Inteligência Artificial 2

Aula 14: O Ecossistema Hugging Face

Prof. Aléssio Miranda Júnior
alessio@cefetmg.br

CEFET-MG - Campus Timoteo
Dep. Engenharia de Computação

Inteligência Artificial 2

Sumário

Onde Paramos?

Na aula anterior, desvendamos a **arquitetura Transformer** — o motor da IA moderna:

- Self-Attention: cada palavra "olha" para todas as outras simultaneamente.
- Multi-Head Attention: múltiplas "perspectivas" de atenção em paralelo.
- GPT (Decoder-only) vs BERT (Encoder-only) vs T5 (Encoder-Decoder).

A Pergunta de Hoje

Treinar um Transformer do zero custa **$100M+**.
Como **reutilizar** modelos prontos sem gastar nada?

Objetivos da Sessão

Ao final desta aula, você será capaz de:

1. **Explicar** o conceito de Transfer Learning e por que ele democratizou a IA.
2. **Navegar** no Hugging Face Hub e escolher modelos por tarefa/idioma.
3. **Usar** a biblioteca `transformers` para carregar e inferir com modelos pré-treinados.
4. **Diferenciar** pipelines de classificação, NER, sumarização e geração.
5. **Avaliar** trade-offs entre modelos grandes (precisão) e pequenos (velocidade).

Conexão

Transfer Learning é o equivalente em IA do `import` em Python — você reutiliza trabalho de trilhões de parâmetros.

O Custo Colossal do From Scratch

- O treinamento de um *Large Language Model* (LLM) base, como o GPT-4 ou LLaMA, exige o processamento de **trilhões** de *tokens* na fase de pré-treinamento (*Pre-Training*).

O Custo Colossal do From Scratch

- O treinamento de um *Large Language Model* (LLM) base, como o GPT-4 ou LLaMA, exige o processamento de **trilhões** de *tokens* na fase de pré-treinamento (*Pre-Training*).
- A passagem desses tensores bilionários pelas camadas da rede neural exige altíssima largura de banda. Um único servidor não consegue comportar a operação.

O Custo Colossal do From Scratch

- O treinamento de um *Large Language Model* (LLM) base, como o GPT-4 ou LLaMA, exige o processamento de **trilhões** de *tokens* na fase de pré-treinamento (*Pre-Training*).
- A passagem desses tensores bilionários pelas camadas da rede neural exige altíssima largura de banda. Um único servidor não consegue comportar a operação.
- São criados *Clusters* com mais de 10.000 GPUs NVIDIA (H100, A100) correndo por meses. O custo energético e de infraestrutura atinge a casa dos $100 Milhões de dólares.

O Custo Colossal do From Scratch

- O treinamento de um *Large Language Model* (LLM) base, como o GPT-4 ou LLaMA, exige o processamento de **trilhões** de *tokens* na fase de pré-treinamento (*Pre-Training*).
- A passagem desses tensores bilionários pelas camadas da rede neural exige altíssima largura de banda. Um único servidor não consegue comportar a operação.
- São criados *Clusters* com mais de 10.000 GPUs NVIDIA (H100, A100) correndo por meses. O custo energético e de infraestrutura atinge a casa dos $100 Milhões de dólares.
- Esse custo inacessível excluiu laboratórios universitários e pesquisadores menores, criando um verdadeiro monopólio das Big Techs no avanço de fronteira da IA.

A Solução de Transferência: Transfer Learning

- Se custa dezenas de milhões para treinar um cérebro digital, como startups, alunos e desenvolvedores independentes podem construir produtos revolucionários?

A Solução de Transferência: Transfer Learning

- Se custa dezenas de milhões para treinar um cérebro digital, como startups, alunos e desenvolvedores independentes podem construir produtos revolucionários?
- Graças ao milagre moderno do **Transfer Learning**. Nós não começamos o nosso trabalho com as matrizes "vazias" (iniciadas aleatoriamente).

A Solução de Transferência: Transfer Learning

- Se custa dezenas de milhões para treinar um cérebro digital, como startups, alunos e desenvolvedores independentes podem construir produtos revolucionários?
- Graças ao milagre moderno do **Transfer Learning**. Nós não começamos o nosso trabalho com as matrizes "vazias" (iniciadas aleatoriamente).
- Gigantes (Meta, Google, Mistral, e iniciativas colaborativas) executam a fase pesada e publicam abertamente as matrizes numéricas (*Pesos*) contendo a "língua e sintaxe" fundamentais.
- Qualquer laptop consegue aplicar o *Transfer Learning* treinando minimamente (fino-ajuste) um subconjunto dessas matrizes, pagando frações de centavos na nuvem.

O "GitHub" da Inteligência Artificial

- Para centralizar a distribuição filantrópica de matrizes gigantes, nasceu a **Hugging Face** (inicialmente uma startup de chatbot que pivotou para infraestrutura).

