            "ritmo de swing é um ritmo natural"]
```



```
features = ['de', 'difícil', 'explicar',  
            'jazz', 'natural', 'ritmo',  
            'swing', 'tem', 'um']
```



Informação estruturada com **BoW** e Frequência do Termo (**TF**)

		'de'	'difícil'	'explicar'	'jazz'	'natural'	'ritmo'	'swing'	'tem'	'um'								
[document1,	1	,	0	,	0	,	1	,	1	,	1	,	1	]				
[document1,	1	,	1	,	1	,	0	,	0	,	0	,	1	,	0	,	0	]
[document1,	1	,	1	,	0	,	0	,	1	,	2	,	1	,	0	,	1	]

# 1 Tokenização - Bag-of-words - Desvantagens

- Ignora o **contexto** em que as palavras aparecem

```
"sem risos e com muitos momentos chatos"
```

```
['chatos', 'com', 'momentos', 'muitos', 'risos', 'sem']
```

```
"com muitos risos e sem momentos chatos"
```

```
['chatos', 'com', 'momentos', 'muitos', 'risos', 'sem']
```

- **Negação** é um problema..

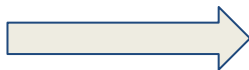
```
"O livro carece de inteligência e humor"
```

# 1 Tokenização - n-gramas

O **n-gramas** tenta minimizar o problema do **contexto** ao concatenar termos na geração dos tokens

```
documents = ["jazz tem um ritmo de swing",  
            "swing é difícil de explicar",  
            "ritmo de swing é um ritmo natural"]
```

3-gramas

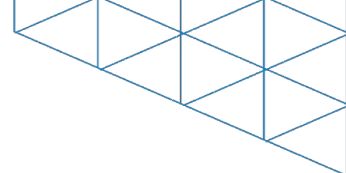


```
['de', 'de explicar', 'de swing',  
 'de swing um', 'difícil', 'difícil de',  
 'difícil de explicar', 'explicar',  
 'jazz', 'jazz tem', 'jazz tem um',  
 'natural', 'ritmo', 'ritmo de',  
 'ritmo de swing', 'ritmo natural',  
 'swing', 'swing difícil', 'swing difícil de',  
 'swing um', 'swing um ritmo', 'tem',  
 'tem um', 'tem um ritmo', 'um',  
 'um ritmo', 'um ritmo de', 'um ritmo natural']
```

# 1 Tokenização - n-gramas

O **n-gramas** tenta minimizar o problema do **contexto** ao concatenar termos na geração dos tokens





## 1 Tokenização - n-gramas - Desvantagens

- Aumenta a quantidade de tokens rapidamente
- O **contexto** adicionado ao concatenar os tempos é limitado e local
- Com  $n=1$  se torna o caso base do **BoW**

# 1 Tokenização - Sentence Tokenize

A tokenização por sentença pode ser útil em alguns contextos em que poucas sentenças descrevem o texto completo. Ex: Resumo de artigos científicos.

```
sentence = """Carpe diem. Aproveitem o dia de hoje, rapazes.  
Tornem as suas vidas extraordinárias."""
```

```
# Sentence Token  
sent_tokenize(sentence)
```

```
['Carpe diem.',  
'Aproveitem o dia de hoje, rapazes.',  
'Tornem as suas vidas extraordinárias.']
```

## 2 Limpeza

O processo de limpeza **envolve diversas etapas** e tem como objetivo a **remoção de tokens pouco significativos** e melhorar a representatividade dos tokens existentes.

- Remoção de Stop Words
- Padronização
- Normalização Lexical

```
['de', 'de explicar', 'de swing',  
'de swing um', 'difícil', 'difícil de',  
'difícil de explicar', 'explicar',  
'jazz', 'jazz tem', 'jazz tem um',  
'natural', 'ritmo', 'ritmo de',  
'ritmo de swing', 'ritmo natural',  
'swing', 'swing difícil', 'swing difícil de',  
'swing um', 'swing um ritmo', 'tem',  
'tem um', 'tem um ritmo', 'um',  
'um ritmo', 'um ritmo de', 'um ritmo natural']
```



```
features = ['de', 'difícil', 'explicar',  
'jazz', 'natural', 'ritmo',  
'swing', 'tem', 'um']
```

## 2 Limpeza - Stop Words

A remoção de ***stop words*** retira tokens que **não carregam informação útil** para caracterizar o documento. Pode ser feito por **lista fixa** ou baseado em **heurísticas** (números, símbolos..)

Quais tokens são “**inúteis**” ?

