

# Laboratório N° 2: Text Mining - Técnicas de Pré-processamento

---

Extração Automática de Informação  
2021/2022

Prof. Joaquim Filipe  
Eng. Filipe Mariano

## Objetivos

- Identificação das principais técnicas de pré-processamento utilizadas no Text Mining
- Implementação de vários passos de pré-processamento de modo a limpar o conteúdo dos documentos

## 1. Pré-processamento

O pré-processamento é uma etapa crítica no processo de *Text Mining* e extração de informação, tendo como principal objetivo extrair conhecimento dos dados não estruturados. Nesta fase, os caracteres, palavras ou conjunto de palavras identificadas são pré-processados porque, normalmente, no texto existem formatos especiais, como datas ou numéricos, assim como as palavras mais comuns de ocorrerem em todos os documentos, que dificilmente ajudam o sistema.

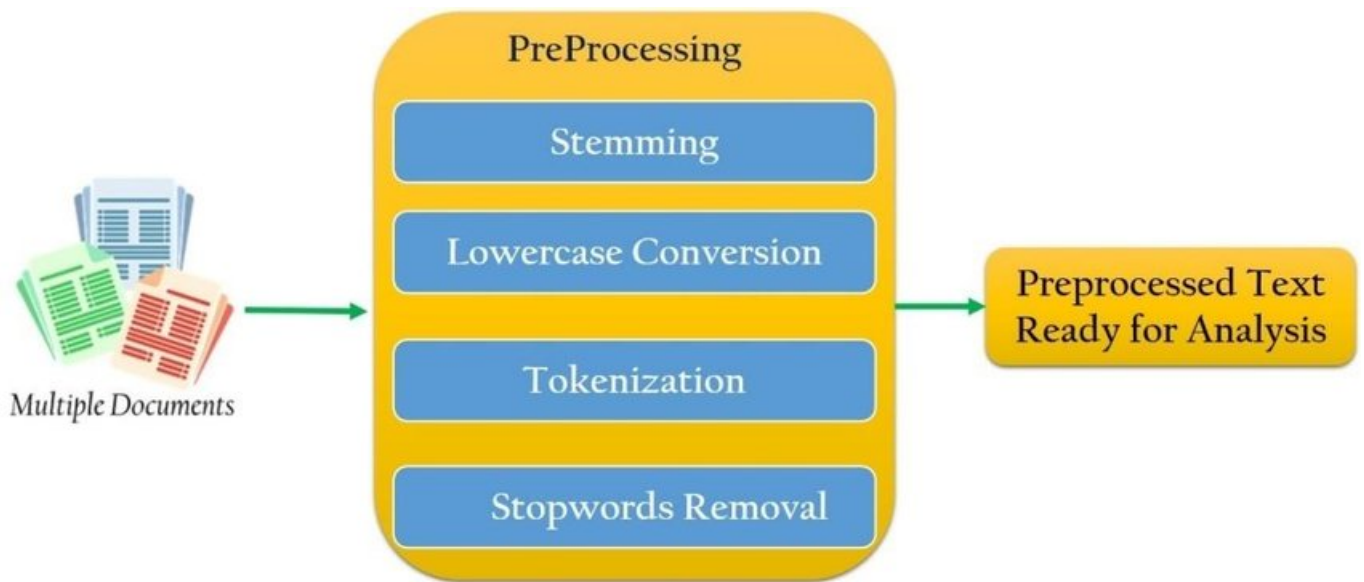
### ***Porque é que é necessário realizar este pré-processamento?***

Este pré-processamento permite reduzir os dados a extrair dos documentos, uma vez que:

- Elementos da língua como pronomes, artigos, preposições, etc, contabilizam cerca de 20-30% do total de palavras existente num documento (**stopwords**).
- Técnicas como o **stemming** permitem reduzir o número de palavras existente nos documentos.

### ***Como é que estas técnicas melhoram a eficiência e eficácia de um sistema de classificação?***

- Remoção das **stopwords** para não confundir com termos que poderão ser considerados importantes.
- **Stemming** para que as palavras com o mesmo radical façam *match*, aumentando o enquadramento das palavras similares nos diversos documentos.



