AVISO DE PROPRIEDADE INTELECTUAL

- ✓ Todo e qualquer conteúdo presente nesse material não deve ser compartilhado, em todo ou em parte, sem prévia autorização escrita por parte do autor.
- ✓ Estão pré-autorizados a manter, copiar e transportar a totalidade desse conteúdo, para fins exclusivos de estudo e controle pessoal, os alunos matriculados em disciplina ou curso de Processamento de Linguagem Natural que tenha sido ministrado em sua totalidade pelo autor, servindo como documento de prova de autorização seu histórico escolar ou declaração da instituição responsável pelo curso, comprovando o referido vínculo.
- ✓ Para o caso de citações de referências extraídas desse material, utilizar:

"CARVALHO, Fabrício Galende Marques de. Notas de aula do curso de Processamento de Linguagem Natural.
São José dos Campos, 2025."

PRÉ-PROCESSAMENTO DE DADOS EM PIPELINE DE PLN

FABRÍCIO GALENDE MARQUES DE CARVALHO

2. PRÉ-PROCESSAMENTO

Nessa seção, serão descritas algumas etapas de pré-processamento da pipeline de PLN.

O objetivo principal é habilitar o leitor a desenvolver componentes de pré-processamento aptos a viabilizar as etapas subsequentes da pipeline, em especial a etapa de extração de características que tipicamente sucede o pré-processamento.

O código-fonte que exemplifica várias das técnicas discutidas nessa seção podem ser encontrados em

https://github.com/fabriciogmc/natural_language_processing/tree/main/python/preprocessing

2.1. DEFINIÇÃO

A etapa de pré-processamento, também conhecida como etapa de **normalização** ou ***text wrangling*** é a primeira etapa de uma pipeline completa de PLN.

Nessa etapa, informações irrelevantes contidas do texto são removidas, são efetuadas conversões de caracteres em maiúsculas (ou minúsculas), palavras são reduzidas às suas formas básicas, são efetuadas correções ortográficas, são efetuadas expansões de abreviaturas e siglas, etc.

De um modo mais sistemático, podem ser listadas as seguintes etapas de pré-processamento básicas em uma pipeline:

- Remoção de marcações (e.g., tags HTML);
- Tokenização de sentenças ou frases;
- Tokenização de palavras;
- Remoção de caracteres especiais;
- Remoção de caracteres acentuados;
- Remoção de palavras de parada (***stop words***);
- Processamento de contrações, siglas e abreviaturas;
- Correção de erros de ortografia;
- Stemização (***stemming***);
- Lematização (***lematization***);
- Rotulação (***tagging***);
- Capitalização e descapitalização.
- Parsing;

Nas próximas seções serão descritas algumas dessas técnicas.

2.2. REMOÇÃO DE MARCAÇÕES

Essa atividade é executada principalmente após a atividade de coleta / raspagem de dados em rede (***web scraping)***.

Marcações (***tags***) e dados não informativos, tais como conteúdos de script, css, propagandas embarcadas em *iframes,* etc., são removidos do texto.

Se essa etapa não for executada, dados ruidosos e não informativos serão alimentados à etapa seguinte do pipeline.

2.3. TOKENIZAÇÃO

Pode ser definida como a quebra do texto em unidades menores (tokens) e mais significativas.

Tokens são unidades independentes que têm atreladas a si uma sintaxe e uma semântica bem definidas.

Tipicamente executa-se a tokenização por sentenças (frases, ***sentence tokenization***) e por palavras (***word tokenization***)

2.3.1. TOKENIZAÇÃO DE SENTENÇAS OU FRASES

Constituem um primeiro nível de quebra do corpus de texto.

Um documento é constituído por vários parágrafos que, por sua vez, podem ser divididos em uma ou mais frases (sentenças).

Algoritmos comuns de tokenização por frases incluem: busca por delimitadores específicos (ex. Pontos finais, ponto e vírgula, etc.), modelos de tokenização pré-treinados e tokenização através de expressão regular (***regex***).