O "GitHub" da Inteligência Artificial

- Para centralizar a distribuição filantrópica de matrizes gigantes, nasceu a **Hugging Face** (inicialmente uma startup de chatbot que pivotou para infraestrutura).
- O *Hugging Face Hub* iniciou focando em Modelos de Linguagem (NLP), mas hoje hospeda centenas de milhares de modelos em Áudio, Vídeo, Biologia Computacional (dobramento de proteínas) e Visão Computacional de código aberto.

O "GitHub" da Inteligência Artificial

- Para centralizar a distribuição filantrópica de matrizes gigantes, nasceu a **Hugging Face** (inicialmente uma startup de chatbot que pivotou para infraestrutura).
- O *Hugging Face Hub* iniciou focando em Modelos de Linguagem (NLP), mas hoje hospeda centenas de milhares de modelos em Áudio, Vídeo, Biologia Computacional (dobramento de proteínas) e Visão Computacional de código aberto.
- Empresas e pesquisadores fazem o *upload* do seu '.safetensors' ou '.bin' (os arquivos de peso). A comunidade consome baixando sob demanda.

A Tríade do Hugging Face Hub

O Hub não tem apenas Modelos. A plataforma abriga um ecossistema trifásico projetado para a reprodutibilidade científica total:

A Tríade do Hugging Face Hub

O Hub não tem apenas Modelos. A plataforma abriga um ecossistema trifásico projetado para a reprodutibilidade científica total:

Models (Pesos)

Mais de 1 milhão de modelos pré-treinados (LLaMA, BERT, Whisper, Flux, Mistral) disponíveis para uso imediato em diferentes arquiteturas.

Datasets

Terabytes de dados de treinamento curados e estruturados para permitir que desenvolvedores realizem *Fine-Tuning* no seu próprio nicho corporativo.

Spaces

Ambientes em nuvem gratuitos contendo interfaces visuais simplificadas (Gradio e Streamlit) para testar os protótipos diretamente pelo navegador.

Transparência Documental: Model Cards

- Ao acessar a página de um modelo, você obrigatoriamente encontra o **Model Card**.

Transparência Documental: Model Cards

- Ao acessar a página de um modelo, você obrigatoriamente encontra o **Model Card**.
- Idealizado para forçar governança e ética em IA, ele age como uma "bula" e descreve:

Transparência Documental: Model Cards

- Ao acessar a página de um modelo, você obrigatoriamente encontra o **Model Card**.
- Idealizado para forçar governança e ética em IA, ele age como uma "bula" e descreve:
  - **Dados de Treino:** Quais corpora ele leu? (Isso evita passivos de violação de direitos autorais empresariais).
  - **Viés (Bias) e Limitações:** Avisos estruturais de se o modelo falha em certas línguas minoritárias ou exibe padrões racistas latentes na base original.
  - **Licença:** O mais crítico. O modelo é Apache 2.0/MIT (uso comercial liberado) ou uma licença proprietária de pesquisa (não comercial)?
  - **Arquitetura Base:** O footprint em memória de VRAM para inferência.

Memória e Tamanho de Parâmetros

- Rodar inferência requer subir os tensores inteiramente para a memória da GPU (VRAM).

Memória e Tamanho de Parâmetros

- Rodar inferência requer subir os tensores inteiramente para a memória da GPU (VRAM).
- A regra de ouro é simples: um modelo em formato *Float16* ou *BFloat16* exige cerca de **2 Gigabytes de RAM de Vídeo** para cada **1 Bilhão de Parâmetros**.

Memória e Tamanho de Parâmetros

- Rodar inferência requer subir os tensores inteiramente para a memória da GPU (VRAM).
- A regra de ouro é simples: um modelo em formato *Float16* ou *BFloat16* exige cerca de **2 Gigabytes de RAM de Vídeo** para cada **1 Bilhão de Parâmetros**.
- O LLaMA-3 com 8B parâmetros precisa de 16 GB de VRAM para rodar e prever tokens sem travar. O modelo GPT-3 possui 175B parâmetros e exigiria $>$ 350 GB de VRAM espalhados em múltiplas placas unidas por barramentos de altíssima velocidade.

Quantização: Rodando IAs Gigantes no Laptop

- Com o movimento Open-Source e para burlar o limite restritivo das GPUs comuns (ex: placas de gamer com 8GB), surgiu uma disciplina paralela poderosa: a **Quantização**.

Quantização: Rodando IAs Gigantes no Laptop

- Com o movimento Open-Source e para burlar o limite restritivo das GPUs comuns (ex: placas de gamer com 8GB), surgiu uma disciplina paralela poderosa: a **Quantização**.
- Em vez de armazenar o *peso* em 16-bits de precisão, aplica-se algoritmos que "arredondam" os pesos da rede para 8-bits ou inacreditáveis 4-bits (`GGUF`, `AWQ`, `GPTQ`).

