



Processamento de Linguagem Natural – Aula 1

Prof. Dr. Rooney R. A. Coelho

Agenda

1. Introdução ao PLN

- O que é e por que é importante
- Pilares: Representação e Modelagem
- Principais tarefas do PLN

2. Pipeline e Pré-processamento

- Etapas: dados → limpeza → modelagem
- Tokenização, lematização e stemming
- Boas práticas de normalização

3. Representações Discretas (Sparse)

- Bag-of-Words e TF-IDF
- Perda de contexto e limitações

4. Classificador Naive Bayes

- Modelo probabilístico básico
- Uso de log-probabilidades
- Limitações e motivação para embeddings



Introdução ao PLN

O Processamento de Linguagem Natural (PLN), do inglês Natural Language Processing (NLP), é um campo de estudo essencial na intersecção entre a ciência da computação, a inteligência artificial e a linguística.

Introdução e Pilares do PLN

Definição do PLN

- O **PLN** é definido como o **estudo de como fazer os computadores processarem, “entenderem” e alavancarem dados de linguagem humana**. Essencialmente, é uma área focada em técnicas e algoritmos para o processamento de dados que consistem em linguagem natural.
- O PLN não se limita apenas ao texto escrito. Ele abrange diversas formas de comunicação simbólica, incluindo o **áudio de fala** (que frequentemente envolve trabalho de Processamento de Sinais), o **texto escrito**, e até mesmo a **linguagem de sinais**.
- É importante notar que, independentemente do modelo computacional utilizado, ele não "entende" a linguagem no sentido humano. O sistema simplesmente **aproxima a compreensão** para atingir um objetivo específico, o que, na prática, tem se mostrado "bom o suficiente" para inúmeras aplicações.

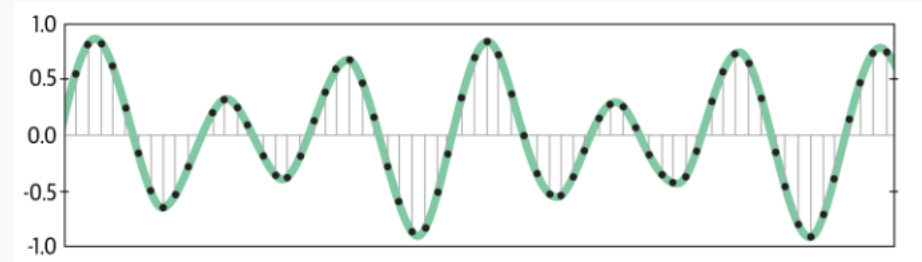
Representação da informação no computador



What We See



What Computers See



[-1274, -1252, -1160, -986, -792, -692, -614, -429, -286, -134, -57, -41, -169, -456, -450, -541, -761, -1067, -1231, -1047, -952, -645, -489, -448, -397, -212, 193, 114, -17, -110, 128, 261, 198, 390, 461, 772, 948, 1451, 1974, 2624, 3793, 4968, 5939, 6057, 6581, 7302, 7640, 7223, 6119, 5461, 4820, 4353, 3611, 2740, 2004, 1349, 1178, 1085, 901, 301, -262, -499, -488, -707, -1406, -1997, -2377, -2494, -2605, -2675, -2627, -2500, -2148, -1648, -970, -364, 13, 260, 494, 788, 1011, 938, 717, 507, 323, 324, 325, 350, 103, -113, 64, 176, 93, -249, -461, -606, -909, -1159, -1307, -1544]

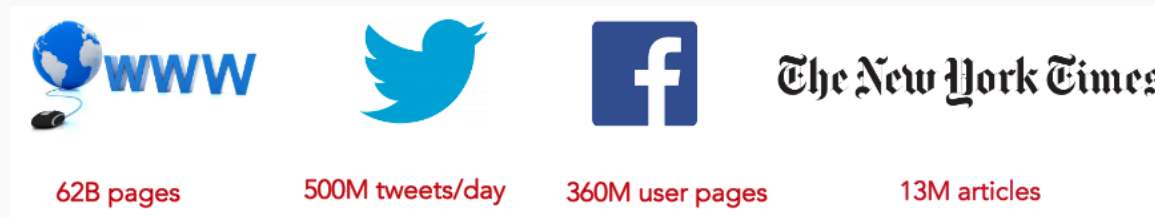
Motivação e Importância do PLN

- A principal motivação para o estudo do PLN reside no fato de que **a linguagem é uma habilidade distintamente humana** e é **fundamental para a evolução humana**. A linguagem é uma parte essencial de como a sociedade humana opera.
- O ponto central da computação é auxiliar os seres humanos. Consequentemente, fazer com que os computadores **“entendam” nossa linguagem** e a forma como nos comunicamos como espécie é um ponto de entrada natural e um passo obrigatório para nos auxiliar significativamente em nossas vidas.



O Desafio da Escala de Dados

- O estudo do PLN é impulsionado pelo imenso volume de dados de linguagem humana que são gerados diariamente. **Nosso mundo digital está inundado de dados, sendo a maior parte deles textuais.**
- A escala desse desafio é massiva, como exemplificado pelos seguintes números que representam a inundação de dados textuais:



- A capacidade de processar, entender e tirar proveito dessa vasta quantidade de dados é o que torna o PLN crucial.

Os Dois Pilares (Cornerstones) do PLN

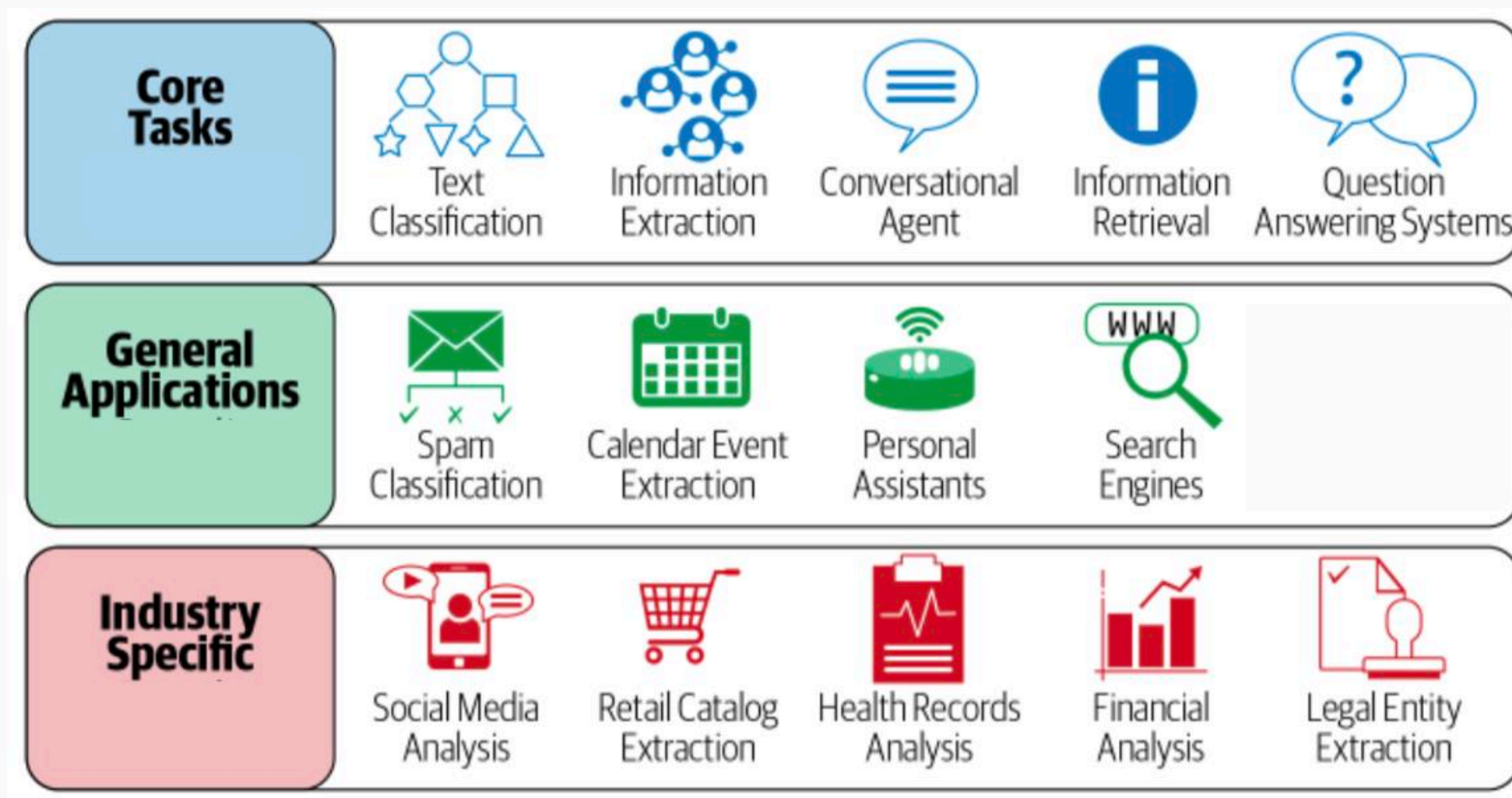
Representação

- Foca em **converter o significado simbólico da linguagem** (palavras, sinais, braile, fala) em uma forma que o computador entenda.
- Como o computador opera apenas com **bits (0 e 1)**, é necessário representar a linguagem em **dados numéricos**, como **vetores densos (embeddings)**.
- O principal desafio é que **a linguagem não possui uma relação direta entre seus bytes e seu significado**, ao contrário de números ou imagens.