2.3.2. TOKENIZAÇÃO DE PALAVRAS

Consiste na quebra de uma frase em palavras que a constituem.

Tipicamente a saída desse processo é utilizada como entrada para a limpeza e normalização dos dados, diretamente relacionada com a stemização e lematização.

2.4. REMOÇÃO DE CARACTERES ESPECIAIS

Essa etapa é necessária quando a acentuação não agrega significado à expressão ou termo. Nesse caso, acentos que são utilizados somente para enfatizar uma sílaba, por exemplo no caso da língua falada, não precisam ser mantidos.

Cabe ressaltar, entretanto, que em idiomas tais como o português, o acento distingue de modo significativo certas palavras, alterando o sentido da frase ou expressão.

Exemplos:

Ele é um bom estudante. (é – verbo)

Ele vai à aula e ele também dorme (e – conjunção coordenativa aditiva)

2.5. REMOÇÃO DE CARACTERES ESPECIAIS

Geralmente são caracteres não alfanuméricos que acrescentam ruído ao texto não estruturado.

Exemplos típicos são sinais de pontuação tais como exclamações, interrogações, etc.

Em alguns contextos, números podem também não agregar muita informação.

Cuidado! Caracteres especiais podem auxiliar a etapa de tokenização. Portanto, só devem ser removidos em etapas subsequentes!

Emojis, em alguns casos, podem ser considerados como conjuntos de caracteres especiais sujeitos à remoção. Em outras situações, podem ser tratados da mesma forma que siglas, contrações e abreviaturas.

2.6. PROCESSAMENTO DE CONTRAÇÕES, SIGLAS E ABREVIATURAS

Contrações, abreviaturas, siglas, etc., podem representar dificuldade ao sistema de PLN caso não sejam devidamente tratadas

Exemplos:

Copo d'água = Copo de água. (contração d'água)

Ltda = Limitada (abreviatura)

PLN = Processamento de Linguagem Natural (sigla)

Tipicamente abreviaturas, contrações, siglas, etc., são substituídas utilizando-se expressões regulares que operam baseadas em um conjunto de padrões mapeados por dicionários.

Exemplo:

```
{"ltda": "limitada", "pln": "processamento de linguagem natural", "d'água": "de agua"}
```

2.7. CORREÇÃO DE ERROS DE ORTOGRAFIA

Tem como objetivo a substituição de palavras que foram incorretamente escritas considerando-se a ortografia considerada correta.

Algumas vezes palavras podem ser escritas de modo incorreto para expressar uma intensidade ou um sentimento. Outras vezes podem ser escritas de modo incorreto por mero engano/descuido.

Exemplo: Faz muito friiiiiiiiiiiooooooooooooo.

Para o caso de padrões com letras repetidas, uma alternativa é proceder com a sucessiva remoção dessas, seguida de uma busca em um corpus de texto que pode ser utilizado como critério de parada.

Para o caso de grafia incorreta onde pode ocorrer múltiplas correspondência, a alternativa e utilizar uma medida de distância entre a palavra incorreta e outras candidatas, obtidas por alterações nesta.

Exemplo: pota

Substituições candidatas:

{porta, pata, pote, portal}

A substituição correta tipicamente leva em consideração uma análise da proximidade dos padrões de substituições em relação ao texto original e, também a ocorrência no corpus de texto utilizado como referência.