```
features = ['de', 'difícil', 'explicar',  
            'jazz', 'natural', 'ritmo',  
            'swing', 'tem', 'um']
```





## 2 Limpeza - Padronização

A padronização tenta unificar conceitos semânticos em um único token.  
Exemplo:

- **Geração de Entidade Nomeada**

- **GoT** = Game of Thrones = Série da Rainha Louca que coloca foco em tudo
- **Presidente** = Bolsonaro = Jair Bolsonaro

- **Modelo de Tópico**

- **Política:** senador, presidente, câmara
- **Filmes:** Star Wars, Senhor dos Anéis, De volta para o Futuro



## 2 Limpeza - Padronização

- **Correção textual**
  - **você** = voce = vc,
  - **fosse** = foçe,
  - **rapidez** = rapideis

Em alguns contextos (twitter) é muito importante a etapa de correção textual.



## 2 Limpeza - Normalização Lexical

O processo de **normalização lexical** de **stemização** ou **lematização** consistem em reduzir uma palavra ao seu **radical** ou ao seu **lema**. Ex:

- meninas = "**menin**"
- gato, gatos, gata, gatas = "**gat**"
- gato, gatos, gata, gatas = "**gato**"
- VERBO = **infinitivo**

A principal vantagem de aplicar a **stemização e lematização** é a redução de vocabulário e abstração de significado.

## 2 Limpeza - Normalização Lexical

Presente	Pretérito Imperfeito	Pretérito Perfeito
eu estudo tu estudas ele estuda nós estudamos vós estudais eles estudam	eu estudava tu estudavas ele estudava nós estudávamos vós estudáveis eles estudavam	eu estudei tu estudaste ele estudou nós estudamos vós estudastes eles estudaram
Pretérito Mais-que-perfeito	Futuro do Presente	Futuro do Pretérito
eu estudara tu estudaras ele estudara nós estudáramos vós estudáreis eles estudaram	eu estudarei tu estudarás ele estudará nós estudaremos vós estudareis eles estudarão	eu estudaria tu estudarias ele estudaria nós estudaríamos vós estudaríeis eles estudariam

# Tokenização e Limpeza

O processo de **tokenização** e **limpeza** são **dependentes** da linguagem, do *corpus*, do momento no tempo, da aplicação....



## 4 Vetorização

A metodologia de **vetorização** dá o **valor** do **token** para um documento. É uma medida de grandeza associada ao token.

Dado Estruturado						
<i>Features</i>						
	A	B	C	D	E	F
D1	0,758	0,001	0,943	0,758	0,84	0,924
D2	0,34	0,014	0,536	0,34	0,416	0,777
D3	0,074	0,002	0,327	0,074	0,121	0,725
D4	0,089	0,007	0,462	0,089	0,15	0,834
D5	0,918	0,67	0,542	0,918	0,681	0,675
D6	0,107	0,012	0,346	0,107	0,163	0,675
D7	0,22	0,008	0,462	0,22	0,298	0,772
D8	0,019	0,005	0,425	0,019	0,036	0,824
D9	0,242	0,012	0,455	0,242	0,316	0,805
D10	0,496	0,046	0,55	0,496	0,522	0,778

## 4 Vetorização - Binária

Se o token está presente no documento. Cria uma matriz binária de representação (0 ou 1)

```
documents = ["jazz tem um ritmo de swing",  
             "swing é difícil de explicar",  
             "ritmo de swing é um ritmo natural"]
```

	'de'	'difícil'	'explicar'	'jazz'	'natural'	'ritmo'	'swing'	'tem'	'um'
[	1	0	0	1	0	1	1	1	1]
[	1	1	1	0	0	0	1	0	0]
[	1	0	0	0	1	1	1	0	1]

## 4 Vetorização - Frequência do Termo (TF)

**Quantidade de tokens** presentes no documento. Medida de grandeza e importância do token para o documento.

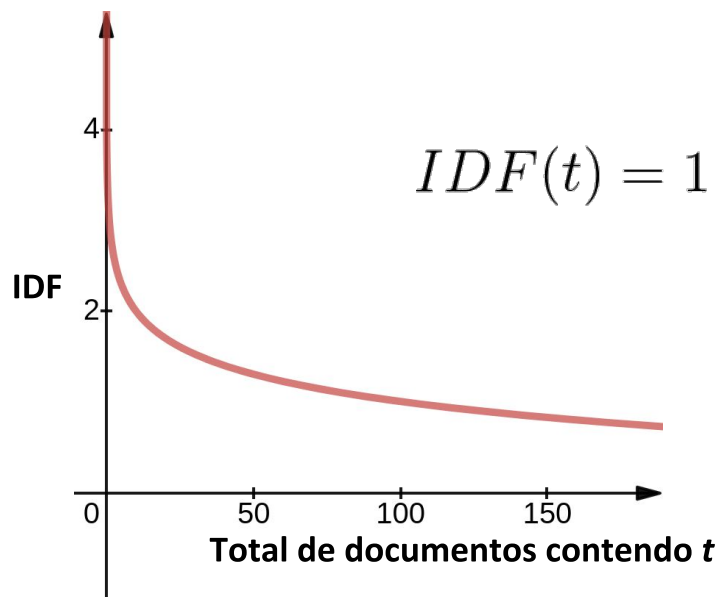
```
documents = ["jazz tem um ritmo de swing",  
            "swing é difícil de explicar",  
            "ritmo de swing é um ritmo natural"]
```

	'de'	'difícil'	'explicar'	'jazz'	'natural'	'ritmo'	'swing'	'tem'	'um'
[	1	0	0	1	0	1	1	1	1]
[	1	1	1	0	0	0	1	0	0]
[	1	0	0	0	1	2	1	0	1]



## Vetorização - TF\*IDF

Enquanto o **TF** mensura “*a relevância de um termo para o documento*”, o Frequência Inversa de Documento (**IDF**) mede “*o quanto o termo é frequente no corpus*”



$$IDF(t) = 1 + \log\left(\frac{\text{total de documentos}}{\text{total de documentos contendo } t}\right)$$

## 4 Vetorização - TF\*IDF

**TF\*IDF** estabelece uma **ponderação** aos tokens presentes em um documento **ao relacionar com o corpus**. Tokens que aparecem muito no corpus tem relevância baixa, tokens que aparecem pouco tem maior relevância.


$$TFIDF(t, d) = TF(t, d) * IDF(t)$$

## Vetorização - TF\*IDF

**TF\*IDF** estabelece uma **ponderação** aos tokens presentes em um documento **ao relacionar com o corpus**. Tokens que aparecem muito no corpus tem relevância baixa, tokens que aparecem pouco tem maior relevância.

