

Inteligência Artificial

Curso Ciência da Computação

Prof. Dr. Domingos Márcio Rodrigues Napolitano

Sem 2 / 2019



Introdução ao Processamento de Linguagem Natural

O que é PNL (Processamento de Linguagem Natural)?

- A PNL é um subcampo da ciência da computação e da inteligência artificial, preocupado com as interações entre computadores e linguagens humanas (naturais).
- É usado para aplicar algoritmos de aprendizado de máquina ao texto e à fala.
- Por exemplo, podemos usar a PNL para criar sistemas como:
 - reconhecimento de fala,
 - resumo de documentos,
 - tradução automática,
 - detecção de spam,
 - reconhecimento de entidade nomeada,
 - resposta a perguntas,
 - preenchimento automático,
 - digitação preditiva etc.
- Atualmente, a maioria de nós possui smartphones com reconhecimento de fala. Esses smartphones usam a PNL para entender o que é dito.

- O Microsoft OS possui um assistente virtual chamado Cortana que pode reconhecer uma voz natural. Você pode usá-lo para configurar lembretes, abrir aplicativos, enviar e-mails, jogar, acompanhar voos e pacotes, verificar o tempo e assim por diante.

- Você pode ler mais sobre os comandos da Cortana em:

<https://www.howtogeek.com/225458/15-things-you-can-do-with-cortana-on-windows-10/>

https://www.youtube.com/watch?v=HHdN1E8rm_I

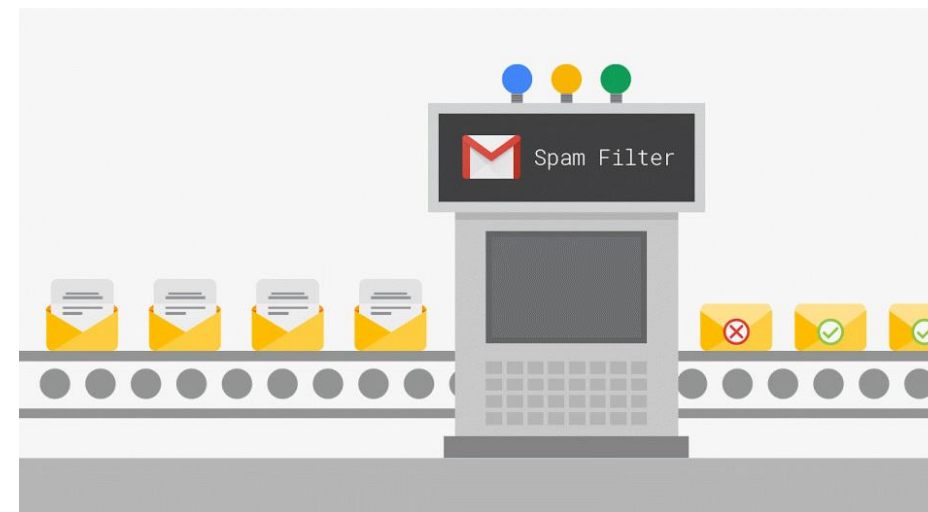


- Siri é um assistente virtual dos sistemas operacionais iOS, watchOS, macOS, HomePod e tvOS da Apple Inc. Novamente, você pode fazer muitas coisas com comandos de voz:
 - iniciar uma chamada,
 - enviar uma mensagem de texto para alguém,
 - enviar um email,
 - definir um cronômetro,
 - tirar uma foto,
 - abrir um aplicativo,
 - definir um alarme,
 - usar a navegação e assim por diante.



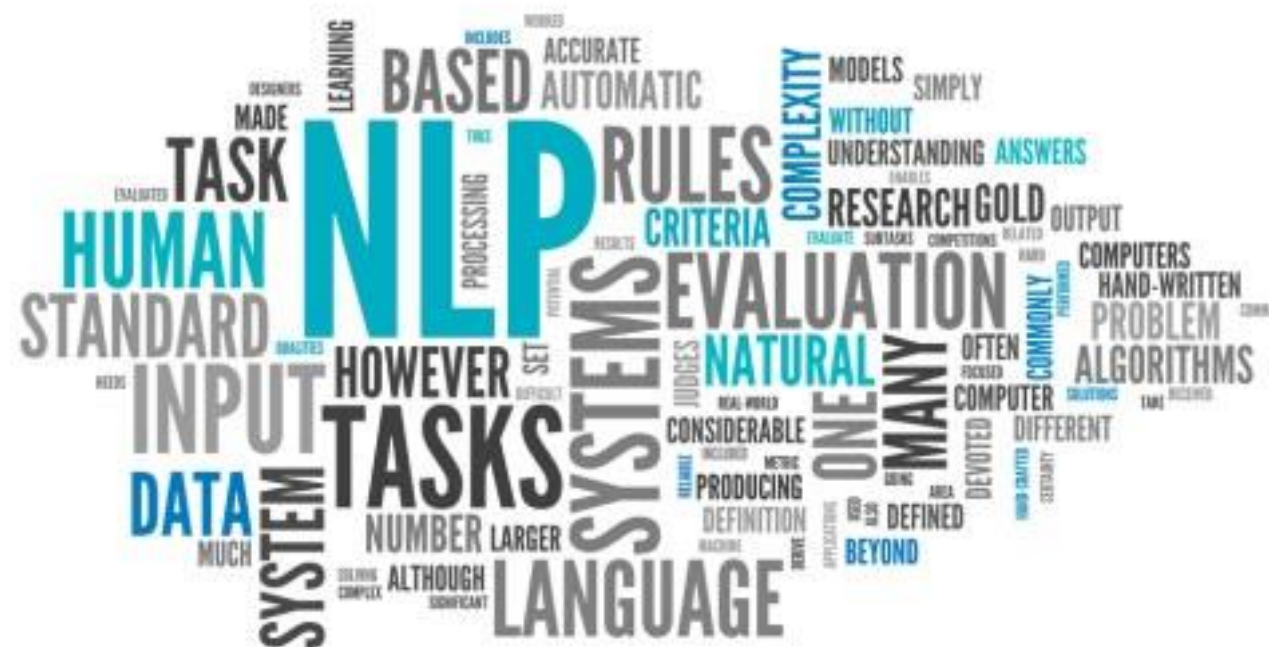
- Você pode saber mais sobre o SIRI aqui:
<https://www.cnet.com/how-to/the-complete-list-of-siri-commands/>

- Quando lhe enviam um e-mail no gmail, ele passa pelo funil apresentado ao lado.
- Se a mensagem passar pelos critérios de filtragem, ela chega na sua caixa de entrada, caso contrário, ela vai para a pasta de spam.
- Há outros fatores que a inteligência artificial do filtro leva em conta para avaliar, que vão muito além de um mero filtro de spam.
- Todos os dados a seu respeito são utilizados.