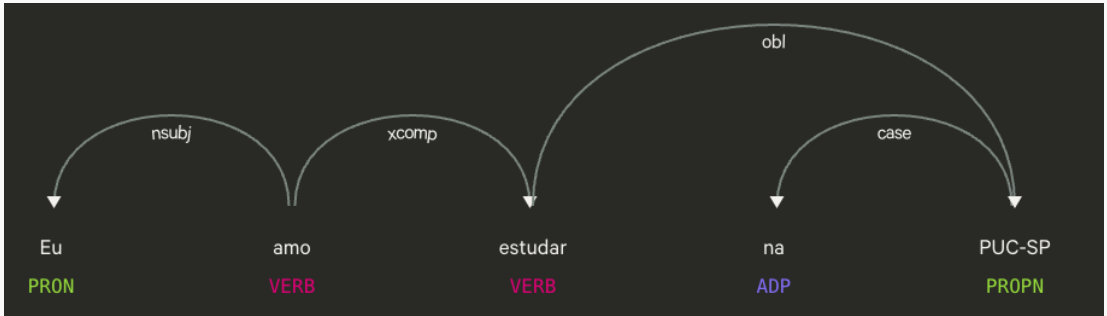
Modelagem

- Usa as **representações numéricas** (vetores, embeddings etc.) para **executar tarefas de linguagem**.
- Envolve o uso de **modelos computacionais**, desde sistemas baseados em regras até **modelos estatísticos** (HMMs, CRFs).
- Hoje, é dominada por **Redes Neurais Profundas**, que lideram as aplicações em **Machine Learning** e **PLN**.

Tarefas Comuns do PLN



Árvore de dependências sintáticas



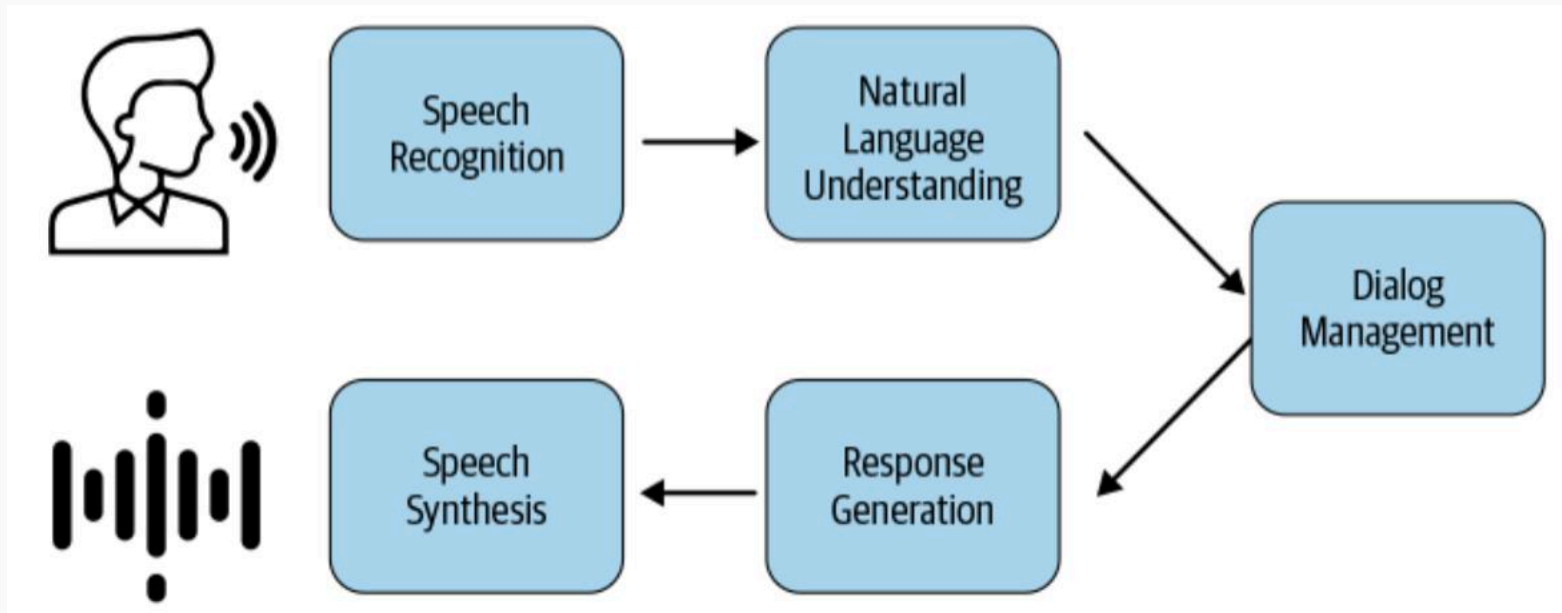
Palavra	Função	Relação (rótulo)	Explicação
Eu	pronome	nsubj (nominal subject)	É o sujeito da frase: quem realiza a ação de amar.
amo	verbo principal	(raiz)	Verbo principal da oração.
estudar	verbo	xcomp (open clausal complement)	É o complemento verbal: “amo estudar ”. Mostra o que é amado.
na	preposição	case	Indica a relação espacial “em + a”. Liga-se a “PUC-SP”.
PUC-SP	substantivo próprio	obl (oblique modifier)	É o termo ligado a “estudar”: onde ocorre o estudo. “Estudar na PUC-SP .”

Named Entity Recognition (NER)

Eu amo estudar no curso de **Ciência de Dados** MISC e **Inteligência Artificial** MISC na **Pontifícia Universidade Católica de São Paulo** LOC com o professor **Rooney Coelho** PER .

Eu amo estudar no curso de **Ciência de Dados** MISC e **Inteligência Artificial** MISC na **Pontifícia Universidade Católica de São Paulo** LOC com o professor **Rooney Coelho** PER . Vou escrever uma empresa conhecida como a **Amazon** ORG para ele rotular corretamente.

Agentes conversacionais



Tarefas Comuns do PLN

Tarefa Comum	Descrição e Contexto nas Fontes
Classificação de Texto (Text Classification)	Envolve categorizar um documento de entrada (d) em uma classe de saída específica (c). É a tarefa mais popular no PLN. Exemplos incluem detecção de spam e análise de sentimentos (e.g., avaliar se uma resenha no Yelp é positiva ou negativa).
Reconhecimento de Entidade Nomeada (NER)	Consiste em encontrar entidades específicas no texto. É uma tarefa de extração de informação que identifica e classifica menções a pessoas, organizações, locais, etc., em texto.
Modelagem de Linguagem (Language Modelling)	É uma tarefa central do PLN e altamente útil para muitas outras tarefas. Um Modelo de Linguagem representa a linguagem usada por uma dada entidade (como uma pessoa, gênero ou classe de texto) e estima a probabilidade de qualquer sequência de palavras. É usado para geração de texto, auto-completar, tradução e sumarização.

Tarefas Comuns do PLN

Tarefa Comum	Descrição e Contexto nas Fontes
Resolução de Coreferência (Coreference Resolution)	A tarefa de determinar quem é quem e o que é o que. Envolve a resolução de ambiguidade linguística, ligando menções pronominais ou nominais (como "ela" ou "o troféu") ao seu antecedente correto no discurso.
Tradução Automática (Machine Translation - MT)	É a tarefa do PLN que visa converter texto de um idioma para outro. Sistemas avançados, como o Google Translate, usaram abordagens revolucionárias como seq2seq + Attention.
Sumarização (Summarization)	Tem como objetivo produzir uma versão abreviada de um texto, mantendo a informação chave e relevante. Pode ser usada para criar resumos, abstracts ou itens de ação de uma reunião.

