



Quando e como usar cada etapa

1. Leitura da Fonte (CSV ou PDF)


 **Por quê:** O primeiro passo é acessar os dados de onde eles estão.


 **Quando:** Sempre que você precisa processar textos que vêm de documentos externos.

 **Como:** Usando bibliotecas como `pandas` para CSV ou `PyMuPDF`, `pdfminer`, `PyPDF2` para PDF.

 **Utilidade:** Obter o conteúdo bruto que será trabalhado.

2. Pré-processamento do Texto

 **Por quê:** Textos vêm com formatações variadas, caracteres especiais, que atrapalham a análise.

 **Quando:** Antes de qualquer análise semântica ou sintática.

 **Como:** Corrigir acentuação, remover símbolos, ajustar `encoding`.

 **Utilidade:** Uniformizar o texto para evitar `ruídos` no processamento.

3. Limpeza


 **Por quê:** Palavras irrelevantes, HTML tags, números e símbolos podem atrapalhar os modelos.


 **Quando:** Logo após o carregamento do texto.

 **Como:** Regex, filtros de caracteres, remoção de HTML, etc.

 **Utilidade:** Reduzir o ruído e melhorar a qualidade do texto para análise.

4. Remoção de Stopwords

 **Por quê:** Palavras como "o", "de", "a" não carregam significado útil.

 **Quando:** Durante ou logo após a limpeza.

⚙️ **Como:** Usando listas prontas (`nltk` , `spaCy` , etc).

🎯 **Utilidade:** Focar nas palavras que realmente têm valor semântico.

5. Normalização

📌 **Por quê:** Reduzir variações da mesma palavra (ex: "correndo", "correu", "correr").

📅 **Quando:** Após remoção de stopwords.

⚙️ **Como:** Stemming ou lematização (`nltk` , `spaCy`).

🎯 **Utilidade:** Agrupar palavras com mesmo significado, melhorando análise semântica.

6. Divisão em Chunks com Overlapping

📌 **Por quê:** Modelos de embeddings têm limite de tokens, e chunking mantém o contexto.

📅 **Quando:** Antes de gerar embeddings.

⚙️ **Como:** Dividir texto em pedaços (ex: 500 tokens) com sobreposição (ex: 50 tokens).

🎯 **Utilidade:** Evitar perda de contexto entre pedaços do texto.

7. Geração de Embeddings

📌 **Por quê:** Representar o significado do texto em vetores numéricos.

📅 **Quando:** Depois de dividir em chunks.

⚙️ **Como:** Usar modelos como OpenAI Embeddings, `sentence-transformers` , `transformers` .

🎯 **Utilidade:** Permite busca semântica, clustering, similaridade de textos etc.

8. Armazenamento em Banco Vetorial Local

📌 **Por quê:** Para realizar buscas rápidas por similaridade sem recalculer embeddings.

 **Quando:** Depois da geração dos embeddings.

 **Como:** Usar bancos como FAISS, ChromaDB, Weaviate, Milvus.

 **Utilidade:** Fazer busca semântica com alta performance e escalabilidade.

Encoding é o processo de **converter caracteres humanos (letras, números, símbolos)** em **números binários (0s e 1s)**, para que computadores possam armazenar, transmitir e processar texto.

Exemplo:

- A letra "A" no encoding **UTF-8** é representada como `01000001`.
- Já em outro encoding (ex: ISO-8859-1), pode ter uma representação diferente.

Ruído em PLN (Processamento de Linguagem Natural) é **qualquer informação irrelevante, distorcida ou inútil** que atrapalha a análise do texto.

Exemplos de ruídos:

- **Caracteres especiais inúteis:** `@#$$%&*!`
- **Tags HTML:** `<p>Olá</p>`
- **URLs ou emails:** `https://site.com` , `contato@email.com`
- **Quebras de linha mal formatadas, espaços em excesso**
- **Palavras truncadas**, erros de OCR ou digitalização
- **Textos fora do idioma esperado**

Como remover ruídos?

Você pode:

- Usar regex (expressões regulares)
- Remover padrões repetitivos
- Corrigir erros com bibliotecas (`ftfy` , `unidecode` , etc.)