```
documents = ["jazz tem um ritmo de swing",
             "swing é difícil de explicar",
             "ritmo de swing é um ritmo natural"]
```

$$\begin{aligned}
 &= TF('tem', 'documento1') * IDF('tem') \\
 &= 1 * (1 + \ln(3=1)) \\
 &= 2,09
 \end{aligned}$$



	'de'	'difícil'	'explicar'	'jazz'	'natural'	'ritmo'	'swing'	'tem'	'um'
[1.0	0.0	0.0	2.09	0.0	1.40	1.0	2.09	1.40]	
[1.0	2.09	2.09	0.0	0.0	0.0	1.0	0.0	0.0]	
[1.0	0.0	0.0	0.0	2.09	2.81	1.0	0.0	1.40]	

# Pipeline de NLP Clássico

```
documents = ["jazz tem um ritmo de swing",  
            "swing é difícil de explicar",  
            "ritmo de swing é um ritmo natural"]
```

**Tokenização**

BoW  
n-gramas

**Limpeza**

Stop words  
Stemming  
Correção Lexical

**Vetorização**

Binário  
TF  
TF-IDF

```
['de' 'difícil' 'explicar' 'jazz' 'natural' 'ritmo' 'swing' 'tem' 'um']  
[1.0  0.0  0.0  2.09  0.0  1.40  1.0  2.09  1.40]  
[1.0  2.09  2.09  0.0  0.0  0.0  1.0  0.0  0.0 ]  
[1.0  0.0  0.0  0.0  2.09  2.81  1.0  0.0  1.40]
```

Existem diversas bibliotecas que podem ajudar nas etapas de pipeline de NLP.

# Pipeline de NLP Clássico



Natural Language Analysis  
with Python NLTK

<https://www.nltk.org/>

gensim

<https://radimrehurek.com/gensim/>

spaCy

<https://spacy.io/>



<https://scikit-learn.org/stable/>

```
from nltk.tokenize import word_tokenize
from nltk.stem.snowball import SnowballStemmer
from sklearn.feature_extraction.text import TfidfVectorizer
import pandas as pd
```

```
documentos = [
    "O rei roeu a rolpa do rato de roma",
    "Deep Learning é muito bom, vem estudar com a gente",
    "A casa amarela em roma é verde"
]
```

```
# Tokenização
def tokenize(text):
    # Tokenização
    tokens = nltk.word_tokenize(text)

    # Stemização
    stems = []
    for item in tokens:
        stems.append(SnowballStemmer("portuguese").stem(item))
    return stems
```

```
tokenize(documentos[0])
```

```
['o', 'rei', 'roeu', 'a', 'rolp', 'do', 'rat', 'de', 'rom']
```

```
# StopWords
#
stop_words = ['de', 'a', 'o', 'é', 'do', ',', '']