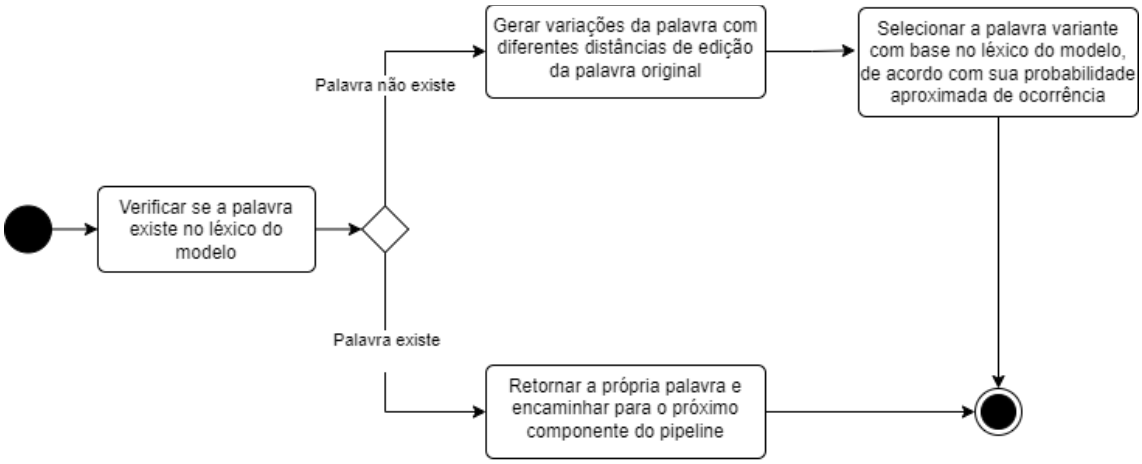
Exemplo: Análise de proximidade (***edit distance)***, considerando a palavra pota:

Porta – uma única operação de inserção entre o e t.

Portal – duas operações de inserção, entre o e t e ao final de da palavra.

O algoritmo de correção deve levar em consideração a probabilidade de ocorrência das substituições candidatas do corpus de texto tomado como referência para efetuar a correção.

Exemplo (cont): Esquematicamente, o mecanismo de correção pode ser implementado de acordo com o seguinte fluxo:



Exemplo: No caso do exemplo em questão, suponha-se que as palavras variantes geradas tenham as seguintes frequências de ocorrência no corpus de texto utilizado como referência:

porta	pote	portal	pata
2/100 = 2%	1/100 = 1%	5/100 = 5%	0/100 = 0%

Nesse caso, a palavra gerada escolhida para substituição seria portal

2.8. STEMIZAÇÃO

É a redução de uma palavra à sua forma raiz sem que haja, no entanto, a preservação do significado, ou seja, o stem não necessariamente faz parte do léxico da linguagem.

Exemplo: aborrecida → aborrec

Nesse caso, houve a eliminação do sufixo (característica do particípio) que foi agregado durante o processo de inflexão.

2.9. LEMATIZAÇÃO

É similar à stemização, mas, nesse caso, a palavra reduzida precisa fazer parte do léxico da linguagem.

Exemplos:

aborrecida → aborrecer

florada → flor

No processo de lematização, é comum que uma determinada classe gramatical seja reduzida a outra de mesma categoria durante o processo (e.g.: verbo flexionado → verbo no infinitivo, etc).

2.10. ROTULAÇÃO

A rotulação, conhecida como ***tagging,*** é a etapa de pré-processamento que identifica categorias de palavras presentes em uma frase de acordo com o contexto local.

A rotulação torna possível o estabelecimento de relações entre as diferentes palavras, substituição por palavras equivalentes, cálculos de estatísticas de frases específicas, entre outras.

Exemplo: O carro está sujo.

Saída do processo de *tagging*: `{o: artigo (DET), carro: substantivo (NN), está: verbo (VB), sujo: adjetivo (ADJ)}`

2.11. CAPITALIZAÇÃO E DESCAPITALIZAÇÃO

São utilizadas primordialmente para facilitar o processo de comparação e casamento de padrões de um ou mais padrões específicos.

Assim como no caso da remoção de caracteres especiais, textos providos de caracteres capitalizados podem auxiliar o processo de tokenização de sentenças e palavras.

2.12. REMOÇÃO DE PALAVRAS DE PARADA

Palavras de parada (***stop words***) são palavras que são muito frequentes em uma linguagem e que podem não agregar muito significado, prejudicando assim o processamento subsequente.

Exemplos:

Na língua portuguesa, tem-se artigos, preposições, etc., tais como {a, as, o, os, de, do, aquele, aquela}.

Observação: Algumas vezes, remover stop words pode alterar significativamente o sentido de uma frase e reduzi-la a outra totalmente diferente.