- O NLTK (Natural Language Toolkit) é uma plataforma líder para a criação de programas Python para trabalhar com dados da linguagem humana.
- Ele fornece interfaces fáceis de usar para muitos corpora e recursos lexicais.
- Além disso, ele contém um conjunto de bibliotecas de processamento de texto para classificação, tokenização, stemming, marcação, análise e raciocínio semântico.
- Usaremos este kit de ferramentas para mostrar alguns conceitos básicos do campo de processamento de idioma natural.
- Para os exemplos abaixo, é necessária a importação do kit de ferramentas NLTK.
- Podemos fazer o seguinte:
`import nltk.`

- Nesta aula, vamos falar sobre seguintes tópicos:
- Tokenização de sentenças
- Tokenização do Word
- Lematização e Stemming de Texto
- Stop Words
- Regex
- Bag-of-Words
- TF-IDF



- A tokenização de sentenças (também chamada de segmentação de sentenças) é o problema de dividir uma sequência de linguagem escrita em suas frases componentes.
- A ideia aqui parece muito simples.
- Em inglês e em alguns outros idiomas, podemos dividir as frases sempre que virmos um sinal de pontuação.
- No entanto, mesmo esse problema não é trivial devido ao uso do caractere ponto final para abreviações.
- Ao processar texto sem formatação, as tabelas de abreviações que contêm períodos podem nos ajudar a impedir a atribuição incorreta dos limites das frases.

- Vejamos um trecho de texto sobre um famoso jogo de tabuleiro chamado gamão.



Backgammon is one of the oldest known board games. Its history can be traced back nearly 5,000 years to archeological discoveries in the Middle East. It is a two player game where each player has fifteen checkers which move between twenty-four points according to the roll of two dice.

Gamão é um dos mais antigos jogos de tabuleiro conhecidos. Sua história remonta quase 5.000 anos às descobertas arqueológicas no Oriente Médio. É um jogo para dois jogadores, onde cada jogador tem quinze damas que se movem entre vinte e quatro pontos de acordo com o lançamento de dois dados.

Resultado (English):

Backgammon is one of the oldest known board games.

Its history can be traced back nearly 5,000 years to archeological discoveries in the Middle East.

It is a two player game where each player has fifteen checkers which move between twenty-four points according to the roll of two dice.

Resultado (Português):

Gamão é um dos mais antigos jogos de tabuleiro conhecidos.

Sua história remonta quase 5.000 anos às descobertas arqueológicas no Oriente Médio.

É um jogo para dois jogadores, onde cada jogador tem quinze damas que se movem entre vinte e quatro pontos de acordo com o lançamento de dois dados.

- A tokenização de palavras (também chamada de segmentação de palavras) é o problema de dividir uma sequência de linguagem escrita em suas palavras componentes.
- Em inglês e em muitos outros idiomas usando alguma forma de alfabeto latino, o espaço é uma boa aproximação de um divisor de palavras.
- No entanto, ainda podemos ter problemas se apenas dividirmos por espaço para alcançar os resultados desejados.
- Alguns substantivos compostos em inglês são escritos de forma variável e, às vezes, contêm um espaço.
- Na maioria dos casos, usamos uma biblioteca para alcançar os resultados desejados.
- Portanto, não se preocupe demais com os detalhes.

Resultado (English):

Backgammon is one of the oldest known board games.

['Backgammon', 'is', 'one', 'of', 'the', 'oldest', 'known', 'board', 'games', '.']

Its history can be traced back nearly 5,000 years to archeological discoveries in the Middle East.

['Its', 'history', 'can', 'be', 'traced', 'back', 'nearly', '5,000', 'years', 'to', 'archeological', 'discoveries', 'in', 'the', 'Middle', 'East', '.']

It is a two player game where each player has fifteen checkers which move between twenty-four points according to the roll of two dice.

['It', 'is', 'a', 'two', 'player', 'game', 'where', 'each', 'player', 'has', 'fifteen', 'checkers', 'which', 'move', 'between', 'twenty-four', 'points', 'according', 'to', 'the', 'roll', 'of', 'two', 'dice', '.']

Resultado (Português):

Gamão é um dos mais antigos jogos de tabuleiro conhecidos.

['Gamão', 'é', 'um', 'dos', 'mais', 'antigos', 'jogos', 'de', 'tabuleiro', 'conhecidos', '.']

Sua história remonta quase 5.000 anos às descobertas arqueológicas no Oriente Médio.

['Sua', 'história', 'remonta', 'quase', '5.000', 'anos', 'às', 'descobertas', 'arqueológicas', 'no', 'Oriente', 'Médio', '.']

É um jogo para dois jogadores, onde cada jogador tem quinze damas que se movem entre vinte e quatro pontos de acordo com o lançamento de dois dados.

['É', 'um', 'jogo', 'para', 'dois', 'jogadores', ',', 'onde', 'cada', 'jogador', 'tem', 'quinze', 'damas', 'que', 'se', 'movem', 'entre', 'vinte', 'e', 'quatro', 'pontos', 'de', 'acordo', 'com', 'o', 'lançamento', 'de', 'dois', 'dados', '.']

- A biblioteca NLTK do Python possui um robusto tokenizer de frases e um etiquetador de POS. O Python possui um tokenizador nativo, a função `.split ()`, na qual você pode passar um separador e ele dividirá a string em que a função é chamada nesse separador.
 - O tokenizer NLTK é mais robusto. Ele simboliza uma frase em palavras e pontuação
 - A lista de tags POS é a seguinte, com exemplos do que cada POS representa.
- | | | |
|--|---|---|
| <ul style="list-style-type: none">• CC coordinating conjunction• CD cardinal digit• DT determiner• EX existential there (like: “there is” ... think of it like “there exists”)• FW foreign word• IN preposition/subordinating conjunction• JJ adjective ‘big’• JJR adjective, comparative ‘bigger’• JJS adjective, superlative ‘biggest’• LS list marker 1)• MD modal could, will• NN noun, singular ‘desk’ | <ul style="list-style-type: none">• NNS noun plural ‘desks’• NNP proper noun, singular ‘Harrison’• NNPS proper noun, plural ‘Americans’• PDT predeterminer ‘all the kids’• POS possessive ending parent’s• PRP personal pronoun I, he, she• PRP\$ possessive pronoun my, his, hers• RB adverb very, silently,• RBR adverb, comparative better• RBS adverb, superlative best• RP particle give up• TO, to go ‘to’ the store.• UH interjection, errrrrrrm | <ul style="list-style-type: none">• VB verb, base form take• VBD verb, past tense took• VBG verb, gerund/present participle taking• VBN verb, past participle taken• VBP verb, sing. present, non-3d take• VBZ verb, 3rd person sing. present takes• WDT wh-determiner which• WP wh-pronoun who, what• WP\$ possessive wh-pronoun whose• WRB wh-adverb where, when |
|--|---|---|

Tokenização e marcação de partes do discurso (POS Tagger)

Backgammon is one of the oldest known board games.