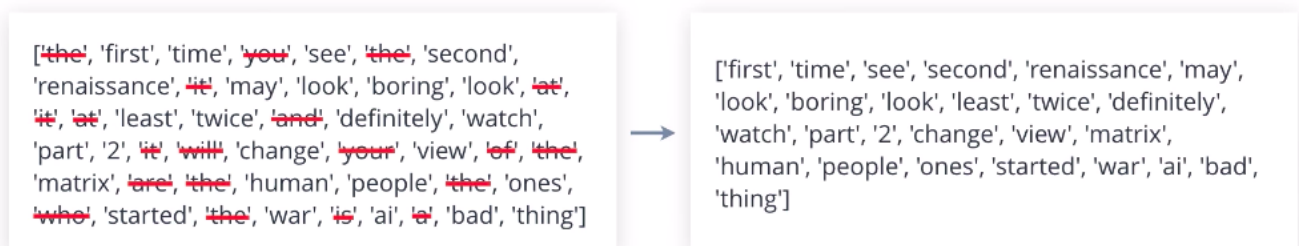
**Figura 1:** Fase de pré-processamento no processo de *text mining*.

## 2. Lower Case

Um dos passos que se costuma fazer no pré-processamento é a transformação de todos os termos existentes no texto, em minúsculas. É um passo importante porque evita que exista múltiplas cópias da mesma palavra ou termo. Um exemplo disso seria na contabilização da palavra **Success** ou **success** que seriam contabilizadas como palavras distintas.

## 3. Stopwords

As *stopwords* são palavras específicas do idioma que não possuem qualquer conteúdo informacional para o processo de classificação, tais como pronomes, preposições, conjunções, etc. Palavras em inglês como "the", "of", "and", "to" são irrelevantes do ponto de vista da análise uma vez que são palavras que ocorrem em todos os documentos, diversas vezes.



**Figura 2:** Remoção de *stopwords*.

## 4. Tokenization

A técnica de *tokenization* é o processo de dividir um texto em palavras, frases, símbolos ou outros elementos significativos, denominados de **token**. O objetivo desta técnica passa por explorar as palavras existentes num texto, criando um conjunto de *tokens* que servirão de *input* numa fase posterior do sistema. O processo de *tokenization* envolve também um passo prévio, que é o da remoção de pontuação (por exemplo, através da utilização de *Regular Expressions*). A principal razão de utilização de *tokenization* é a de identificar as palavras-chave com maior valor significativo.

## 4.1. Regular Expressions

As *regular expressions*, também conhecidas como *regex*, proporcionam uma forma concisa e flexível de identificar cadeias de caracteres, caracteres, palavras ou padrões de caracteres. Na fase de pré-processamento num sistema de *Text Mining*, a sua utilização prende-se principalmente com o objetivo de limpeza de caracteres sem conteúdo informacional para análise, como pontuação e numeração.

Em Javascript, as *regex* podem ser utilizadas a partir da classe `RegExp` (ver documentação) ou da função `match` existente no prototype da classe `String` (ver documentação).

## 4.2. N-Gram

Por vezes, os *tokens* são conhecidos como n-gramas, que correspondem às combinações de **N** palavras "juntas" que podem ser criadas a partir de um texto.

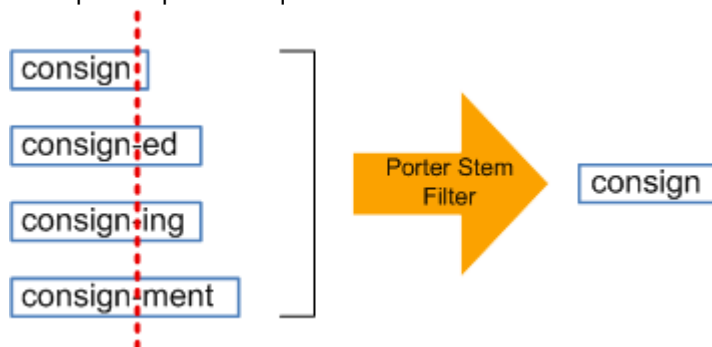
**Exemplo:**

```
"alpha bravo charlie"
// 1-Gram (unigrama de palavras): [['alpha'], ['bravo'], ['charlie']]
// 2-Gram (bigrama de palavras): [['alpha', 'bravo'], ['bravo', 'charlie']]
```

## 5. Stemming

*Stemming* é o nome dado à técnica de processamento de texto onde todas as palavras encontradas são reduzidas ao seu radical (*stem*), ou seja, à sua raíz ou forma base, removendo quaisquer sufixos da mesma. Este processo é bastante importante, visto que palavras cuja base seja igual geralmente têm significados semelhantes ou representam conceitos relativamente próximos, pelo que as palavras podem ser reduzidas ao seu radical. Um dos algoritmos mais conhecidos e amplamente utilizado neste contexto é o **Porter Stemming**.