Common NLP Tasks (aka problems)

Syntax

Morphology

Word Segmentation

Part-of-Speech Tagging

Parsing

- Constituency

- Dependency

Discourse

Summarization

Coreference Resolution

Semantics

Sentiment Analysis

Topic Modelling

Named Entity Recognition (NER)

Relation Extraction

Word Sense Disambiguation

Natural Language Understanding (NLU)

Natural Language Generation (NLG)

Machine Translation

Entailment

Question Answering

Language Modelling

Common NLP Tasks (aka problems)

Syntax

Morphology
Word Segmentation
Part-of-Speech Tagging
Parsing
Constituency
Dependency

Discourse

Summarization
Coreference Resolution

Semantics

Sentiment Analysis
Topic Modelling



“Overall, Pfizer’s COVID-19 vaccine is very safe and one of the most effective vaccines ever produced”

Question Answering
Language Modelling

Common NLP Tasks (aka problems)

Syntax

Morphology

Word Segmentation

Part-of-Speech Tagging

Parsing

Constituency

Dependency

Discourse

Summarization

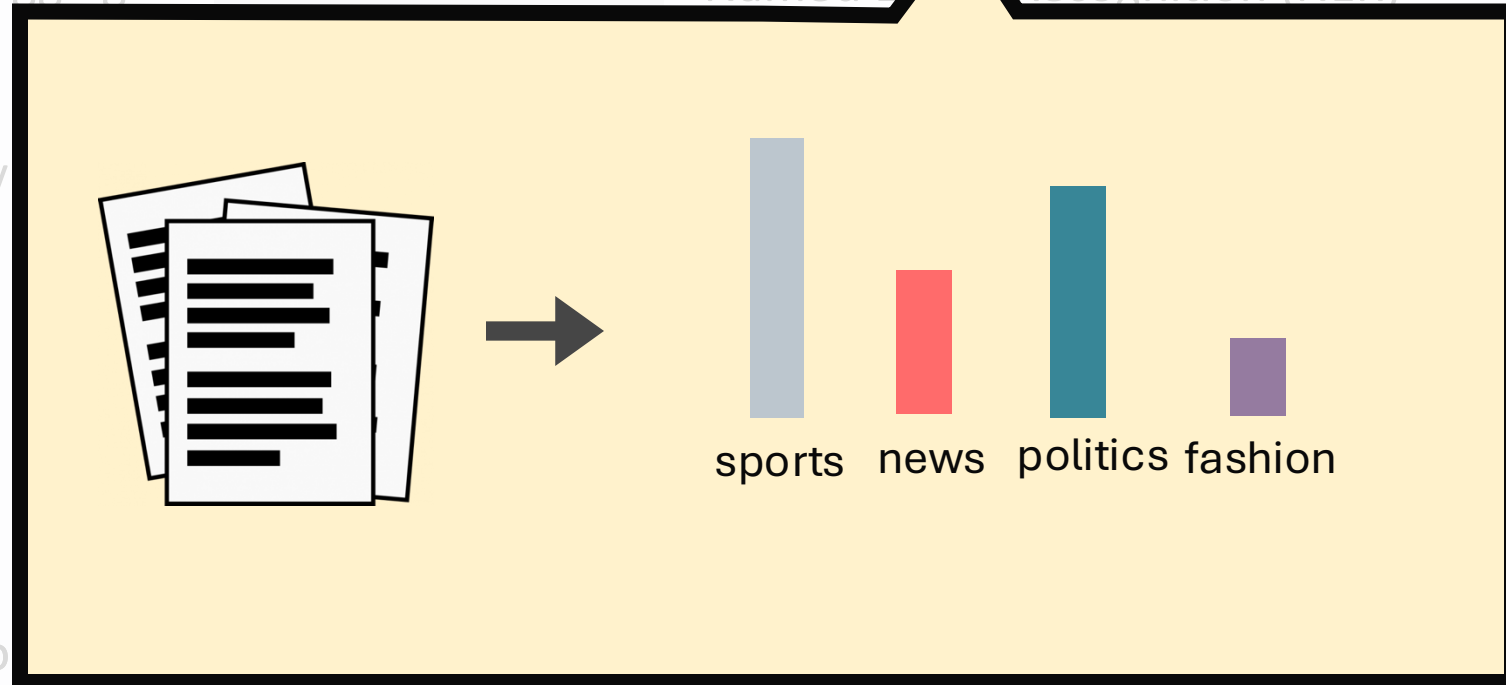
Coreference Resolution

Semantics

Sentiment Analysis

Topic Modelling

Named Entity Recognition (NER)



Language Modelling

Common NLP Tasks (aka problems)

Syntax

Morphology

Word Segmentation

Part-of-Speech

Parsing

Constituency

Dependency

Discourse

Summarization

Coreference Resolution

Semantics

“Alexa, play Drivers License by Olivia Rodrigo”



“Alexa, play Drivers License by Olivia Rodrigo”

INTENT

SONG

ARTIST

Natural Language Understanding (NLU)

Natural Language Generation (NLG)

Machine Translation

Entailment

Question Answering

Language Modelling

Common NLP Tasks (aka problems)

Syntax

Morphology

Word Segmentation

Part-of-Speech Tagging

Parsing

Constituency

Dependency

Discourse

Summarization

Coreference Resolution

Semantics

Sentiment Analysis

Topic Modelling

Named Entity Recognition (NER)

Natural Language Generation (NLG)