# Vetorização
tfidf = TfidfVectorizer(tokenizer=tokenize, stop_words=stop_words,

# Features
tfs = pd.DataFrame(tfidf.fit_transform(documentos).todense(),
                    columns=tfidf.get_feature_names())
tfs
```

	amarel	bom	cas	com	deep	em	estud	gent	learning
0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
1	0.000000	0.353553	0.000000	0.353553	0.353553	0.000000	0.353553	0.353553	0.3
2	0.467351	0.000000	0.467351	0.000000	0.000000	0.467351	0.000000	0.000000	0.0

# Pipeline de NLP Clássico



Natural Language Analysis  
with Python NLTK

<https://www.nltk.org/>

gensim

<https://radimrehurek.com/gensim/>

spaCy

<https://spacy.io/>



<https://scikit-learn.org/stable/>

Comparison of Python NLP libraries Pros and Cons		
	⊕ PROS	⊖ CONS
 Natural Language Toolkit	<ul style="list-style-type: none"><li>• The most well-known and full NLP library</li><li>• Many third-party extensions</li><li>• Plenty of approaches to each NLP task</li><li>• Fast sentence tokenization</li><li>• Supports the largest number of languages compared to other libraries</li></ul>	<ul style="list-style-type: none"><li>- Complicated to learn and use</li><li>- Quite slow</li><li>- In sentence tokenization, NLTK only splits text by sentences, without analyzing the semantic structure</li><li>- Processes strings which is not very typical for object-oriented language Python</li><li>- Doesn't provide neural network models</li><li>- No integrated word vectors</li></ul>
 spaCy	<ul style="list-style-type: none"><li>• The fastest NLP framework</li><li>• Easy to learn and use because it has one single highly optimized tool for each task</li><li>• Processes objects; more object-oriented, comparing to other libs</li><li>• Uses neural networks for training some models</li><li>• Provides built-in word vectors</li><li>• Active support and development</li></ul>	<ul style="list-style-type: none"><li>- Lacks flexibility, comparing to NLTK</li><li>- Sentence tokenization is slower than in NLTK</li><li>- Doesn't support many languages. There are models only for 7 languages and "multi-language" models</li></ul>
 NLTK	<ul style="list-style-type: none"><li>• Has functions which help to use the bag-of-words method of creating features for the text classification problems</li><li>• Provides a wide variety of algorithms to build machine learning models</li><li>• Has good documentation and intuitive classes' methods</li></ul>	<ul style="list-style-type: none"><li>- For more sophisticated preprocessing things (for example, pos-tagging), you should use some other NLP library and only after it you can use models from scikit-learn</li><li>- Doesn't use neural networks for text preprocessing</li></ul>
 gensim	<ul style="list-style-type: none"><li>• Works with large datasets and processes data streams</li><li>• Provides tf-idf vectorization, word2vec, document2vec, latent semantic analysis, latent Dirichlet allocation</li><li>• Supports deep learning</li></ul>	<ul style="list-style-type: none"><li>- Designed primarily for unsupervised text modeling</li><li>- Doesn't have enough tools to provide full NLP pipeline, so should be used with some other library (Spacy or NLTK)</li></ul>
 Pattern	<ul style="list-style-type: none"><li>• Allows part-of-speech tagging, n-gram search, sentiment analysis, WordNet, vector space model, clustering and SVM</li><li>• There are web crawler, DOM parser, some APIs (like Twitter, Facebook etc.)</li></ul>	<ul style="list-style-type: none"><li>- Is a web miner; can be not enough optimized for some specific NLP tasks</li></ul>
 Polyglot	<ul style="list-style-type: none"><li>• Supports a large number of languages (16-196 languages for different tasks)</li></ul>	<ul style="list-style-type: none"><li>- Not as popular as, for example, NLTK or Spacy; can be slow issues solutions or weak community support</li></ul>
Created by ActiveWizards		

[https://activewizards.com/content/blog/Comparison\\_of\\_Python\\_NLP\\_libraries/nlp-librares-python-prs-and-cons01.png](https://activewizards.com/content/blog/Comparison_of_Python_NLP_libraries/nlp-librares-python-prs-and-cons01.png)

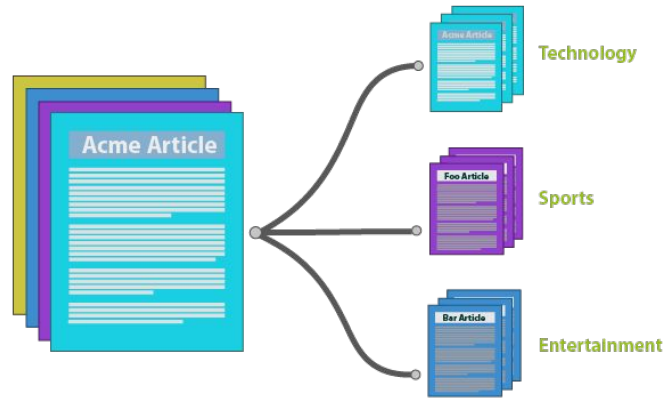
# Pipeline de NLP Clássico

## Dado Estruturado

0,758	0,001	0,943	0,758	0,84	0,924
0,34	0,014	0,536	0,34	0,416	0,777
0,074	0,002	0,327	0,074	0,121	0,725
0,089	0,007	0,462	0,089	0,15	0,834
0,918	0,67	0,542	0,918	0,681	0,675
0,107	0,012	0,346	0,107	0,163	0,675
0,22	0,008	0,462	0,22	0,298	0,772
0,019	0,005	0,425	0,019	0,036	0,824
0,242	0,012	0,455	0,242	0,316	0,805
0,496	0,046	0,55	0,496	0,522	0,778

## Machine Learning





## Classificação de Texto



# Classificação de Texto

O problema pode ser resolvido de forma **supervisionada** utilizando um **dataset rotulado**  $(x, y)$  para treinar um classificador capaz de extrair os padrões nas features  $(X)$ .

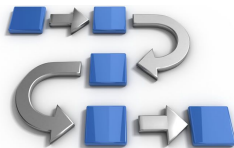


	title	category
9	Em teste, WhatsApp Business permite que empres...	tec
40	Por que é mais difícil para as mulheres lutar ...	equilibrioesaude
52	Compare as mensalidades de 1.104 escolas priva...	educacao
65	Em série sobre Revolução Russa, Mika Lins lê t...	tv
81	Programa espacial soviético teve ideal social ...	ciencia
124	Pesquisadores e estudantes marcham na Paulista...	ciencia
162	Educar aluno não é apenas ensinar conteúdo, di...	educacao
207	Livro conta os 'fracassos' de cientistas que c...	ciencia
285	Evento da Folha discute crise no financiamento...	ciencia
309	Cientista modifica forma de bactérias para com...	equilibrioesaude

# Classificação de Texto

O problema pode ser resolvido de forma **supervisionada** utilizando um **dataset rotulado** (x, y) para treinar um classificador capaz de extrair os padrões nas features (X).

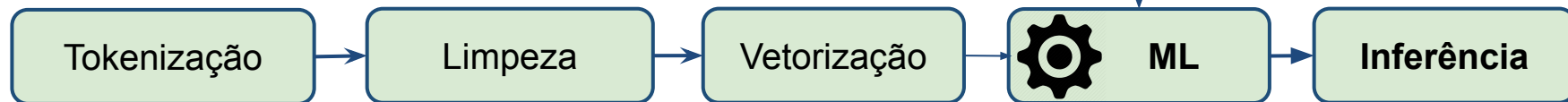
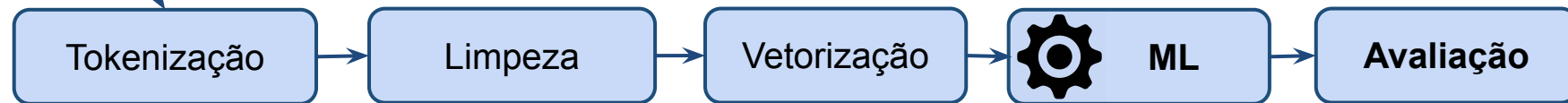
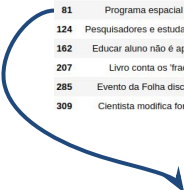
	title	category
9	Em teste, WhatsApp Business permite que empres...	tec
40	Por que é mais difícil para as mulheres lutar ...	equilibrioesaude
52	Compare as mensalidades de 1.104 escolas priva...	educacao
65	Em série sobre Revolução Russa, Mika Lins lê t...	tv
81	Programa espacial soviético teve ideal social ...	ciencia
124	Pesquisadores e estudantes marcham na Paulista...	ciencia
162	Educar aluno não é apenas ensinar conteúdo, di...	educacao
207	Livro conta os 'fracassos' de cientistas que c...	ciencia
285	Evento da Folha discute crise no financiamento...	ciencia
309	Cientista modifica forma de bactérias para com...	equilibrioesaude



	Features						Target
	A	B	C	D	E	F	
D1	0,758	0,001	0,943	0,758	0,84	0,924	Classe 1
D2	0,34	0,014	0,536	0,34	0,416	0,777	Classe 2
D3	0,074	0,002	0,327	0,074	0,121	0,725	Classe 1
D4	0,089	0,007	0,462	0,089	0,15	0,834	Classe 2
D5	0,918	0,67	0,542	0,918	0,681	0,675	Classe 2
D6	0,107	0,012	0,346	0,107	0,163	0,675	Classe 1
D7	0,22	0,008	0,462	0,22	0,298	0,772	Classe 1
D8	0,019	0,005	0,425	0,019	0,036	0,824	Classe 2
D9	0,242	0,012	0,455	0,242	0,316	0,805	Classe 2
D10	0,496	0,046	0,55	0,496	0,522	0,778	Classe 1

# Classificação de Texto

	title	category
9	Em teste, WhatsApp Business permite que empres...	tec
40	Por que é mais difícil para as mulheres lutar ...	equilibrioesaude
52	Compare as mensalidades de 1.104 escolas priva...	educacao
65	Em série sobre Revolução Russa, Mika Lins lê t...	tv
81	Programa espacial soviético teve ideal social ...	ciencia
124	Pesquisadores e estudantes marcham na Paulista...	ciencia
162	Educar aluno não é apenas ensinar conteúdo, di...	educacao
207	Livro conta os "tracassos" de cientistas que c...	ciencia
285	Evento da Folha discute crise no financiamento...	ciencia
309	Cientista modifica forma de bactérias para com...	equilibrioesaude



**RAW**

"0 livro carece de inteligência e humor"

# Classificação de Texto - Classificador

A qualidade da classificação está associada ao processo de NLP para extração de características do texto.

O Pipeline é bastante sensível e pode ser tunado de diversas formas.

	title	category
9	Em teste, WhatsApp Business permite que empres...	tec
40	Por que é mais difícil para as mulheres lutar ...	equilibrioesaude
52	Compare as mensalidades de 1.104 escolas priva...	educacao