Exemplo de palavras que têm um tronco em comum:



**Figura 3:** Exemplificação da aplicação do algoritmo de *Porter Stemming*.

## 6. Exercícios

1. Criar uma tabela `trainingSet` que terá o seu próprio id e o id da tabela `corpus`, de modo a identificar quais os documentos que pertencerão ao conjunto de treino.
2. Importar para a nova tabela criada os 50 primeiros documentos de cada classe (*happy* e *not happy*). No total serão utilizados 100 textos no conjunto de treino, metade de cada classe. Pode fazê-lo diretamente

através do MySQL.

3. Crie uma classe `train.js` na raiz do projeto. com uma função interna `getTrainingSet` que devolva todos os textos inseridos na nova tabela e que deverão ser alvo do pré-processamento.

4. Criar uma diretoria `preprocessing` e criar um módulo `stopwords.js`.

a. Instalar através do npm o módulo `stopword` ([ver documentação](#)).

b. Exportar uma função que recebe como parâmetro de entrada o texto e devolve o texto limpo, removendo as *stopwords* através do módulo instalado.

5. Na diretoria `preprocessing` criar um novo módulo `clean.js`.

a. Faça uma função que recebe um texto e coloca o texto em minúsculas.

b. Faça uma função que recebe um texto e faz a limpeza de espaços em branco no início, no fim e tudo o que seja mais do que um espaço em branco deverá ser convertido em apenas um espaço.

c. Faça uma função que recebe um texto e remove do texto tudo o que seja caracteres numéricos ou de pontuação.

d. Exporte uma função que recebe um texto e aplique todas as funções de limpeza descritas anteriormente, que principalmente o que irão fazer é através de *regex* eliminar pontuação, números e espaços a mais, retornando no final o texto limpo.

6. Na diretoria `preprocessing` criar um novo módulo `tokenization.js`.

a. Instale o módulo do *n-gram* através do npm ([ver documentação](#)).

b. Criar uma função chamada `ngram` (exportar essa função), que recebe como parâmetro de entrada um texto e um valor N e devolve um array com as combinações possíveis.

7. Na diretoria `preprocessing` criar um módulo `stemming.js`.

a. Instalar através do npm o módulo `stemmer` ([ver documentação](#)).

b. Utilizar o `stemmer` para normalizar o texto, reduzindo as palavras à sua forma de base. A função que irá criar deve receber um texto e converter cada palavra nesse texto na sua forma de base. Para tal, pondere a utilização do `split` por espaço ou da função `ngram` criada anteriormente, para criar um array de palavras e iterar cada palavra para substituí-la pela palavra processada pelo algoritmo de *Porter Stemming*.

8. Na diretoria `preprocessing` criar um módulo `index.js`, que deve receber um texto e um número e deverá importar os módulos presentes na mesma diretoria e aplicar as funções exportadas para, em primeiro lugar, limpar o texto, em segundo lugar aplicar o *stemming* e posteriormente dividir o texto por *tokens* mediante o número recebido na função. Esta função deverá exportar um objeto com as propriedades necessárias para visualizar a informação na página a ser construída no exercício seguinte.

9. Crie uma nova página para testar a limpeza de um texto, com duas caixa de texto (uma para colocar um texto e outra para colocar um número) e um botão:

a. Ao clicar no botão, deverá ser feita uma chamada ao servidor para aplicar as funções de limpeza no texto introduzido. O *output* dessa limpeza deverá ser impresso na página.

b. Para além de mostrar o texto "limpo", deverá também mostrar o texto após a aplicação do *stemming* no texto "limpo".

c. Por fim, deverá listar na página todos n-gramas de palavras para o texto introduzido