Machine Translation

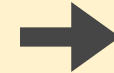
Entailment

Question Answering

Language Modelling

El perro marrón

SPANISH

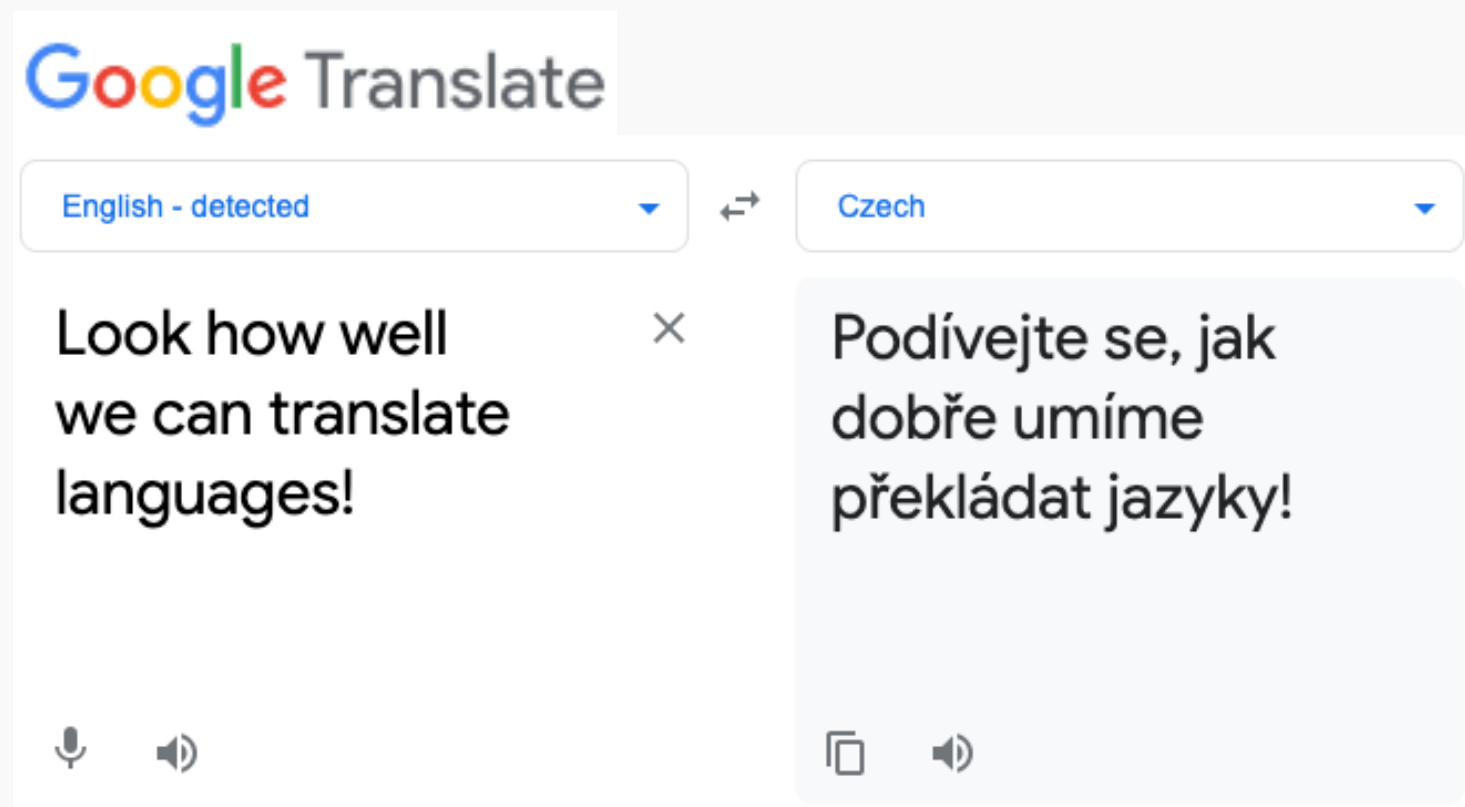


The brown dog

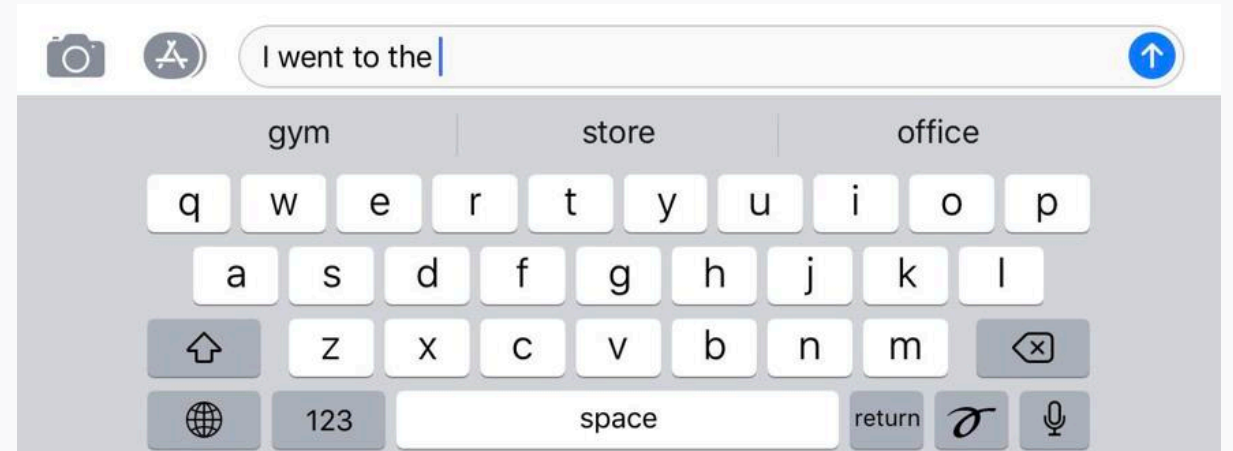
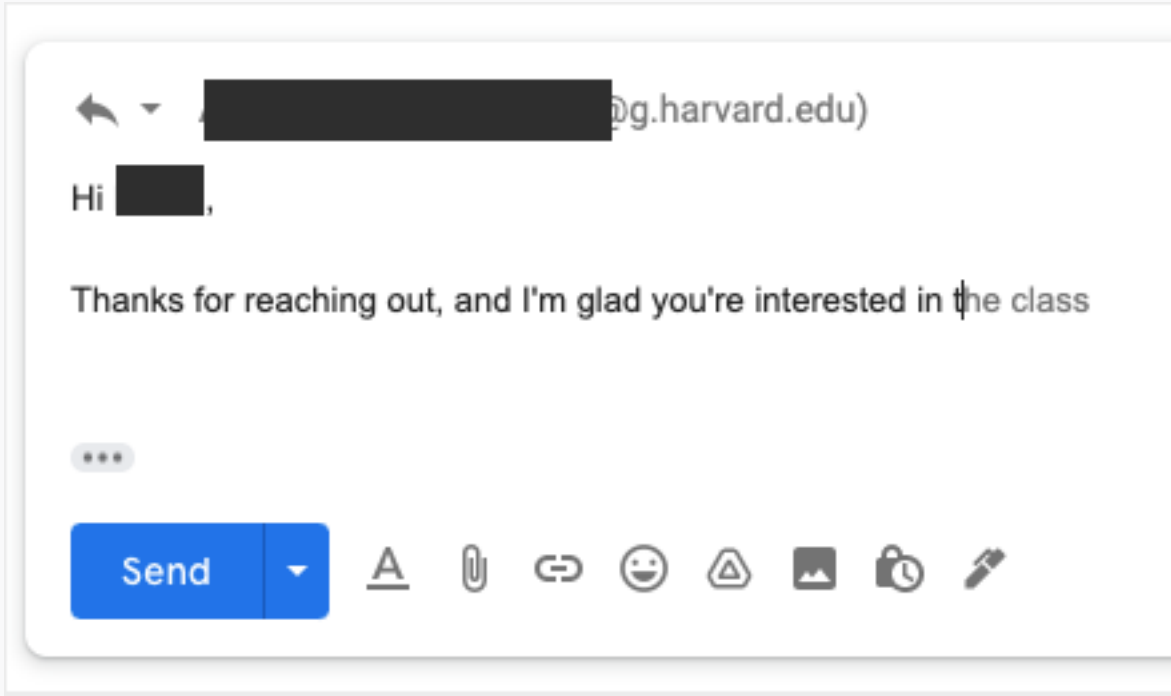
ENGLISH

(NLU)

Tradução



Auto-complete



Classificação de texto



Spam



Not Spam



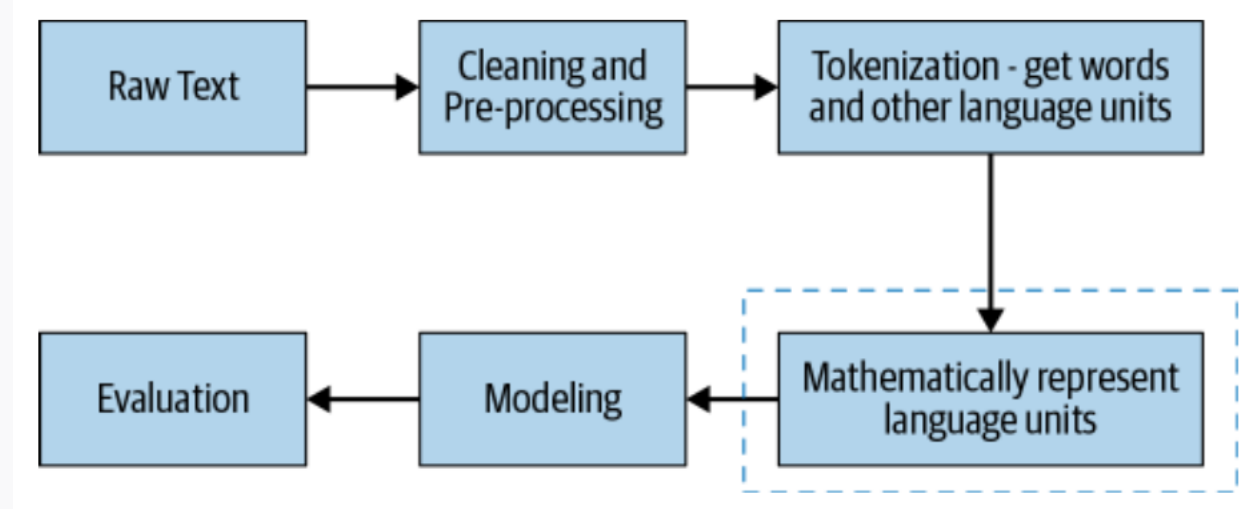
Mecanismos de busca



Pré-processamento de Texto e Pipeline

Etapas da Pipeline Genérica de PLN

- Para construir qualquer sistema de PLN, é necessário seguir um fluxo de trabalho estruturado, frequentemente chamado de **Pipeline de PLN**. Embora o processo real no mundo industrial não seja estritamente linear, pois frequentemente envolve iterações e ciclos de retorno entre as etapas, ele é conceitualmente dividido em fases distintas.



Etapas da Pipeline Genérica de PLN

Etapa	Foco e Detalhes Conceituais
1. Aquisição de Dados	<p>O dado é o coração de qualquer sistema de ML e frequentemente o gargalo de projetos. Nesta fase, define-se a estratégia para coletar dados relevantes, seja usando datasets públicos, rotulados (labeled) ou aumentando a base de dados existente (por exemplo, com rotulagem fraca ou data augmentation).</p>
2. Limpeza (Text Cleaning)	<p>Envolve a extração de texto bruto removendo informações não textuais, como marcações HTML/XML, metadados, e artefatos de raspagem (scraped data). É também onde ocorre a correção ortográfica (especialmente crucial para dados de mídias sociais) e a normalização de Unicode. Esta é uma etapa crítica que afeta todas as fases subsequentes.</p>
3. Pré-processamento	<p>Esta etapa visa converter o texto em uma forma canônica e estruturalmente utilizável, pois o software de PLN normalmente opera no nível de sentença e palavra. Inclui tarefas como Tokenização e Segmentação de Sentença (que são preliminares essenciais), além de Redução de Vocabulário (Stemming/Lematização) e lowercasing.</p>

Etapas da Pipeline Genérica de PLN

Etapa	Foco e Detalhes Conceituais
4. Feature Engineering	<p>Também conhecido como Text Representation, este é o processo de capturar as características do texto em um vetor numérico que possa ser compreendido por algoritmos de ML. O objetivo é transformar o significado textual em valores matemáticos. Pode ser feito via features artesanais (handcrafted) (típico do ML clássico, oferecendo interpretabilidade) ou via representações densas aprendidas diretamente pelo modelo (típico do Deep Learning - DL, melhorando o desempenho, mas reduzindo a interpretabilidade).</p>
5. Modelagem	<p>Envolve a escolha e treinamento do algoritmo (e.g., Regressão Logística, Naive Bayes, ou modelos DL). Projetos geralmente começam com heurísticas simples, especialmente na fase inicial, e aumentam a complexidade à medida que a quantidade e a qualidade dos dados evoluem.</p>
6. Avaliação	<p>Mede o desempenho do modelo em dados não vistos (unseen data). A avaliação é fundamental para a iteração do projeto. Os critérios de avaliação (métrica) devem ser definidos com antecedência e podem incluir acurácia, Precisão, Recall e F1 score.</p>

Tokenização: A Unidade Atômica do Texto

- A tokenização é uma das técnicas de processamento de texto **mais fundamentais** e é a **primeira etapa crucial** de pré-processamento.
- **Definição:** A tokenização é o processo de **dividir uma *string de caracteres em pedaços menores, geralmente palavras***, chamadas *tokens*. Essencialmente, é a etapa de quebrar uma *string* em suas **unidades atômicas** que serão usadas pelo modelo.
- O PLN geralmente opera no **nível de palavra**, pois esta é a menor unidade de significado conveniente.