```
# Split Dataset
X_train, X_test, y_train, y_test = train_test_split(df[['title']], df.category, random_state=42)
X_train.shape
```

```
(8302, 1)
```

```
# Stop Words
stop_words = nltk.corpus.stopwords.words('portuguese')

# NLP Pipeline
text_clf = Pipeline([
    # Vectorize
    ('vect', TfidfVectorizer(tokenizer=tokenize,
                             stop_words=stop_words,
                             ngram_range=(1,2))),
    # Classificador
    ('clf', KNeighborsClassifier(n_jobs=-1)),
])
```

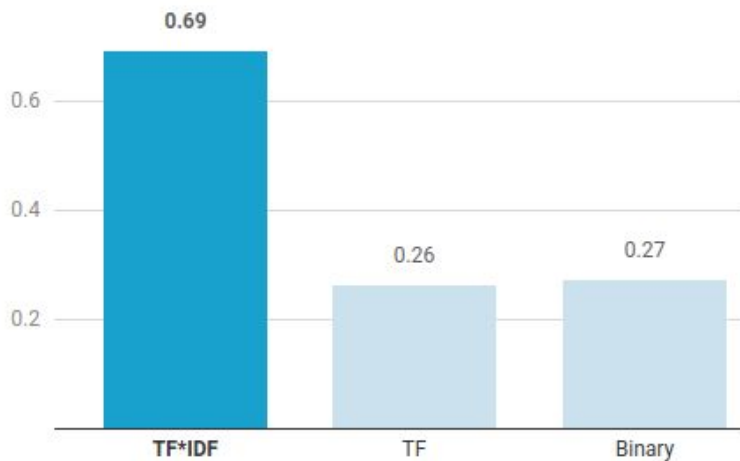
```
# Train
text_clf = text_clf.fit(X_train.title, y_train)
```