Exemplo de prejuízo ao pipeline de PLN após a remoção de uma hipotética *stop word,* considerando o seguinte conjunto de stop words sw = {eu, não, do}:

Frase original	Frase após remoção da stop word
Eu não gostei do computador	gostei computador
Eu gostei do computador	gostei computador

Nesse caso, frases com sentidos totalmente opostos acabaram por ser convertidas a expressões textuais equivalentes.

EXERCÍCIOS E PROBLEMAS:

TERMINOLOGIA E CONCEITOS

TC.2.1. Sob o ponto de vista linguístico, qual a diferença entre corpus de texto, parágrafo, frase, oração e palavra? Ilustre com um exemplo e indique como diferentes tipos de significado podem estar atrelados a cada um desses elementos.

TC.2.2. Qual um possível efeito da não remoção de um *iframe* ou de *scripts* e CSS's de um documento capturado através de uma raspagem de dados?

TC.2.3. Considerando que a tokenização corresponde à quebra do corpus de texto em elementos menores, exemplifique como diferentes quebras podem dar origem a diferentes estatísticas associadas a cada unidade menor constituinte (dê um exemplo numérico simples).

TC.2.4. Cite exemplos de **tokens de frases** e **tokens de palavras** que podem significar:

a) Uma opinião negativa referente a um produto de uma loja de vestuário;
b) Uma opinião positiva referente a um produto de uma loja de vestuário;
c) Uma opinião neutra referente a um produto de uma loja de vestuário;
d) Uma opinião negativa relacionada a um carro vendido por uma concessionária automotiva;
e) Uma opinião positiva relacionada a um carro vendido por uma concessionária automotiva;
f) Uma opinião neutra relacionada a um carro vendido por uma concessionária automotiva;

TC.2.5. Explique como a alteração do corpora de texto pode alterar o comportamento de um corretor ortográfico. Ilustre com uma frase simples que apresenta um erro de ortografia e que pode originar uma correção incorreta em virtude do processamento estatístico do corretor.

TC.2.6. Explique o que é uma expressão regular e como ela pode ser utilizada para projeto de corretor ortográfico para o caso de palavras com três ou mais caracteres repetidos.

TC.2.7. Cite três algoritmos de stemização e explique, em linhas gerais com um exemplo textual simples, como eles operam.

TC.2.8. Explique sistematicamente como você pode selecionar stop words para um corpora de texto quando você não tiver disponível uma biblioteca contendo uma lista de stop words. Utilizar

bibliotecas contendo listas de stop words sem efetuar a análise que você citou pode acarretar algum problema prático para a pipeline?

PRÁTICA DE PROGRAMAÇÃO

PP.2.1. Desenvolva um sistema que faça a raspagem de dados de um site contendo opiniões de produtos. O sistema deve possuir uma interface onde o usuário informa uma determinada URL e um botão. A saída do sistema deve ser uma listagem contendo o texto plano das avaliações de um determinado produto.

PP.2.2. Desenvolva um sistema web que receba como entrada uma ou mais frases digitadas pelo usuário. O sistema efetua a tokenização por frases e, a seguir, efetua o tagging das classes gramaticais das frases e exibe, ao lado de cada frase listada, as palavras e seus rótulos, tais como o seguinte exemplo:

Entrada: Eu quero dormir.

Saída: {eu: pronome, quero: verbo, dormir: verbo}

PP.2.3. Tomando como base o código-fonte fornecido pelo professor e efetuando uma raspagem de dados que opere sobre alguma página contendo revisão de produtos, efetue uma comparação de desempenho utilizando 3 modelos de **tokenizadores de sentença**. Qual sua conclusão?

Obs: Na sua análise e resposta, contabilize o número de tokens gerados e explique a diferença. Meça também o tempo necessário para executar cada um dos processos de tokenização.

PP. 2.4. Descreva e exemplifique o comportamento de um tokenizador de sentença utilizando expressões regulares. Utilize um dado de revisão obtido no problema **PP.2.3.** (ou seja efetue a tokenização a partir de um dado recuperado de uma review da internet). Na sua resposta, mostre a quantidade dos tokens de saída após executar o programa e efetue uma validação para avaliar se o desempenho esperado é o obtido.