Dependência do Sistema de Escrita

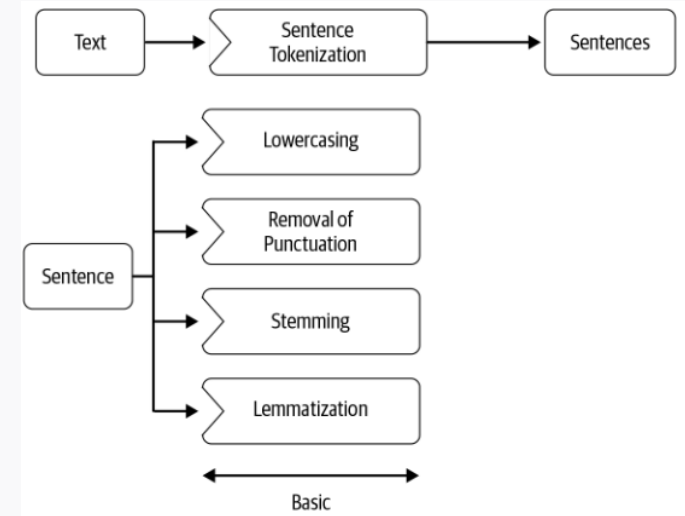
O processo de tokenização **será afetado pelo sistema de escrita da linguagem** com a qual você está trabalhando.

- Em idiomas como o inglês, um tokenizer simples pode usar espaços em branco para dividir as palavras. No entanto, mesmo o inglês apresenta complexidades, como a necessidade de lidar com pontuações (por exemplo, NLP. que deve ser dividido em NLP e .) e abreviações.
- Para idiomas com sistemas de escrita diferentes (por exemplo, Grego ou Ge'ez/Amharic), a forma como a tokenização é realizada precisa ser ajustada.

Dependência do Sistema de Escrita

Existem três estratégias principais de tokenização:

- **Tokenização de Caractere:** Divide o texto em caracteres individuais.
- **Tokenização de Palavra:** Divide o texto em palavras. Se o vocabulário se tornar muito grande (podendo chegar a milhões, devido a declinações, conjugações ou erros de digitação), isso exige um número enorme de parâmetros na rede neural.
- **Tokenização de Subpalavra (*Subword*):** Tenta ser um **compromisso entre as duas**. Palavras frequentes são mantidas como entidades únicas, enquanto palavras raras ou complexas (como erros de grafia) são divididas em unidades menores para gerenciar o tamanho do vocabulário e a sequência de entrada. Exemplos incluem WordPiece (usado pelo BERT) ou BPE (*Byte-Pair Encoding*).



Normalização e Redução de Vocabulário

- Após a tokenização inicial, a **Redução de Vocabulário** torna-se crucial, pois o vetor de *features* inicial terá uma dimensão igual ao tamanho do vocabulário. O objetivo é agrupar palavras que são variações morfológicas da mesma raiz, reduzindo a dimensionalidade e tratando palavras semanticamente similares de forma equivalente. As duas técnicas primárias são Stemming e Lematização.

Lematização (*Lemmatization*)

A lematização é o processo de **mapear todas as diferentes formas de uma palavra à sua forma base, ou *lemma***. O *lemma* é a **palavra de entrada que possui uma entrada completa em um dicionário**.

- **Exemplos:** "gatos" lematiza para "gato" (*cat*), e o adjetivo "melhor" (*better*) lematiza para "bom" (*good*).
- **Requisito de POS Tagging:** A lematização **requer análise linguística e conhecimento do contexto**, e por isso **geralmente exige a marcação da Classe Gramatical (*Part-of-Speech - POS Tagging*)** da palavra para obter o *lemma* correto. Por exemplo, para obter o *lemma* de "run", é preciso saber se ele está sendo usado como verbo ou substantivo.

Stemming

O *stemming* é o processo de **remover sufixos e reduzir uma palavra a uma forma base** (o *stem*) por meio da aplicação de um **conjunto fixo de regras heurísticas**.

- **Resultado:** O resultado do *stemming* pode **não ser uma forma base linguisticamente correta** (ou seja, um *stem* não precisa ser uma palavra real). Por exemplo, "revolution" (*revolução*) pode ser reduzida para "revolut".
- **Vantagens:** O *stemming* tem a vantagem de **não exigir praticamente nenhuma memória** (ao contrário da lematização, que necessita de um dicionário e, frequentemente, um modelo de POS).

Outras Formas de Normalização

A normalização é um passo mais amplo de limpeza baseado em heurísticas, frequentemente executado em conjunto com a redução de vocabulário, e inclui:

- **Lowercasing (Conversão para minúsculas):** Converter todo o texto para minúsculas, pois, em muitos casos, o caso não é relevante para o problema (e.g., classificação de notícias).
- **Remoção de *Stop Words*:** Eliminar palavras muito frequentes que não carregam significado de conteúdo útil para o problema (e.g., "a", "an", "o", "de", "em").

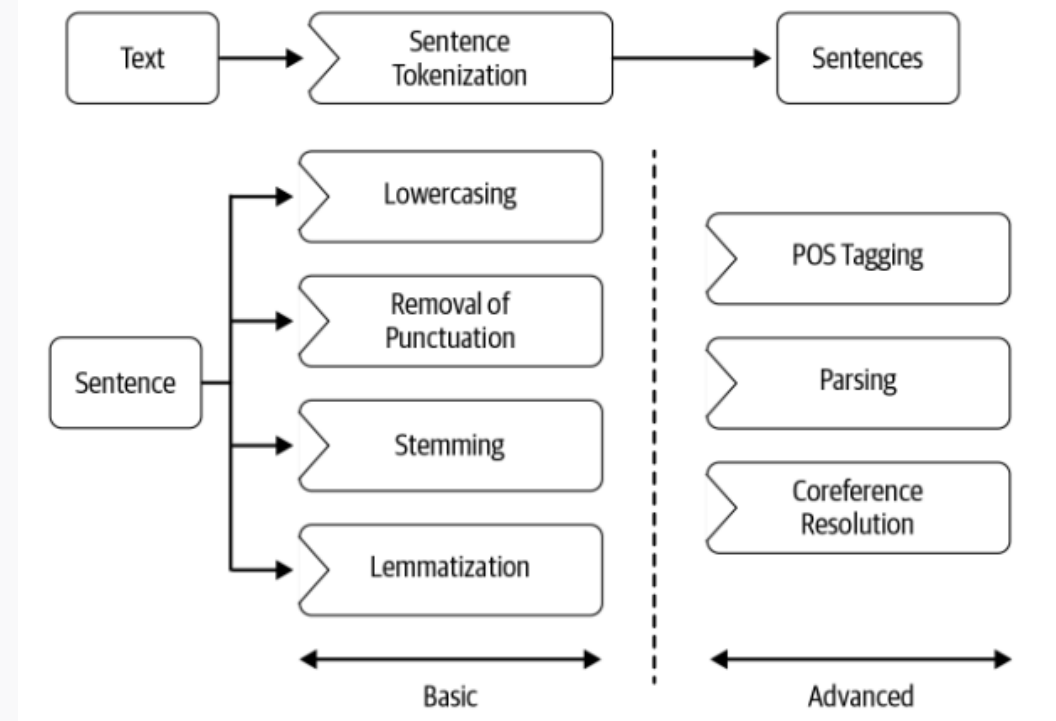
Ordem Recomendada dos Processos

A ordem na qual as etapas de pré-processamento são executadas é crucial, especialmente ao lidar com a lematização, que depende do contexto linguístico.