Backgammon NNP

is VBZ

one CD

of IN

the DT

oldest JJS

known VBN

board NN

games NNS

Its history can be traced back nearly 5,000 years to archeological discoveries in the Middle East.

Its PRP\$

history NN

can MD

be VB

traced VBN

back RB

nearly RB

5,000 CD

years NNS

to TO

archeological JJ

discoveries NNS

in IN

the DT

Middle NNP

East NNP

It is a two player game where each player has fifteen checkers which move between twenty-four points according to the roll of two dice.

It PRP

twenty-four NN

is VBZ

points NNS

a DT

according VBG

two CD

to TO

player NN

the DT

game NN

roll NN

where WRB

of IN

each DT

two CD

player NN

dice NNS

has VBZ

fifteen VBN

checkers NNS

which WDT

move VBP

between IN

- Em textos da língua portuguesa ou inglesa temos diferentes palavras flexionadas em **gênero**, **número** ou **grau**, além de inúmeros tempos verbais distintos.
 - Trabalho Trabalhadora
 - Degrau Degraus
 - Amigo Amigão
 - am, are, is be
 - dog, dogs, dog's, dogs' dog
- A “normalização de palavras” pode ser entendida como a **redução ou a simplificação** de palavras.
- Duas técnicas mais importantes
 - *Stemming*
 - *Lemmatization*

- Por motivos gramaticais, os documentos podem conter diferentes formas de uma palavra, como gênero, modo e outros tipos de flexão.
- Além disso, às vezes temos palavras relacionadas com um significado semelhante, como nação, nacional, nacionalidade.
- O objetivo do stemming e da lematização é reduzir as formas flexionadas e, às vezes, derivadas de uma palavra para uma base comum.
- O stemming geralmente se refere a um processo heurístico bruto que corta o final das palavras na esperança de atingir esse objetivo corretamente na maioria das vezes, e geralmente inclui a remoção de afixos derivativos.
- A lematização geralmente se refere a fazer as coisas corretamente com o uso de uma análise de vocabulário e morfologia das palavras, normalmente com o objetivo de remover apenas terminações flexionadas e retornar a forma básica ou de dicionário de uma palavra, conhecida como lema.

- O ***processo de stemming*** consiste em reduzir a palavra a sua raiz (sem levar em conta a classe gramatical)
 - **amig** : amigo, amiga, amigao
 - **gat** : gato, gata, gatos, gatas
- ***Stemming*** geralmente refere-se a um processo de heurística que **corta as extremidades das palavras** inclui frequentemente a remoção de afixos derivacionais.
- Pode ser representado por um conjunto de regras que dependem da linguagem.

for
example**le**
compressed**ed**

and
compression**ion**
are**e**
both

accepted**ed**
as
equivalent**ent**
to

compress.



for
exempl
compress

and
compress
ar
both

accept
as
equival
to

compress

Sample text: Such an analysis can reveal features that are not easily visible from the variations in the individual genes and can lead to a picture of expression that is more biologically transparent and accessible to interpretation

Lovins stemmer: such an analysis can reveal features that are not easily visible from the variations in the individual genes and can lead to a picture of expression that is more biologically transparent and accessible to interpretation

Porter stemmer: such an analysis can reveal features that are not easily visible from the variations in the individual genes and can lead to a picture of expression that is more biologically transparent and accessible to interpretation

Paice stemmer: such an analysis can reveal features that are not easily visible from the variations in the individual genes and can lead to a picture of expression that is more biologically transparent and accessible to interpretation

- O ***processo de lemmatisation*** consiste aplicar uma técnicas para deflexionar as palavras (retira a conjugação verbal, caso seja um verbo, e altera os substantivos e os adjetivos para o singular masculino, de maneira a reduzir a palavra ate sua forma de dicionário)
 - **amigo** : amigo, amiga, amigao
 - **gato** : gato, gata, gatos, gatas
 - **ter** : tinha, tenho, tiver, tem
- ***Lemmatisation*** geralmente usa um dicionario de palavras (a heurística é mais sofisticada).

- Um vocabulário menor pode levar a um melhor desempenho na busca.
- As operações de stemming e lemmatization são operações comuns e iniciais utilizadas em sistemas de recuperação de informação:
 - Busca: Stemming?
 - Tradução: Lemmatisation?

Tokenização e marcação de partes do discurso (POS Tagger)

Backgammon is one of the oldest known board games.

Backgammon NNP

is VBZ

one CD

of IN

the DT

oldest JJS

known VBN

board NN

games NNS

Its history can be traced back nearly 5,000 years to archeological discoveries in the Middle East.

Its PRP\$

history NN

can MD

be VB

traced VBN

back RB

nearly RB

5,000 CD

years NNS

to TO

archeological JJ

discoveries NNS

in IN

the DT

Middle NNP

East NNP

It is a two player game where each player has fifteen checkers which move between twenty-four points according to the roll of two dice.

It PRP

twenty-four NN

is VBZ

points NNS

a DT

according VBG

two CD

to TO

player NN

the DT

game NN

roll NN

where WRB

of IN

each DT

two CD

player NN

dice NNS

has VBZ

fifteen VBN

checkers NNS

which WDT

move VBP

between IN

Backgammon is one of the oldest known board games.

['Backgammon', 'is', 'one', 'of', 'the', 'oldest', 'known', 'board', 'games', '.']

['NNP', 'VBZ', 'CD', 'IN', 'DT', 'JJ', 'VBN', 'NN', 'NNS', '.']

Its history can be traced back nearly 5,000 years to archeological discoveries in the Middle East.

['Its', 'history', 'can', 'be', 'traced', 'back', 'nearly', '5,000', 'years', 'to', 'archeological', 'discoveries', 'in', 'the', 'Middle', 'East', '.']

['PRP\$', 'NN', 'MD', 'VB', 'VBN', 'RB', 'RB', 'CD', 'NNS', 'TO', 'JJ', 'NNS', 'IN', 'DT', 'NNP', 'NNP', '.']

It is a two player game where each player has fifteen checkers which move between twenty-four points according to the roll of two dice.

['It', 'is', 'a', 'two', 'player', 'game', 'where', 'each', 'player', 'has', 'fifteen', 'checkers', 'which', 'move', 'between', 'twenty-four', 'points', 'according', 'to', 'the', 'roll', 'of', 'two', 'dice', '.']

['PRP', 'VBZ', 'DT', 'CD', 'NN', 'NN', 'WRB', 'DT', 'NN', 'VBZ', 'VBN', 'NNS', 'WDT', 'VBP', 'IN', 'NN', 'NNS', 'VBG', 'TO', 'DT', 'NN', 'IN', 'CD', 'NNS', '.']

Comparando Stemming com Lematização

Verbos

Backgammon is one of the oldest known board games.

Stemmer: backgammon

Lemmatizer: Backgammon

Stemmer: is

Lemmatizer: be

Stemmer: known

Lemmatizer: know

Substantivos

Backgammon is one of the oldest known board games.