PP.2.5. Fazer o estudo comparativo de saída e desempenho dos **tokens de palavras** do nltk default, Treebank e Toktok. Utilize dados de uma review de **P.2.3.** No seu estudo, descreva, baseando-se nas saídas dos tokenizadores, quais as principais diferenças observadas quantificando os resultados com exemplos de textos simples.

PP.2.6. Exemplifique o funcionamento de um token de palavras utilizando expressão regular. Explique, com um programa exemplo, como configurar o padrão para obter um comportamento diferente do tokenizador.

PP.2.7. Tomando-se como base os problemas **TC.2.4** e **PP.2.3**, exemplifique a classificação de uma opinião (e.g., um comentário sobre um produto na Internet) pela mera **busca de um ou mais tokens de palavras**. Aplique as etapas de processamento ilustradas na aula de PLN. Na sua resolução mostre o seguite:

a) A obtenção dos tokens a partir da(as) sentença (as) origina(ais).

b) A identificação dos tokens que caracterizam a opinião positiva, negativa ou neutra.

c) A classificação da opinião baseada na mera ocorrência do token (faça um esquema em que um ou mais tokens podem ser utilizados para classificar a opinião).

d) Ilustre um caso onde esse mecanismo de classificação não funciona, mesmo que os tokens sejam encontrados.

PP.2.8. Exemplifique o funcionamento de um corretor ortográfico, aplicável à língua portuguesa, que efetue correção de palavras baseado em um corpus de texto considerado como referência e que utilize métricas de distância e estatísticas de ocorrência de palavras no corpus considerado. Alterar o corpus pode afetar o comportamento do corretor? Se sim, dê um exemplo prático utilizando dados diferentes para o corpus.

A resposta a esse problema deverá ser um programa que:

a) Leia uma frase digitada pelo usuário.

b) Verifique se há ou não palavras potencialmente incorretas.

c) Informe ao usuário a frase potencialmente corrigida ou então diga que a frase aparenta estar correta.

PP.2.9. Aplique técnicas de tokenização e correção de erros de ortografia a dados de revisão de produtos que tenham sido raspados de uma página de revisão da Internet. Ilustre o comportamento e o desempenho do seu trecho de pipeline de PLN identificando os principais gargalos e sugira uma melhoria possível. Esclareça o porquê da ordem dos elementos em sua pipeline.

Na sua resposta ilustre, através de uma aplicação web, desenvolvida em Python, com front end e back end, os seguintes pontos:

a) O tempo necessário para se efetuar a raspagem de dados.

b) O tempo necessário à remoção de ruído

c) O tempo necessário à quebra em tokens de frases e tokens de palavras.

d) Substituição de abreviaturas

d) O tempo necessário para se processar e corrigir palavras digitadas incorretamente.

A entrada para seu programa deve ser uma URL.

As saídas devem ser cada uma das etapas citadas da pipeline de pré-processamento e os tempos necessários para se executar tais etapas.

Exemplo

Entrada	Saída	Etapa	Tempo
http://teste.com.br	"<html>...textos </html>"	Raspagem	5ms
"<html>...textos </html>"	"Produto interessantee. Gostei bastante. Mt!"	Remoção de ruído.	1ms
"Produto interessante. Gostei bastante. Mt!"	['Produto interessantee.', 'Gostei bastante.', 'Mt!"]	Tokenização por frases	2ms
['Produto interessantee.', 'Gostei bastante.', 'Mt!"]	[['Produto', 'interessantee'], ['Gostei', 'bastante'],['Mt']]	Tokenização por palavras	2.5ms
[['Produto', 'interessantee'], ['Gostei', 'bastante'],['Mt']]	[['Produto', 'interessantee'], ['Gostei', 'bastante'],['Muito']]	Expansão de siglas e abreviaturas	1ms
[['Produto', 'interessantee'],	[['produto', 'interessantee'],	Conversão para minúsculas	3ms