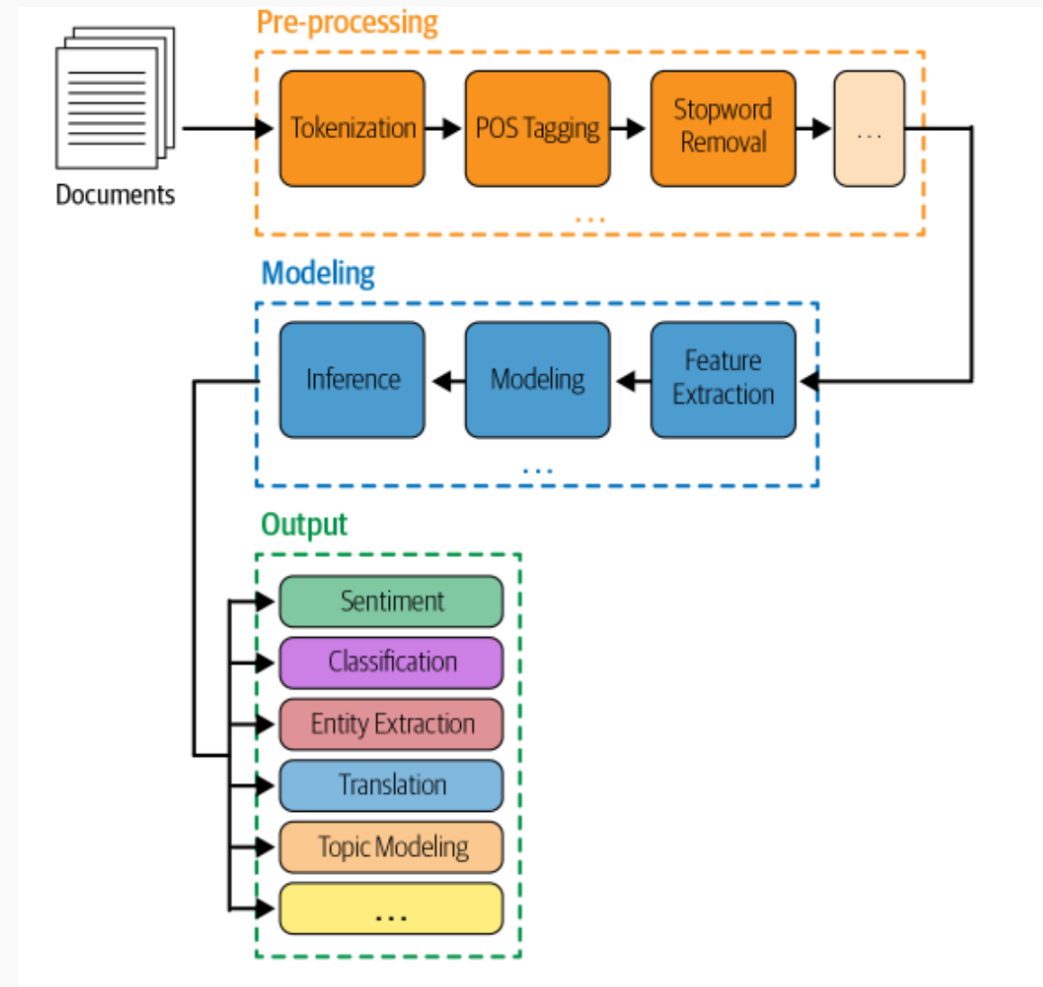
- **A Lematização deve ocorrer antes de remover *stopwords* e aplicar *lowercasing*.**
- É importante **não remover tokens ou aplicar *lowercasing* antes da lematização.**
- O motivo é que, para que a lematização funcione corretamente, é necessário que o **Part-of-Speech Tagging (POS)** seja executado. O POS Tagging, por sua vez, depende de o **contexto completo da sentença e a presença de letras maiúsculas/minúsculas estarem intactos**, pois isso auxilia na determinação da função gramatical da palavra (e, conseqüentemente, do *lemma* correto).
- Já o *lowercasing* é tipicamente realizado **antes do *stemming*.**

Ordem Recomendada dos Processos

Em resumo, a ordem de processamento deve **preservar o máximo de contexto linguístico** (incluindo o caso da letra) para a lematização e o POS Tagging. A remoção de *stopwords* e o *lowercasing* devem ser realizados após a obtenção do *lemma* e antes do *feature engineering* final.



Abordagem clássica para o PLN



POS Tagging (Part-of-Speech Tagging)

- “POS” = classe gramatical (substantivo, verbo, adjetivo etc.)
- Processo de atribuir etiquetas gramaticais a cada palavra.
- Exemplo: “O gato dorme.” → O/DET, gato/NOUN, dorme/VERB
- Base para análise sintática, tradução e outras tarefas de PLN.

Input

Chaplin wrote, directed, and composed the music for most of his films.

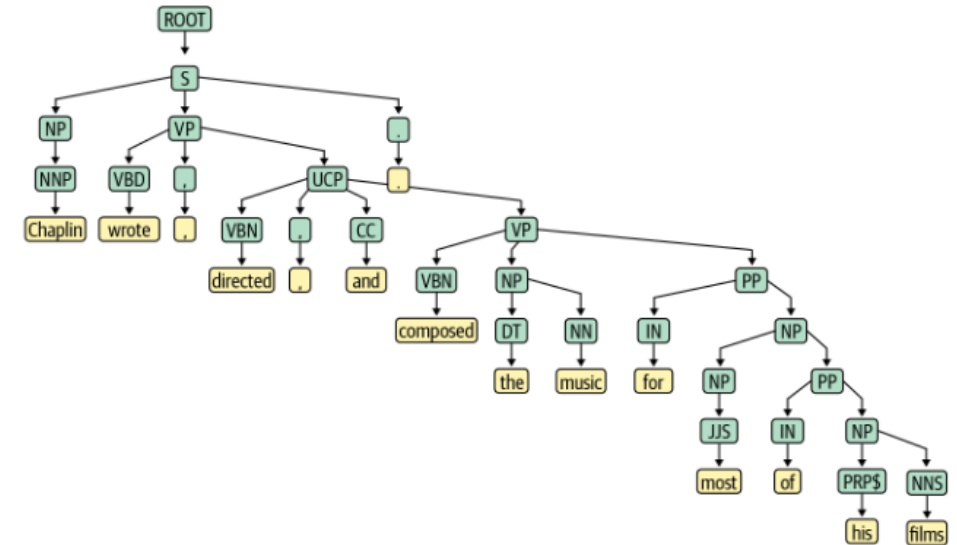
Tokenization with Lemmatization

Chaplin wrote, directed, and composed the music for most of his films.

POS Tagging

Chaplin wrote, directed, and composed the music for most of his films.

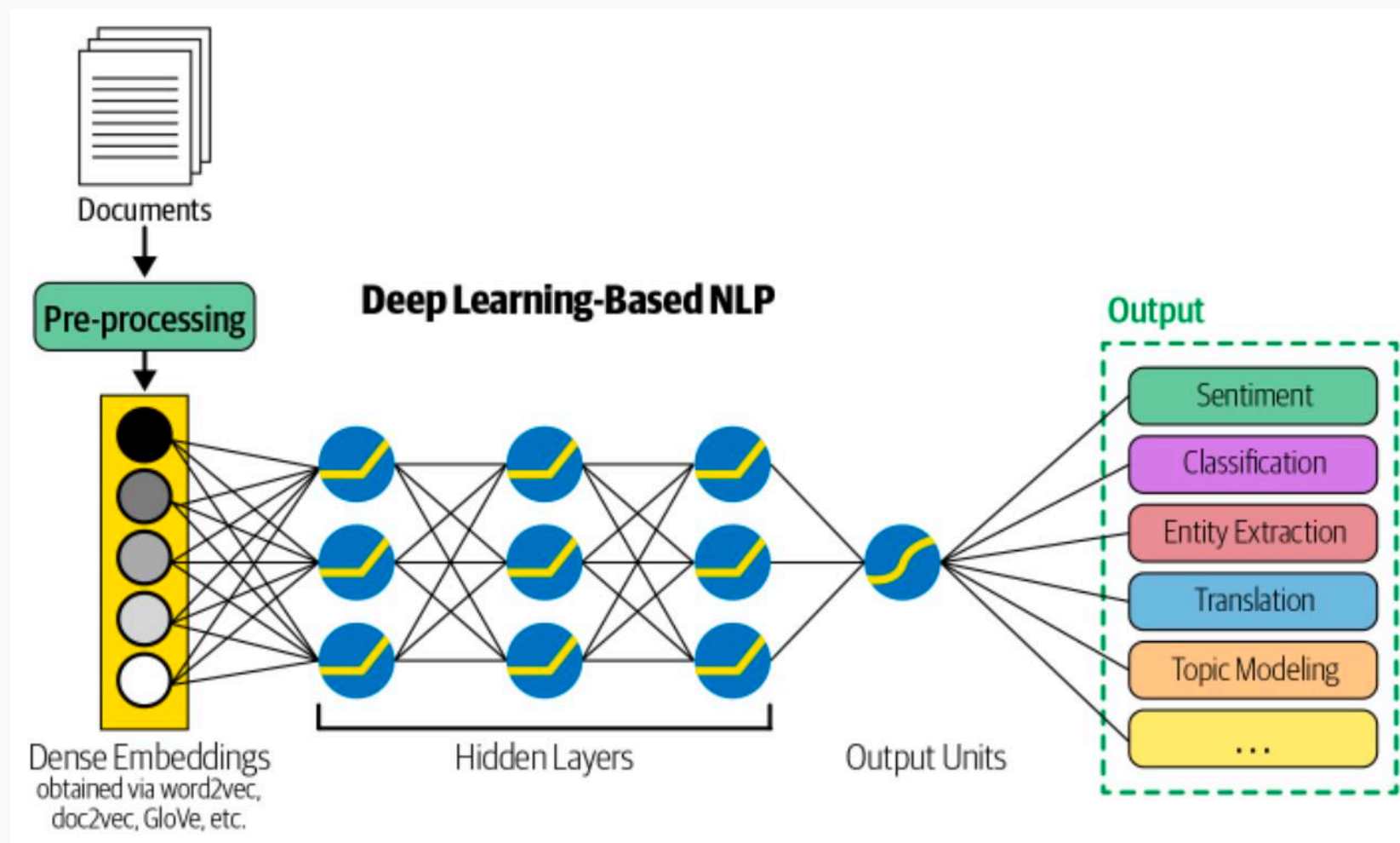
Parse Tree



Coreference Resolution

Chaplin wrote, directed, and composed the music for most of his films.