Stemmer: backgammon

Lemmatizer: Backgammon

Adjetivos

Backgammon is one of the oldest known board games.

Stemmer: oldest

Lemmatizer: old

Adverbios

Backgammon is one of the oldest known board games.

Comparando Stemming com Lematização

Its history can be traced back nearly 5,000 years to archeological discoveries in the Middle East.

Stemmer: it

Lemmatizer: Its

Stemmer: histori

Lemmatizer: history

Stemmer: can

Lemmatizer: can

Stemmer: be

Lemmatizer: be

Stemmer: trace

Lemmatizer: traced

Stemmer: back

Lemmatizer: back

Stemmer: nearli

Lemmatizer: nearly

Its history can be traced back nearly 5,000 years to archeological discoveries in the Middle East.

Stemmer: 5,000

Lemmatizer: 5,000

Stemmer: year

Lemmatizer: years

Stemmer: to

Lemmatizer: to

Stemmer: archeolog

Lemmatizer: archeological

Stemmer: discoveri

Lemmatizer: discoveries

Stemmer: in

Lemmatizer: in

Stemmer: the

Lemmatizer: the

Its history can be traced back nearly 5,000 years to archeological discoveries in the Middle East.

Stemmer: middl

Lemmatizer: Middle

Stemmer: east

Lemmatizer: East

- Stopwords são aquelas que são filtradas antes ou após o processamento do texto. Ao aplicar o aprendizado de máquina ao texto, essas palavras podem adicionar muito ruído. É por isso que queremos remover essas palavras irrelevantes.
- As palavras de parada geralmente se referem às palavras mais comuns, como "e", "the", "a" em um inglês, mas não existe uma lista universal única de stopwords. A lista das palavras de parada pode mudar dependendo do seu aplicativo.



- A ferramenta NLTK possui uma lista predefinida de stopwords que se refere às palavras mais comuns. Se você o usar pela primeira vez, precisará fazer o download das palavras de parada usando este código: `nltk.download("stopwords")`.
- Depois de concluir o download, podemos carregar o pacote stopwords no `nltk.corpus` e usá-lo para carregar as palavras stop.



['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"]

['de', 'a', 'o', 'que', 'e', 'do', 'da', 'em', 'um', 'para', 'com', 'não', 'uma', 'os', 'no', 'se', 'na', 'por', 'mais', 'as', 'dos', 'como', 'mas', 'ao', 'ele', 'das', 'à', 'seu', 'sua', 'ou', 'quando', 'muito', 'nos', 'já', 'eu', 'também', 'só', 'pelo', 'pela', 'até', 'isso', 'ela', 'entre', 'depois', 'sem', 'mesmo', 'aos', 'seus', 'quem', 'nas', 'me', 'esse', 'eles', 'você', 'essa', 'num', 'nem', 'suas', 'meu', 'às', 'minha', 'numa', 'pelos', 'elas', 'qual', 'nós', 'lhe', 'deles', 'essas', 'esses', 'pelas', 'este', 'dele', 'tu', 'te', 'vocês', 'vos', 'lhes', 'meus', 'minhas', 'teu', 'tua', 'teus', 'tuas', 'nosso', 'nossa', 'nossos', 'nossas', 'dela', 'delas', 'esta', 'estes', 'estas', 'aquele', 'aquela', 'aqueles', 'aquelas', 'isto', 'aquilo', 'estou', 'está', 'estamos', 'estão', 'estive', 'esteve', 'estivemos', 'estiveram', 'estava', 'estávamos', 'estavam', 'estivera', 'estivéramos', 'esteja', 'estejamos', 'estejam', 'estivesse', 'estivéssemos', 'estivessem', 'estiver', 'estivermos', 'estiverem', 'hei', 'há', 'havemos', 'hão', 'houve', 'houvemos', 'houveram', 'houvera', 'houvéramos', 'haja', 'hajamos', 'hajam', 'houvesse', 'houvéssemos', 'houvessem', 'houver', 'houvermos', 'houverem', 'houverei', 'houverá', 'houveremos', 'houverão', 'houveria', 'houveríamos', 'houveriam', 'sou', 'somos', 'são', 'era', 'éramos', 'eram', 'fui', 'foi', 'fomos', 'foram', 'fora', 'fôramos', 'seja', 'sejamos', 'sejam', 'fosse', 'fôssemos', 'fossem', 'for', 'formos', 'forem', 'serei', 'será', 'seremos', 'serão', 'seria', 'seríamos', 'seriam', 'tenho', 'tem', 'temos', 'tém', 'tinha', 'tínhamos', 'tinham', 'tive', 'teve', 'tivemos', 'tiveram', 'tivera', 'tivéramos', 'tenha', 'tenhamos', 'tenham', 'tivesse', 'tivéssemos', 'tivessem', 'tiver', 'tivermos', 'tiverem', 'terei', 'terá', 'teremos', 'terão', 'teria', 'teríamos', 'teriam']

Backgammon is one of the oldest known board games.

Sentença:

Backgammon is one of the oldest known board games.

Palavras:

['Backgammon', 'is', 'one', 'of', 'the', 'oldest', 'known', 'board', 'games', '.']

Palavras – stopwords:

['Backgammon', 'one', 'oldest', 'known', 'board', 'games', '.']

Its history can be traced back nearly 5,000 years to archeological discoveries in the Middle East.

Sentença:

Its history can be traced back nearly 5,000 years to archeological discoveries in the Middle East.

Palavras:

['Its', 'history', 'can', 'be', 'traced', 'back', 'nearly', '5,000', 'years', 'to', 'archeological', 'discoveries', 'in', 'the', 'Middle', 'East', '.']

Palavras – stopwords:

['Its', 'history', 'traced', 'back', 'nearly', '5,000', 'years', 'archeological', 'discoveries', 'Middle', 'East', '.']

It is a two player game where each player has fifteen checkers which move between twenty-four points according to the roll of two dice.

Sentença:

It is a two player game where each player has fifteen checkers which move between twenty-four points according to the roll of two dice.

Palavras:

['It', 'is', 'a', 'two', 'player', 'game', 'where', 'each', 'player', 'has', 'fifteen', 'checkers', 'which', 'move', 'between', 'twenty-four', 'points', 'according', 'to', 'the', 'roll', 'of', 'two', 'dice', '.']

Palavras – stopwords:

['It', 'two', 'player', 'game', 'player', 'fifteen', 'checkers', 'move', 'twenty-four', 'points', 'according', 'roll', 'two', 'dice', '.']

Backgammon is one of the oldest known board games.

Sentença:

Gamão é um dos mais antigos jogos de tabuleiro conhecidos.

Palavras:

['Gamão', 'é', 'um', 'dos', 'mais', 'antigos', 'jogos', 'de', 'tabuleiro', 'conhecidos', '.']

Palavras – stopwords:

['Gamão', 'é', 'antigos', 'jogos', 'tabuleiro', 'conhecidos', '.']