```
# Evaluate
text_clf.score(X_test.title, y_test)
```

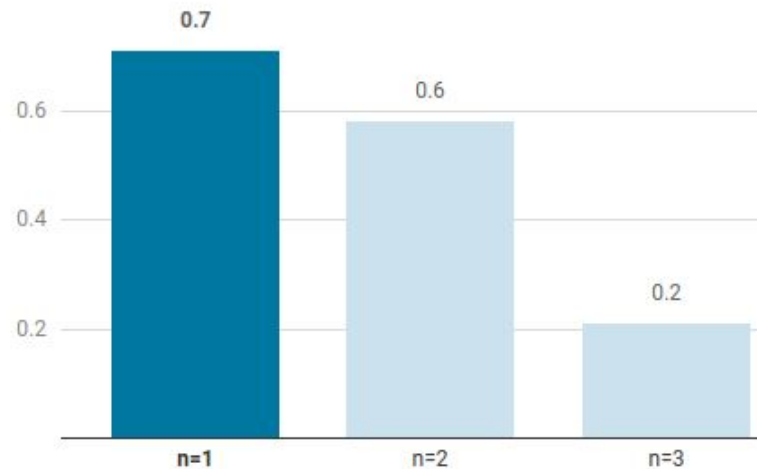
```
0.6940028901734104
```

# Classificação de Texto - Avaliação

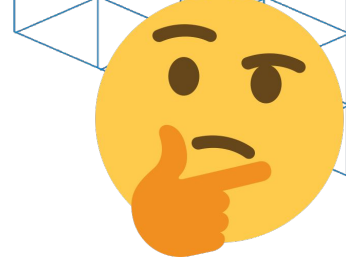
## Avaliação por Vetorização



## Avaliação por N-gramas



# Outros Problemas...

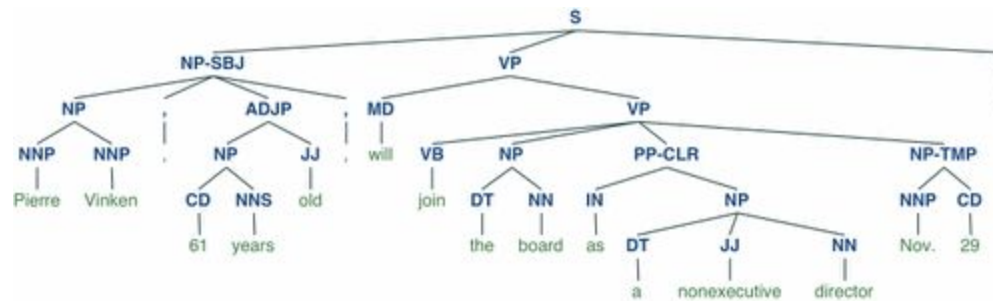


Como podemos utilizar essa modelagem em um **problema de busca** de documentos? - **Similaridade de Documentos**

	<i>Features</i>					
	A	B	C	D	E	F
D1	0,758	0,001	0,943	0,758	0,84	0,924
D2	0,34	0,014	0,536	0,34	0,416	0,777
D3	0,074	0,002	0,327	0,074	0,121	0,725
D4	0,089	0,007	0,462	0,089	0,15	0,834
D5	0,918	0,67	0,542	0,918	0,681	0,675
D6	0,107	0,012	0,346	0,107	0,163	0,675
D7	0,22	0,008	0,462	0,22	0,298	0,772
D8	0,019	0,005	0,425	0,019	0,036	0,824
D9	0,242	0,012	0,455	0,242	0,316	0,805
D10	0,496	0,046	0,55	0,496	0,522	0,778

Como podemos utilizar essa modelagem em um problema de **Clusterização de Documentos**?

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}},$$



## Part Of Speech (POS) Tagging

# POS Tagging

Refere-se à **categorização das palavras** em uma frase **em funções sintáticas** ou gramaticais específicas. A marcação POS é a tarefa de anexar uma dessas categorias a cada uma das palavras ou tokens em um texto.

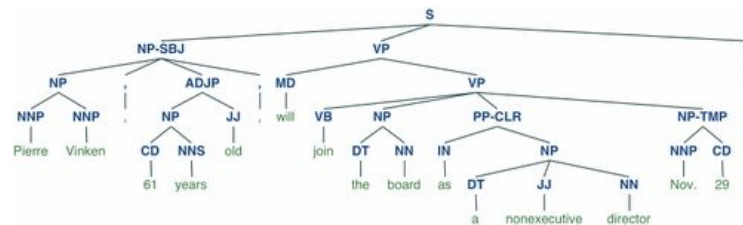




# POS Tagging

As principais aplicações de POS são:

- Extração de **Entidade Nomeada**
- Enriquecimento das informações sintáticas
- Novas **Features**



# POS Tagging - spaCy

```
import spacy
from spacy import displacy
```

```
# Load Pt Model
nlp = spacy.load('pt')
```

```
sentence = "O rato roeu a roupa do rei de Roma"
doc       = nlp(sentence)
```

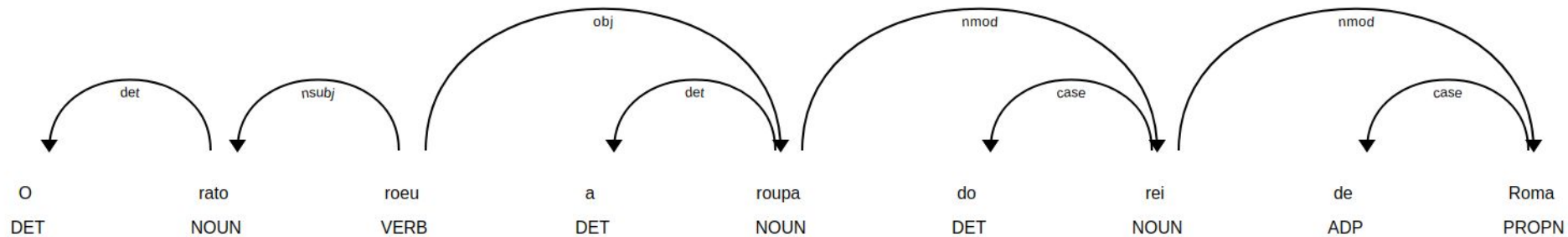
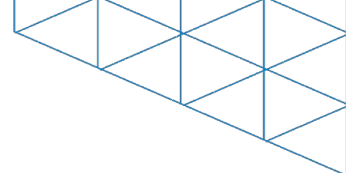
```
for token in doc:
    print((token.text, token.pos_))
```