Abordagem baseada em Deep Learning





Representações de Texto Discretas

Vetorização e o Modelo de Espaço Vetorial (VSM)

Conceito Fundamental: Vetorização de Documentos

- **Definição:** A vetorização é o processo de **representar documentos como vetores de *features*** (características). Este é o princípio central do Modelo de Espaço Vetorial (*Vector Space Model* - VSM).
- **Mecanismo:** Cada **palavra única** no vocabulário total (V) do *corpus* recebe uma dimensão (*feature index*) no vetor.
 - Um documento d é mapeado para um vetor v_d no espaço $\mathbb{R}^{|V|}$.
- **Matematicamente (Representação Inicial):** Documento $d \rightarrow v_d \in \mathbb{R}^{|V|}$
Onde $|V|$ é o tamanho do vocabulário do *corpus*.

Vetorização e o Modelo de Espaço Vetorial (VSM)

O Caráter "Sparse" (Discreto/Esparso)

- **Alta Dimensionalidade (Maldição da Dimensionalidade):** O vocabulário $|V|$ pode ser **extremamente grande**, ultrapassando **100.000 palavras**.
- **Vetores Esparsos:** Como um documento individual contém apenas uma pequena fração das palavras do vocabulário total, a maioria dos valores no vetor v_d será zero.

Objetivo

- Transformar o texto bruto em um formato que possa ser alimentado em algoritmos de Machine Learning (ML).

Bag-of-Words (BoW)

Conceito: Abordagem Contagem-Baseada Clássica

- **Definição:** *Bag-of-Words* (Saco de Palavras) é a técnica mais direta de representação vetorial, onde o texto é tratado como uma "bolsa" ou **coleção de palavras, ignorando completamente a ordem e a estrutura sintática**.
 - Esta representação é **insensível ao contexto** (*context-insensitive*). Por exemplo, "o cavalo comeu" é representado da mesma forma que "comeu o cavalo".
- **Mecanismo:** O valor de cada dimensão no vetor v_d é a **contagem da frequência** com que a palavra correspondente aparece no documento.

Representação BoW (Contagem de Frequência)

O valor do vetor v_d na dimensão i , correspondente à palavra w_i , é a contagem de ocorrências de w_i no documento d :

$$\text{BoW}(w_i, d) = \text{Count}(w_i, d) = f_{w_i}$$

Onde f_{w_i} é o número de vezes que a palavra w_i aparece no documento d .

Exemplo: Para o vocabulário [dog, the, went, fast, ...], o documento "the dog went fast" poderia ser representado como um vetor onde 1 é a presença do termo (versão Booleana, menos comum) ou um vetor onde a contagem é registrada (versão mais comum).

Principais Fraquezas (Shared Weaknesses)

A representação BoW define as bases para as representações esparsas, mas introduz fraquezas cruciais que afetam também o TF-IDF:

1. **Visão achatada (*Flattened View*) do Documento.**
2. **Insensível ao Contexto:** Perda total da informação de ordem.
3. **Maldição da Dimensionalidade (*Curse of Dimensionality*):** O vocabulário pode ter mais de 100.000 termos, resultando em vetores gigantes.
4. **Ortogonalidade:** Não há noção de similaridade semântica no nível da palavra. A distância entre palavras semanticamente relacionadas (e.g., *dog* e *cat*) é a mesma que a distância entre palavras não relacionadas (e.g., *dog* e *chair*): $d(\text{dog}, \text{cat}) = d(\text{dog}, \text{chair})$.

Bag-of-words (BoW)

Let's say our dataset's entire *vocabulary* is just 10 words.
Each unique word can have its own dimension (feature index).

[0	0	0	0	0	0	0	0	0]
dog	the	quick	went	brown	a	jumped	fast	over	store	

NOTE: This is the Boolean version, which isn't the most popular BoW representation

Bag-of-words (BoW)

Each document's vector has a 1 if the word is present. Otherwise, 0.

e.g., “the dog jumped” is represented as

[1	1	0	0	0	0	1	0	0	0]
	dog	the	quick	went	brown	a	jumped	fast	over	store	

NOTE: This is the Boolean version, which isn't the most popular BoW representation

Bag-of-words (BoW)

Each document's vector has a 1 if the word is present. Otherwise, 0.

e.g., “the dog went fast” is represented as

[1	1	0	1	0	0	0	1	0	0]
	dog	the	quick	went	brown	a	jumped	fast	over	store	

NOTE: This is the Boolean version, which isn't the most popular BoW representation

Bag-of-words (BoW)

Imagine a document is a sports broadcast transcript, which concerns a few teams but mostly discusses the local home team, the Cubs

[1 1 1 1 0 1 0 0 0 1]									
baseball	chicago	cubs	the	wrigley	padres	shohei	mvp	homerun	crowd

Bag-of-words (BoW)

We have no indication of *how much* the document is about the Cubs.

baseball	1	1	1	1	0	1	0	0	0	1
chicago										
cubs										
the										
wrigley										
padres										
shohei										
mvp										
homerun										
crowd										

Bag-of-words (BoW)

Now we can see that it's much more about the **Chicago Cubs** than the **Padres**.

[2	9	17	8	0	2	0	0	0	2]
	baseball	chicago	cubs	the	wrigley	padres	shohei	mvp	homerun	crowd	

TF-IDF (Term Frequency-Inverse Document Frequency)

Motivação: Ponderação Baseada na Relevância

- **Problema do BoW:** A contagem bruta de palavras penaliza documentos curtos e supervaloriza palavras muito comuns (como *stopwords*, e.g., "o", "a", "de") que aparecem em quase todos os documentos, mas que não carregam significado discriminativo.
- **Solução TF-IDF:** O TF-IDF tenta quantificar a **importância de uma palavra** ponderando-a. Ele atribui pesos maiores a termos que são frequentes em um documento, mas raros no *corpus* total. O objetivo é **penalizar desproporcionalmente termos comuns que aparecem em muitos documentos**.

TF-IDF (Term Frequency-Inverse Document Frequency)

O score TF-IDF é o produto de duas componentes: a Frequência do Termo (TF) e a Frequência Inversa do Documento (IDF):

$$\text{TFIDF}(w_i, d) = \text{TF}(w_i, d) \times \text{IDF}(w_i)$$

1. Term Frequency (TF)

Mede a frequência com que o termo w_i aparece no documento d . Uma forma comum de calculá-lo é usando a contagem bruta:

$$\text{TF}(w_i, d) = f_{w_i}$$

Onde f_{w_i} é o número de vezes que a palavra w_i aparece no documento d .

TF-IDF (Term Frequency-Inverse Document Frequency)

- **2. Inverse Document Frequency (IDF)**

Mede a raridade do termo w_i no corpus total. Documentos que contêm o termo w_i são contabilizados em $DF(w_i)$. O IDF penaliza termos que aparecem em muitos documentos (alta DF):

$$IDF(w_i) = \log \left(\frac{\# \text{ docs no corpus}}{\# \text{ docs contendo } w_i} \right)$$

Onde a função log é usada para amortecer o efeito da frequência.