Its history can be traced back nearly 5,000 years to archeological discoveries in the Middle East.

Sentença:

Its history can be traced back nearly 5,000 years to archeological discoveries in the Middle East.

Palavras:

['Its', 'history', 'can', 'be', 'traced', 'back', 'nearly', '5,000', 'years', 'to', 'archeological', 'discoveries', 'in', 'the', 'Middle', 'East', '.']

Palavras – stopwords:

['Its', 'history', 'traced', 'back', 'nearly', '5,000', 'years', 'archeological', 'discoveries', 'Middle', 'East', '.']

It is a two player game where each player has fifteen checkers which move between twenty-four points according to the roll of two dice.

Sentença:

It is a two player game where each player has fifteen checkers which move between twenty-four points according to the roll of two dice.

Palavras:

['It', 'is', 'a', 'two', 'player', 'game', 'where', 'each', 'player', 'has', 'fifteen', 'checkers', 'which', 'move', 'between', 'twenty-four', 'points', 'according', 'to', 'the', 'roll', 'of', 'two', 'dice', '.']

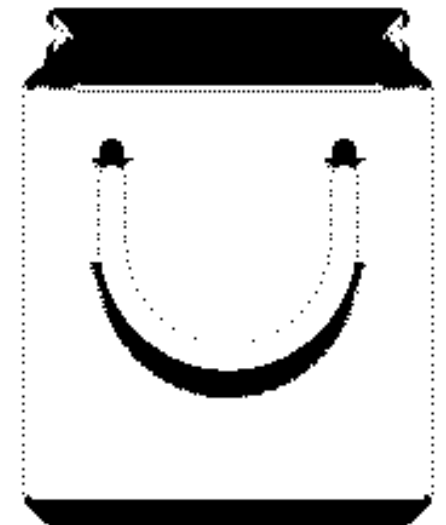
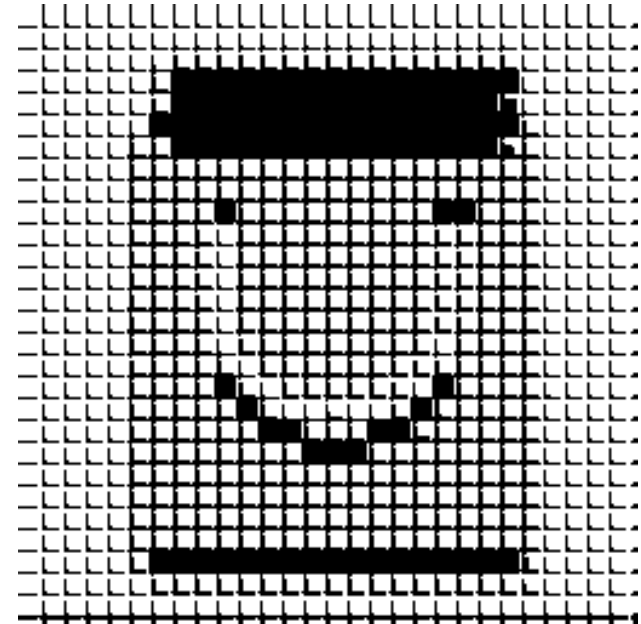
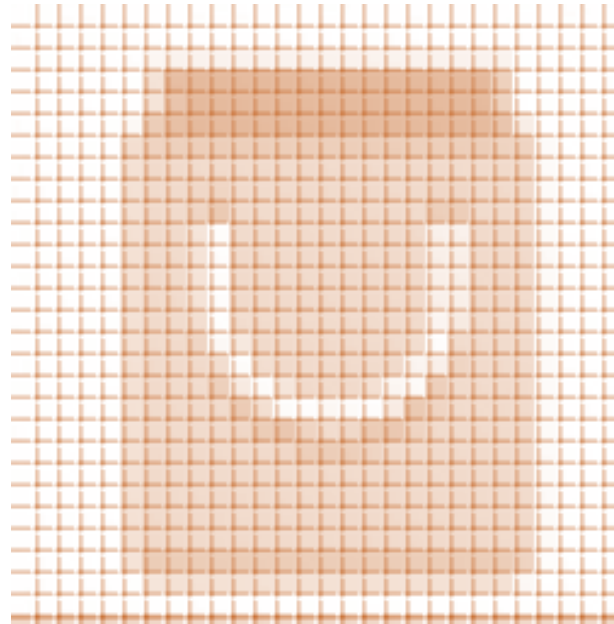
Palavras – stopwords:

['It', 'two', 'player', 'game', 'player', 'fifteen', 'checkers', 'move', 'twenty-four', 'points', 'according', 'roll', 'two', 'dice', '.']

- Os algoritmos de aprendizado de máquina não podem trabalhar diretamente com o texto bruto, precisamos converter o texto em vetores de números. Isso é chamado de extração de atributos.
- O modelo de Bag-of-Words é uma técnica popular e simples de extração de recursos usada quando trabalhamos com texto. Ele descreve a ocorrência de cada palavra em um documento.



- Os algoritmos de aprendizado de máquina não podem trabalhar diretamente com o texto bruto, precisamos converter o texto em vetores de números. Isso é chamado de extração de recurso.
- O modelo de Bag-of-Words é uma técnica popular e simples de extração de atributos usada quando trabalhamos com texto. Ele descreve a ocorrência de cada palavra em um documento.



- Para usar este modelo, precisamos:
 - Crie um vocabulário de palavras conhecidas (também chamadas de tokens)
 - Escolha uma medida da frequência de palavras conhecidas
- Qualquer informação sobre a ordem ou estrutura das palavras é descartada.
- Por isso é chamado de Bag-of Words.
- Este modelo está tentando entender se uma palavra conhecida ocorre em um documento, mas não sabe onde está essa palavra no documento.
- A intuição é que documentos semelhantes tenham conteúdos semelhantes. Além disso, a partir de um conteúdo, podemos aprender algo sobre o significado do documento.

- Vamos ver quais são as etapas para criar um modelo de Bag-of-Words. Neste exemplo, usaremos apenas quatro frases para ver como esse modelo funciona. Nos problemas do mundo real, você trabalha com quantidades muito maiores de dados.

- Digamos que esses são nossos dados e queremos carregá-los como uma matriz.

I like this movie, it's funny.

I hate this movie.

This was awesome! I like it.

Nice one. I love it.

Eu gosto deste filme, é engraçado.

Eu odeio esse filme.

Isso foi demais! Eu gosto disso.

Agradável. Eu amo isso.