```
('O', 'DET')
('rato', 'NOUN')
('roeu', 'VERB')
('a', 'DET')
('roupa', 'NOUN')
('do', 'DET')
('rei', 'NOUN')
('de', 'ADP')
('Roma', 'PROPN')
```

## Alphabetical listing

- [ADJ](#): adjective
- [ADP](#): adposition
- [ADV](#): adverb
- [AUX](#): auxiliary
- [CCONJ](#): coordinating conjunction
- [DET](#): determiner
- [INTJ](#): interjection
- [NOUN](#): noun
- [NUM](#): numeral
- [PART](#): particle
- [PRON](#): pronoun
- [PROPN](#): proper noun
- [PUNCT](#): punctuation
- [SCONJ](#): subordinating conjunction
- [SYM](#): symbol
- [VERB](#): verb
- [X](#): other

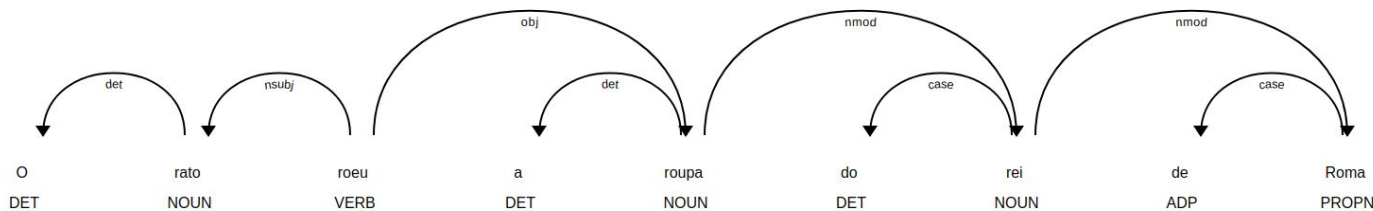
# POS Tagging - spaCy



# POS Tagging

Como podemos **modelar** uma solução de **POS Tagging com ML**?

- Problema supervisionado, não supervisionado ou RL?
- Quais as features? como montar o dataset?



# POS Tagging - Dataset

A **NLTK** oferece diferentes datasets em PT-BR com marcação de POS.

MAC-MORPHO conta com mais de **1 milhão** de palavras de textos jornalísticos

```
from nltk.corpus import mac_morpho

# print sample
print(mac_morpho.words(), len(mac_morpho.words()))

['Jersei', 'atinge', 'média', 'de', 'Cr$', '1,4', ...] 1170095
```

```
for t in mac_morpho.tagged_words()[:6]:
    print(t)
```

```
('Jersei', 'N')
('atinge', 'V')
('média', 'N')
('de', 'PREP')
('Cr$', 'CUR')
('1,4', 'NUM')
```

```
print(mac_morpho.tagged_sents()[0])
```

```
[('Jersei', 'N'), ('atinge', 'V'), ('média', 'N'), ('de', 'PREP'), ('Cr$', 'CUR'), ('1,4', 'NUM'), ('milhão', 'N'), ('em', 'PREP|+'), ('a', 'ART'), ('venda', 'N'), ('de', 'PREP|+'), ('a', 'ART'), ('Pinhal', 'NPRO P'), ('em', 'PREP'), ('São', 'NPRO P'), ('Paulo', 'NPRO P')]
```

'Jersei atinge média de Cr\$ 1,4 milhão em a venda de a Pinhal em São Paulo'

<http://nilc.icmc.usp.br/macmorpho/macmorpho-manual.pdf>

[https://github.com/tousif/nltk-gae/blob/master/nltk/test/portuguese.doctest\\_latin1](https://github.com/tousif/nltk-gae/blob/master/nltk/test/portuguese.doctest_latin1)

# POS Tagging - Features

```
def sentence_features(st, ix):
    d_ft = {}
    d_ft['word'] = st[ix]
    d_ft['dist_from_first'] = ix - 0
    d_ft['dist_from_last'] = len(st) - ix
    d_ft['capitalized'] = st[ix][0].upper() == st[ix][0]
    d_ft['prefix1'] = st[ix][0]
    d_ft['prefix2'] = st[ix][:2]
    d_ft['prefix3'] = st[ix][:3]
    d_ft['suffix1'] = st[ix][-1]
    d_ft['suffix2'] = st[ix][-2:]
    d_ft['suffix3'] = st[ix][-3:]
    d_ft['prev_word'] = '' if ix==0 else st[ix-1]
    d_ft['next_word'] = '' if ix==(len(st)-1) else st[ix+1]
    d_ft['numeric'] = st[ix].isdigit()
    return d_ft
```

As **features** de cada palavra dependem da posição da palavra na frase, da vizinhança, do prefixo e sufixo....

'Jersei atinge média de Cr\$ 1,4 milhão em a venda de a Pinhal em São Paulo'

	capitalized	dist_from_first	dist_from_last	next_word	numeric	prefix1	prefix2	prefix3	prev_word	suffix1	suffix2	suffix3	word
0	True	0	16	atinge	False	J	Je	Jer		i	ei	sei	Jersei
1	False	1	15	média	False	a	at	ati	Jersei	e	ge	nge	atinge
2	False	2	14	de	False	m	mé	méd	atinge	a	ia	dia	média
3	False	3	13	Cr\$	False	d	de	de	média	e	de	de	de
4	True	4	12	1,4	False	C	Cr	Cr\$	de	\$	r\$	Cr\$	Cr\$

# POS Tagging - Features - Transformação

	capitalized	dist_from_first	dist_from_last	next_word	numeric	prefix1	prefix2	prefix3	prev_word	suffix1	suffix2	suffix3	word
0	True	0	16	atinge	False	J	Je	Jer		i	ei	sei	Jersei
1	False	1	15	média	False	a	at	ati	Jersei	e	ge	nge	atinge
2	False	2	14	de	False	m	mê	méd	atinge	a	ia	dia	média
3	False	3	13	Cr\$	False	d	de	de	média	e	de	de	de
4	True	4	12	1,4	False	C	Cr	Cr\$	de	\$	r\$	Cr\$	Cr\$

Transforma as *features* categóricas usando **one-hot-enc** e gera um novo dataset contendo apenas números.