TF-IDF (Term Frequency-Inverse Document Frequency)


- Score final que equilibra frequência local (TF) e raridade global (IDF).
- Valor alto → termo é **frequente no documento**, mas **raro no *corpus***.
- O IDF **aproxima-se do tratamento probabilístico do Naive Bayes**, que atribui pouco peso a termos comuns a todas as classes.

$$\text{TFIDF}(w_i, d) = \text{TF}(w_i, d) \times \text{IDF}(w_i)$$

Exemplo Numérico de Ponderação

O exemplo demonstra como o IDF alto aumenta a importância de palavras raras ("bites", "eats"), mesmo que a frequência local (TF) seja menor:

- $D_1 = \text{Dog bites man}$
- $D_2 = \text{Man eats meat}$
- $D_3 = \text{Dog eats food}$
- $D_4 = \text{Cat eats food}$



Dog	bites	man	eats	meat	food
0.136	0.17	0.136	0	0	0

Word	TF score	IDF score	TF-IDF score
dog	$\frac{1}{3} = 0.33$	$\log_2(4/3) = 0.4114$	$0.4114 * 0.33 = 0.136$
bites	$\frac{1}{6} = 0.17$	$\log_2(4/2) = 1$	$1 * 0.17 = 0.17$
man	0.33	$\log_2(4/3) = 0.4114$	$0.4114 * 0.33 = 0.136$
eats	0.17	$\log_2(4/2) = 1$	$1 * 0.17 = 0.17$
meat	$\frac{1}{12} = 0.083$	$\log_2(4/1) = 2$	$2 * 0.083 = 0.17$
food	0.083	$\log_2(4/1) = 2$	$2 * 0.083 = 0.17$

Relação com Naive Bayes

O conceito de penalizar palavras frequentes em muitos documentos (via IDF) **aproxima-se do tratamento probabilístico do Naive Bayes.**

- **Foco na Discriminação:** Assim como o IDF reduz a importância de palavras comuns (que não são discriminativas), o Classificador Naive Bayes, que frequentemente assume que **a ordem das palavras não importa**, atribui pouco peso a termos que são comuns a *todas* as classes, pois eles não contribuem para a probabilidade condicional de uma classe específica.
- **Independência de Palavras:** O uso de representações de contagem (como BoW e TF-IDF) é compatível com a suposição de que as palavras são independentes, uma simplificação central do Naive Bayes.

Fraquezas do TF-IDF

O TF-IDF herda as principais fraquezas da representação BoW:

- **Contexto:** Permanece **insensível à ordem das palavras**.
- **Dimensionalidade:** Continua a sofrer com a **Maldição da Dimensionalidade**.
- **Ortogonalidade:** Não tem conceito de similaridade semântica (e.g., $d(\text{dog}, \text{cat}) = d(\text{dog}, \text{chair})$).

TF-IDF

Notice that longer documents will naturally have higher counts than shorter documents.

[2	9	17	8	0	2	0	0	0	2]
	baseball	chicago	cubs	the	wrigley	padres	shohei	mvp	homerun	crowd	

TF-IDF

Also notice that “the” has a fairly high count, too.

[2	9	17	8	0	2	0	0	0	2]
	baseball	chicago	cubs	the	wrigley	padres	shohei	mvp	homerun	crowd	



TF-IDF

Simple ideas. Let's:

- disproportionately weight the common words that appear in many documents
- Use that info and combine it with the word frequency info



Classificador Naive Bayes e Limitações

Definição e Hipótese Central

Definição e Uso

- O **Classificador Naive Bayes (NB)** é um modelo probabilístico simples e eficiente.
- É frequentemente usado como **modelo *baseline*** (linha de base) em experimentos de classificação de texto.
- É notavelmente **rápido de treinar** e robusto a perturbações.
- No contexto de **Classificação de Texto** (*Text Classification*), o NB é usado para prever a classe c (ex: "spam", "positiva") dado um documento de entrada d .

O Teorema de Bayes

- O Teorema de Bayes permite calcular a probabilidade de uma hipótese (c) ser verdadeira, dadas as evidências (d).
- A fórmula fundamental do Teorema de Bayes é:

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)}$$

Onde cada componente significa:

- $P(c|d)$ (**Probabilidade Posterior**): É a probabilidade que estamos tentando encontrar: a probabilidade de o documento pertencer à classe c , dado que o documento d foi observado.
- $P(d|c)$ (**Probabilidade da Evidência**): É a probabilidade de observar o documento d , dado que a classe c é verdadeira.
- $P(c)$ (**Probabilidade Prévia**): É a probabilidade inicial (antes de ver o documento) de que o documento pertença à classe c .
- $P(d)$ (**Probabilidade da Evidência Total**): É a probabilidade de observar o documento d em qualquer classe.

Por que é importante?

- Permite **atualizar probabilidades** com base em novas evidências.
- É o **fundamento matemático** de vários algoritmos de Inteligência Artificial e Aprendizado de Máquina.
- Ajuda a **tomar decisões sob incerteza**, combinando o que já sabemos (*probabilidade prévia*) com o que observamos (*evidência*).
- O Teorema de Bayes transforma informação nova em conhecimento útil — é a base para “aprender com a experiência”.

Exemplo: E-mail com “promoção”

- No geral, **20%** dos e-mails são spam $\rightarrow P(\text{spam}) = 0,2$
- A maioria dos spams (**90%**) tem "promoção" $\rightarrow P(\text{"promoção"}|\text{spam}) = 0,9$
- Só **10%** dos e-mails normais têm "promoção" $\rightarrow P(\text{"promoção"} \mid \text{"não spam"}) = 0,1$

$$P(\text{spam} \mid \text{"promoção"}) = \frac{P(\text{"promoção"} \mid \text{spam}) \times P(\text{spam})}{P(\text{"promoção"})} = \frac{0,9 \times 0,2}{0,26} = 0,69$$

Aplicação no Classificador Naive Bayes

- No Classificador Naive Bayes, o objetivo é encontrar a classe c^* que maximiza a probabilidade posterior $P(c|d)$:

$$c_{pred}^* = \arg \max_{c \in \mathcal{C}} P(c|d)$$

Para tomar essa decisão de classificação, o Classificador Naive Bayes utiliza uma simplificação do Teorema de Bayes:

- **Ignorando o Denominador ($P(d)$):** O denominador, $P(d)$, representa a probabilidade de observar o documento d . Como o documento d é o mesmo para todas as classes c que estão sendo consideradas, $P(d)$ é uma **constante** e não afeta a determinação de qual classe tem a probabilidade máxima.
- Portanto, a regra de decisão é simplificada para:

$$c_{pred}^* = \arg \max_{c \in \mathcal{C}} P(d|c)P(c)$$

A Hipótese Ingênua (*Naïve Assumption*)

- Para calcular $P(d|c)$, que é a probabilidade do documento, dado a classe, o Naive Bayes aplica a **suposição ingênua** de que **todas as palavras no documento são mutuamente independentes**. O modelo também **assume que a ordem das palavras não importa**.
- Se o documento d consiste nas palavras w_1, w_2, \dots, w_n , a probabilidade do documento dado a classe, $P(d|c)$, é reescrita como o produto das probabilidades de cada palavra individualmente, dado c :

$$P(w_1, w_2, \dots, w_n|c) = \prod_{i=1}^n P(w_i|c)$$

- Combinando o Teorema de Bayes simplificado com a Hipótese Ingênua, a fórmula de decisão final do Naive Bayes se torna:

$$c_{pred}^* = \arg \max_{c \in \mathcal{C}} P(c) \prod_{i=1}^n P(w_i|c)$$

Prevenção de *Underflow*

- Na prática, a multiplicação de muitas probabilidades pequenas (valores entre 0 e 1), como no produto $\prod_{i=1}^n P(w_i|c)$, pode levar ao problema de ***underflow***, onde o resultado é arredondado para zero, causando instabilidade numérica.
- Para evitar isso, a implementação do Naive Bayes utiliza logaritmos, transformando a multiplicação em uma **soma de log-probabilidades**:

$$\arg \max_{c \in \mathcal{C}} \left[\log P(c) + \sum_{i=1}^n \log P(w_i|c) \right]$$

Exemplo

d = Ganhe um brinde grátis na promoção!
como Spam ou Não Spam (\neg Spam).

Palavras relevantes: "grátis" e "promoção".

Probabilidade	Valor	Descrição
$P(\text{spam})$	0.4	40% das mensagens são spam
$P(\neg \text{spam})$	0.6	60% não são spam
$P(\text{grtis} \mid \text{spam})$	0.8	"grátis" é comum em spam
$P(\text{promoo} \mid \text{spam})$	0.7	"promoção" é comum em spam
$P(\text{grtis} \mid \neg \text{spam})$	0.1	"grátis" é rara em não spam
$P(\text{promoo} \mid \neg \text{spam})$	0.05	"promoção" é muito rara em não spam

Cálculos (ignorando $P(d)$):

$$P(\text{spam} \mid d) \propto 0.4 \times 0.8 \times 0.7 = 0.224$$

$$P(\neg \text{spam} \mid d) \propto 0.6 \times 0.1 \times 0.05 = 0.003$$

Decisão:

Como $0.224 > 0.003$, o classificador prevê Spam.

Na próxima aula

- **Introdução a Modelos Featurizados e Embeddings**
 - Motivação: representações densas com menos *features* que o vocabulário
- **Word Embeddings (Type-Based)**
 - Vetores densos para representar significado
 - Cada palavra → um vetor único (*lookup table*)
- **Word2vec e Distribuição Semântica**
 - Ideia de contexto e significado
 - Modelos CBOW e Skip-gram
- **Representações Contextualizadas**
 - Limitações do Word2vec (polissemia)
 - Modelos *token-based* (ELMo, BERT)
 - Aplicações: similaridade e classificação
- **Uso de Embeddings**
 - Geração de *embeddings* com GloVe/Word2vec
 - Representação por média vetorial (*doc2vec simples*)
 - Classificação com Logistic Regression / Naive Bayes

Obrigado!