- Para conseguir isso, podemos simplesmente ler o arquivo e dividi-lo por linhas.

```
["I like this movie, it's funny.I hate this  
movie.", 'This was awesome!', 'I like it.',  
'Nice one.', 'I love it.']
```

```
['Eu gosto deste filme, é engraçado.', 'Eu  
odeio esse filme.', 'Isso foi incrível!', 'Eu  
gosto.', 'Legal.', 'Eu amo.']
```

```
text2 = "I like this movie, it's funny.I hate  
this movie. This was awesome! I like it.  
Nice one. I love it."
```

```
texto2 = "Eu gosto deste filme, é  
engraçado. Eu odeio esse filme. Isso foi  
incrível! Eu gosto. Legal. Eu amo."
```

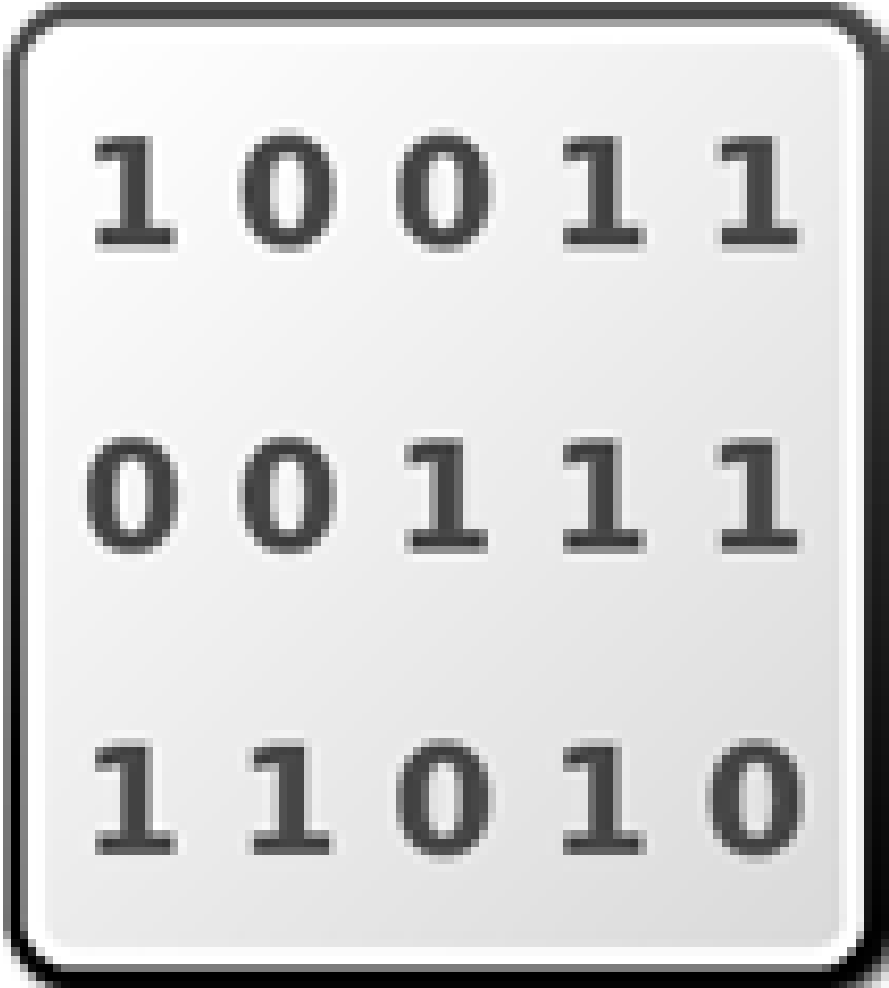
```
documents = nltk.sent_tokenize(text2)  
documentos = nltk.sent_tokenize(texto2)  
print(documents)  
print(documentos)
```

- Vamos obter todas as palavras exclusivas das quatro frases carregadas, ignorando o caso, a pontuação e os tokens de um caractere.
- Essas palavras serão o nosso vocabulário (palavras conhecidas).
- Podemos usar a classe `CountVectorizer` da biblioteca `sklearn` para criar nosso vocabulário.
- Veremos como podemos usá-lo depois de ler a próxima etapa também.



Criar os vetores de documento

- Em seguida, precisamos marcar as palavras em cada documento.
- A tarefa aqui é converter cada texto bruto em um vetor de números.
- Depois disso, podemos usar esses vetores como entrada para um modelo de aprendizado de máquina.
- O método de pontuação mais simples é marcar a presença de palavras com 1 para presente e 0 para ausência.
- Agora, vamos ver como podemos criar um modelo de palavras usando a classe CountVectorizer mencionada acima.



1	0	0	1	1
0	0	1	1	1
1	1	0	1	0

Criação do Modelo Bag-of-Words Inglês

```
# Importe as bibliotecas
from sklearn.feature_extraction.text import CountVectorizer
import pandas as pd

# Passo 2. Crie o vocabulario
# O padrão de token remove os tokens de um único caractere. É por isso que não temos os tokens "l" e "s" no modelo de
saída usando a classe CountVectorizer mencionada acima.
count_vectorizer = CountVectorizer()

# Passo 3. Criar o modelo Bag-of-Words Model
bag_of_words = count_vectorizer.fit_transform(documents)

# Mostre o Modelo Bag-of-Words como um DataFrame Pandas (Tabela)
feature_names = count_vectorizer.get_feature_names()
pd.DataFrame(bag_of_words.toarray(), columns = feature_names)
```

	awesome	funny	hate	it	like	love	movie	nice	one	this	was
0	0	1	1	1	1	0	2	0	0	2	0
1	1	0	0	0	0	0	0	0	0	1	1
2	0	0	0	1	1	0	0	0	0	0	0
3	0	0	0	0	0	0	0	1	1	0	0
4	0	0	0	1	0	1	0	0	0	0	0

Criação do Modelo Bag-of-Words Português

```
# Importe as bibliotecas
from sklearn.feature_extraction.text import CountVectorizer
import pandas as pd

# Passo 2. Crie o vocabulário
# O padrão de token remove os tokens de um único caractere. É por isso que não temos os tokens "l" e "s" no modelo de
# saída usando a classe CountVectorizer mencionada acima.
count_vectorizer = CountVectorizer()

# Passo 3. Criar o modelo Bag-of-Words Model
bag_of_words2 = count_vectorizer.fit_transform(documentos)

# Mostre o Modelo Bag-of-Words como um DataFrame Pandas (Tabela)
nomes_atributos = count_vectorizer.get_feature_names()
pd.DataFrame(bag_of_words2.toarray(), columns = nomes_atributos)

In [ ]:
```

	amo	deste	engraçado	esse	eu	filme	foi	gosto	incrível	isso	legal	odeio
0	0	1	1	0	1	1	0	1	0	0	0	0
1	0	0	0	1	1	1	0	0	0	0	0	1
2	0	0	0	0	0	0	1	0	1	1	0	0
3	0	0	0	0	1	0	0	1	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	1	0
5	1	0	0	0	1	0	0	0	0	0	0	0

- A complexidade do modelo de BoW depende da decisão de como criar o vocabulário de palavras conhecidas (tokens) e como classificar a presença de palavras conhecidas.
- Quando o tamanho do vocabulário aumenta, a representação vetorial dos documentos também aumenta. No exemplo acima, o comprimento do vetor do documento é igual ao número de palavras conhecidas.
- Em alguns casos, podemos ter uma quantidade enorme de dados e, nesses casos, o comprimento do vetor que representa um documento pode ser milhares ou milhões de elementos. Além disso, cada documento pode conter apenas algumas das palavras conhecidas no vocabulário.
- Portanto, as representações vetoriais terão muitos zeros. Esses vetores que possuem muitos zeros são chamados de vetores esparsos. Eles exigem mais memória e recursos computacionais.