```
# https://scikit-learn.org/stable/modules/generated/sk.  
from sklearn.feature_extraction import DictVectorizer  
v = DictVectorizer(sparse=False)  
X_transformed = v.fit_transform(X)
```



	0	1	2	3	4	5	6	7	8	9	...	16871	16872	16873	16874	16875	16876	16877	16878	16879	16880
0	1.0	0.0	16.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	1.0	15.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	2.0	14.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	3.0	13.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	1.0	4.0	12.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 16881 columns



# POS Tagging - Features - Predição

	0	1	2	3	4	5	6	7	8	9	...	16871	16872	16873	16874	16875	16876	16877	16878	16879	16880
0	1.0	0.0	16.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	1.0	15.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	2.0	14.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	3.0	13.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	1.0	4.0	12.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 16881 columns

Com o **dataset rotulado** é possível **treinar um classificador** e utilizá-lo para inferir qualquer novo texto.

```
# train model
rf = model.fit(...) #
```

```
# Evaluate
predictions = rf.predict(X_test)
accuracy_score(y_test, predictions)
```

0.92675

```
# sentence
test_sentence = "0 rato roeu a roupa do rei de Roma"

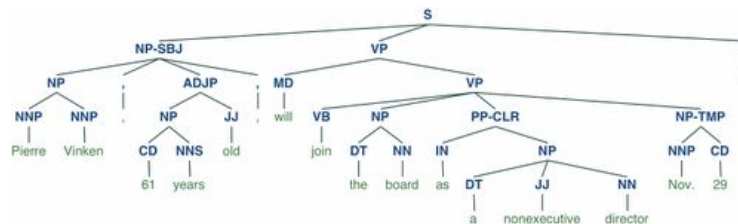
# predict
for tagged in predict_pos_tags(test_sentence.split()):
    print(tagged)
```

```
('0', 'ART')
('rato', 'N')
('roeu', 'V')
('a', 'ART')
('roupa', 'N')
('do', 'NPROP')
('rei', 'N')
('de', 'PREP')
('Roma', 'NPROP')
```

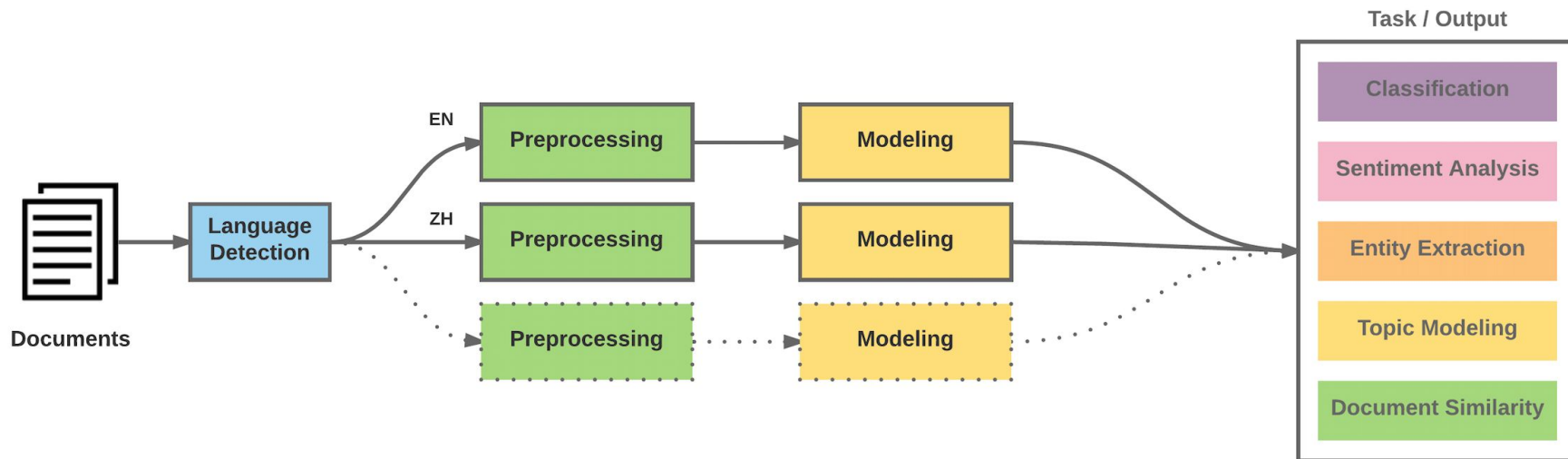


# POS Tagging - Conclusão

- O POS Tagging pode ser uma **etapa do Pipeline de NLP** para adicionar mais informação semântica ao problema.
- É um passo necessário para o problema de Extração de Entidade Nomeada.



# Pipeline de NLP - Conclusão



# Conclusão

- **Pipeline de NLP clássico é uma tarefa árdua**, cheia de detalhes, sensível a diferentes problemas e geralmente “overfitada” a um único problema. (Br sofre...)
- **Existem diferentes bibliotecas** que podem ajudar e simplificar esse processo. É importante dominar algumas delas.
- No curso ficará visível os motivos de Deep Learning ter dominado a área de NLP. Mas **NLP clássico ainda tem seu valor** e pode ser utilizado como um baseline para seu problema.

# Links úteis

- <https://github.com/marlesson/NLP-DeepLearningBrasil-Aula2>
- <https://towardsdatascience.com/machine-learning-nlp-text-classification-using-scikit-learn-python-and-nltk-c52b92a7c73a>
- <https://spacy.io/usage/linguistic-features>
- [http://www.nltk.org/howto/portuguese\\_en.html](http://www.nltk.org/howto/portuguese_en.html)
- <https://leportella.com/pt-br/2017/11/30/brincando-de-nlp-com-spacy.html>
- <https://medium.com/greyatom/learning-pos-tagging-chunking-in-nlp-85f7f811a8cb>
- <https://sunscrapers.com/blog/6-best-python-natural-language-processing-nlp-libraries/>
- <https://leportella.com/pt-br/2017/11/30/brincando-de-nlp-com-spacy.html>

# Obrigado(a)!