- Podemos diminuir o número de palavras conhecidas ao usar um modelo de saco de palavras para diminuir a memória e os recursos computacionais necessários.
- Podemos usar as técnicas de limpeza de texto antes de criarmos nosso modelo BoW:
 - Ignorando o case (MAIÚSCULAS ou minúsculas) das palavras
 - Ignorando pontuação
 - Removendo as palavras de parada dos nossos documentos
 - Reduzindo as palavras à sua forma base (Lematização e Stemming de Texto)
 - Corrigindo palavras com erros ortográficos

- Outra maneira mais complexa de criar um vocabulário é usar palavras agrupadas.
- Isso altera o escopo do vocabulário e permite que o modelo de saco de palavras obtenha mais detalhes sobre o documento.
- Essa abordagem é chamada n-gramas.
- Um n-grama é uma sequência de vários itens (palavras, letra, números, dígitos, etc.).
- No contexto dos corpora de texto, n-gramas geralmente se referem a uma sequência de palavras.
- Um unigrama é uma palavra, um bigram é uma sequência de duas palavras, um trigram é uma sequência de três palavras, etc.
- O "n" no "n-grama" refere-se ao número de palavras agrupadas. Apenas os n-gramas que aparecem no corpus são modelados, nem todos os n-gramas possíveis.

- Outra maneira mais complexa de criar um vocabulário é usar palavras agrupadas.
- Isso altera o escopo do vocabulário e permite que o modelo de saco de palavras obtenha mais detalhes sobre o documento.
- Essa abordagem é chamada n-gramas.
- Um n-grama é uma sequência de vários itens (palavras, letra, números, dígitos, etc.).
- No contexto dos corpora de texto, n-gramas geralmente se referem a uma sequência de palavras.
- Um unigrama é uma palavra, um bigram é uma sequência de duas palavras, um trigrama é uma sequência de três palavras, etc.
- O "n" no "n-grama" refere-se ao número de palavras agrupadas. Apenas os n-gramas que aparecem no corpus são modelados, nem todos os n-gramas possíveis.
- O Bag of Bigrams é mais poderoso do que a abordagem BoW.

- Vamos criar uma função e gerar os ngrams que julgarmos necessários.

```
from nltk.tokenize import word_tokenize
from nltk.util import ngrams
def get_ngrams(text, n ):
    n_grams = ngrams(word_tokenize(text), n)
    return [ ' '.join(grams) for grams in n_grams]
for doc in documents:
    print(get_ngrams(doc, 2))
for doc in documentos:
    print(get_ngrams(doc, 2))
```

```
['I like', 'like this', 'this movie', 'movie ,', ', it', "it 's", "'s funny.I", 'funny.I hate', 'hate this', 'this movie', 'movie .']
['This was', 'was awesome', 'awesome !']
['I like', 'like it', 'it .']
['Nice one', 'one .']
['I love', 'love it', 'it .']
```

```
['Eu gosto', 'gosto deste', 'deste filme', 'filme ,', ', é', 'é engraçado', 'engraçado .']
['Eu odeio', 'odeio esse', 'esse filme', 'filme .']
['Isso foi', 'foi incrível', 'incrível !']
['Eu gosto', 'gosto .']
['Legal .']
['Eu amo', 'amo .']
```

- Vamos criar uma função e gerar os ngrams que julgarmos necessários.

```
from nltk.tokenize import word_tokenize
from nltk.util import ngrams
def get_ngrams(text, n ):
    n_grams = ngrams(word_tokenize(text), n)
    return [ ' '.join(grams) for grams in n_grams]
for doc in documents:
    print(get_ngrams(doc, 3))
for doc in documentos:
    print(get_ngrams(doc, 3))
```

['I like this', 'like this movie', 'this movie ,', 'movie , it', ', it 's', "it 's funny.I", "'s funny.I hate", 'funny.I hate this', 'hate this movie', 'this movie .']

['This was awesome', 'was awesome !']

['I like it', 'like it .']

['Nice one .']

['I love it', 'love it .']

['Eu gosto deste', 'gosto deste filme', 'deste filme ,', 'filme , é', ', é engraçado', 'é engraçado .']

['Eu odeio esse', 'odeio esse filme', 'esse filme .']

['Isso foi incrível', 'foi incrível !']

['Eu gosto .']

[]

['Eu amo .']

- Vamos criar uma função e gerar os ngrams que julgarmos necessários.

```
from nltk.tokenize import word_tokenize
from nltk.util import ngrams
def get_ngrams(text, n ):
    n_grams = ngrams(word_tokenize(text), n)
    return [ ' '.join(grams) for grams in n_grams]
for doc in documents:
    print(get_ngrams(doc, 3))
for doc in documentos:
    print(get_ngrams(doc, 3))
```

```
['I like this movie', 'like this movie ,', 'this movie , it', 'movie , it 's', ', it 's funny.I', 'it 's funny.I hate', "'s funny.I hate this", 'funny.I hate this movie', 'hate this movie .']
```

```
['This was awesome !']
```

```
['I like it .']
```

```
[]
```

```
['I love it .']
```

```
['Eu gosto deste filme', 'gosto deste filme ,', 'deste filme , é', 'filme , é engraçado', ', é engraçado .']
```

```
['Eu odeio esse filme', 'odeio esse filme .']
```

```
['Isso foi incrível !']
```

```
[]
```

```
[]
```

```
[]
```

- Uma vez que criamos nosso vocabulário de palavras conhecidas, precisamos pontuar a ocorrência das palavras em nossos dados. Vimos uma abordagem muito simples - a abordagem binária (1 para presença, 0 para ausência).
- Alguns métodos de pontuação adicionais são:
 - Contagem. Conte o número de vezes que cada palavra aparece em um documento.
 - Frequências. Calcule a frequência em que cada palavra aparece no documento com todas as palavras do documento. No corpus são modelados, nem todos os n-gramas possíveis.

- Um problema com a frequência das palavras de pontuação é que as palavras mais frequentes no documento começam a ter as pontuações mais altas.
- Essas palavras frequentes podem não conter tanto "ganho informacional" para o modelo em comparação com algumas palavras mais raras e específicas de domínio.
- Uma abordagem para corrigir esse problema é penalizar as palavras que são frequentes em todos os documentos.
- Essa abordagem é chamada de TF-IDF.

O que é a Matriz TF-IDF

- TF-IDF, abreviação de termo frequência inversa à frequência de documentos é uma medida estatística usada para avaliar a importância de uma palavra para um documento em uma coleção ou corpus.
- O valor da pontuação TF-IDF aumenta proporcionalmente ao número de vezes que uma palavra aparece no documento, mas é compensado pelo número de documentos no corpus que contêm a palavra.
- Vamos ver a fórmula usada para calcular uma pontuação TF-IDF para um determinado termo x dentro de um documento y .

$$w_{x,y} = \text{tf}_{x,y} \times \log \left(\frac{N}{\text{df}_x} \right)$$

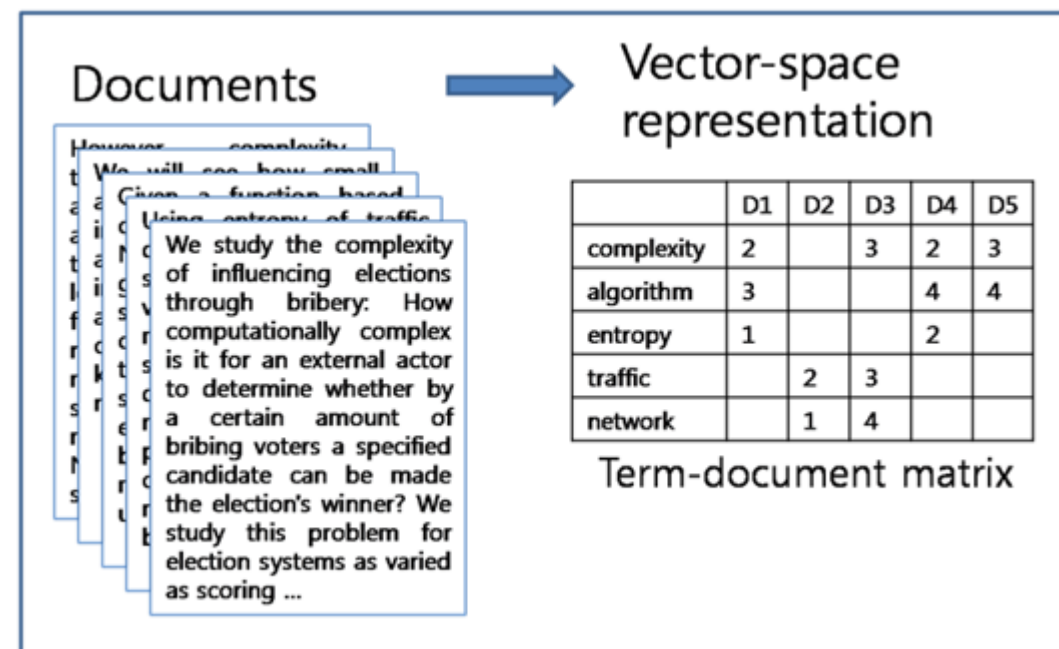
TF-IDF

Term x within document y

$\text{tf}_{x,y}$ = frequency of x in y

df_x = number of documents containing x

N = total number of documents



- Agora, vamos dividir um pouco essa fórmula e ver como as diferentes partes da fórmula funcionam.
- Frequência do termo (TF): uma pontuação da frequência da palavra no documento atual.
- $TF(\text{termo}) = \frac{\text{Numero de Vezes que o termo aparece no documento}}{\text{Numero total de termos em um documento}}$
- Inverse Term Frequency (ITF): uma pontuação de quão rara a palavra é nos documentos.
- $IDF(\text{termo}) = \log \left(\frac{\text{Numero Total de documentos}}{\text{Numero total de documentos com o termo}} \right)$
- Finalmente, podemos usar as fórmulas anteriores para calcular a pontuação do TF-IDF para um determinado termo como este:
- $TFIDF(\text{termo}) = TF(\text{termo}) * IDF(\text{termo})$
- Ou numa Fórmula Geral TF-IDF
- $TFIDF(\text{termo}) = \left(\frac{\text{Numero de Vezes que o termo aparece no documento}}{\text{Numero total de termos em um documento}} \right) \times \log \left(\frac{\text{Numero Total de documentos}}{\text{Numero total de documentos com o termo}} \right)$

TDIDF Inglês

	awesome	funny	hate	it	like	love	movie	nice	one	this	was
I like this movie, it's funny.I hate this movie.	0.000000	0.321029	0.321029	0.214997	0.259005	0.000000	0.642059	0.000000	0.000000	0.518009	0.000000
This was awesome!	0.614189	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.495524	0.614189
I like it.	0.000000	0.000000	0.000000	0.638711	0.769447	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Nice one.	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.707107	0.707107	0.000000	0.000000
I love it.	0.000000	0.000000	0.000000	0.556451	0.000000	0.830881	0.000000	0.000000	0.000000	0.000000	0.000000

TDIDF Português

	amo	deste	engraçado	esse	eu	filme	foi	gosto	incrível	isso	legal	odeio
Eu gosto deste filme, é engraçado.	0.00000	0.5201	0.5201	0.000000	0.308554	0.426489	0.00000	0.426489	0.00000	0.00000	0.0	0.000000
Eu odeio esse filme.	0.00000	0.0000	0.0000	0.575019	0.341135	0.471523	0.00000	0.000000	0.00000	0.00000	0.0	0.575019
Isso foi incrível!	0.00000	0.0000	0.0000	0.000000	0.000000	0.000000	0.57735	0.000000	0.57735	0.57735	0.0	0.000000
Eu gosto.	0.00000	0.0000	0.0000	0.000000	0.586157	0.000000	0.00000	0.810198	0.00000	0.00000	0.0	0.000000
Legal.	0.00000	0.0000	0.0000	0.000000	0.000000	0.000000	0.00000	0.000000	0.00000	0.00000	1.0	0.000000
Eu amo.	0.86004	0.0000	0.0000	0.000000	0.510227	0.000000	0.00000	0.000000	0.00000	0.00000	0.0	0.000000

- Nesta aula você aprendeu o básico da PNL para texto. Mais especificamente, você aprendeu o seguinte:
 - A PNL é usada para aplicar algoritmos de aprendizado de máquina ao texto e à fala.
 - A tokenização de sentenças é o problema de dividir uma sequência de linguagem escrita em suas frases componentes
 - A tokenização de palavras é o problema de dividir uma sequência de linguagem escrita em suas palavras componentes
 - O objetivo da stemmização e da lematização é reduzir as formas flexionadas de uma palavra para uma base comum.
 - Stopwords são aquelas que são filtradas antes ou após o processamento do texto. Eles geralmente se referem às palavras mais comuns em um idioma.
 - O modelo de Bag-of-Words é uma técnica popular e simples de extração de recursos usada quando trabalhamos com texto. Ele descreve a ocorrência de cada palavra em um documento.
 - TF-IDF é uma medida estatística usada para avaliar a importância de uma palavra para um documento em uma coleção ou corpus.

QUIZZ # 10

- <https://forms.gle/eJFCGZqs4a5xsLCC6>