Quantização: Rodando IAs Gigantes no Laptop

- Com o movimento Open-Source e para burlar o limite restritivo das GPUs comuns (ex: placas de gamer com 8GB), surgiu uma disciplina paralela poderosa: a **Quantização**.
- Em vez de armazenar o *peso* em 16-bits de precisão, aplica-se algoritmos que "arredondam" os pesos da rede para 8-bits ou inacreditáveis 4-bits (`GGUF`, `AWQ`, `GPTQ`).
- Com um modelo GGUF em 4-bits, um modelo de 8B de parâmetros cai de 16GB para algo perto de 5GB de tamanho. Você consegue invocar um "ChatGPT" inteiro localmente utilizando a placa de vídeo de um Macbook sem latência!

O Motor Central: `transformers`

- O Hub é apenas a hospedagem. O que transformou a *Hugging Face* num monstro que vale bilhões foi a sua biblioteca oficial em Python, inteligentemente chamada de `transformers`.

O Motor Central: `transformers`

- O Hub é apenas a hospedagem. O que transformou a *Hugging Face* num monstro que vale bilhões foi a sua biblioteca oficial em Python, inteligentemente chamada de `transformers`.
- O objetivo dela é **abstrair a fúria matemática** do PyTorch e JAX.

O Motor Central: `transformers`

- O Hub é apenas a hospedagem. O que transformou a *Hugging Face* num monstro que vale bilhões foi a sua biblioteca oficial em Python, inteligentemente chamada de `transformers`.
- O objetivo dela é **abstrair a fúria matemática** do PyTorch e JAX.
- Você não precisa codificar dezenas de matrizes Q, K, V e *FeedForward Networks*. A biblioteca orquestra o download dos binários da rede para a pasta local `~/.cache/huggingface` e inicializa as classes preenchendo as matrizes matematicamente corretas.

A Classe Mãe: Pipeline

- O nível de abstração máximo da biblioteca é a função `pipeline()`.

A Classe Mãe: Pipeline

- O nível de abstração máximo da biblioteca é a função `pipeline()`.
- Para o engenheiro corporativo focado em entregar valor rápido, ela esconde todo o mecanismo encadeado da geração (leitura, inferência e tradução do output).

A Classe Mãe: Pipeline

- O nível de abstração máximo da biblioteca é a função `pipeline()`.
- Para o engenheiro corporativo focado em entregar valor rápido, ela esconde todo o mecanismo encadeado da geração (leitura, inferência e tradução do output).
- Ela assume as restrições padrão de inferência da máquina e executa o fluxo inteiro em segundo plano. Mas o que exatamente a `pipeline` está orquestrando no escuro?

A Anatomia Interna de um Pipeline: Três Passos

Quando mandamos a string *"O filme foi terrível"* para a API avaliar, três estágios obrigatórios ocorrem:

A Anatomia Interna de um Pipeline: Três Passos

Quando mandamos a string *"O filme foi terrível"* para a API avaliar, três estágios obrigatórios ocorrem:

1. **1. O Tokenizer**: Transforma *Strings* em IDs Matemáticos através de um dicionário específico daquele modelo (geralmente *Subword Tokenization* como BPE). A Rede Neural jamais vê a string. Ela vê um vetor de inteiros: `[101, 8820, 3140, 102]`.

A Anatomia Interna de um Pipeline: Três Passos

Quando mandamos a string *"O filme foi terrível"* para a API avaliar, três estágios obrigatórios ocorrem:

1. **1. O Tokenizer**: Transforma *Strings* em IDs Matemáticos através de um dicionário específico daquele modelo (geralmente *Subword Tokenization* como BPE). A Rede Neural jamais vê a string. Ela vê um vetor de inteiros: `[101, 8820, 3140, 102]`.
2. **2. O Forward Pass (O Modelo)**: Ocorre a passagem multiplicativa pelas dezenas de camadas *Encoder/Decoder* usando Auto-Atenção. O resultado puramente aritmético é um vetor de saídas latentes de alto nível.

A Anatomia Interna de um Pipeline: Três Passos

Quando mandamos a string *"O filme foi terrível"* para a API avaliar, três estágios obrigatórios ocorrem:

1. **1. O Tokenizer**: Transforma *Strings* em IDs Matemáticos através de um dicionário específico daquele modelo (geralmente *Subword Tokenization* como BPE). A Rede Neural jamais vê a string. Ela vê um vetor de inteiros: `[101, 8820, 3140, 102]`.
2. **2. O Forward Pass (O Modelo)**: Ocorre a passagem multiplicativa pelas dezenas de camadas *Encoder/Decoder* usando Auto-Atenção. O resultado puramente aritmético é um vetor de saídas latentes de alto nível.
3. **3. O Pós-Processamento (Logits)**: Converte a tensão algébrica não normalizada da última camada neural nas respostas legíveis que o usuário espera, usando Softmax ou ArgMax (Ex: `[POSITIVO: 1% , NEGATIVO: 99%]`).

RAG versus Fine-Tuning no Ecosssistema

- Muitos pesquisadores e empresas se deparam com a pergunta fundamental: *"Devo treinar um modelo base com meus dados (Fine-Tuning) ou usar RAG?"*

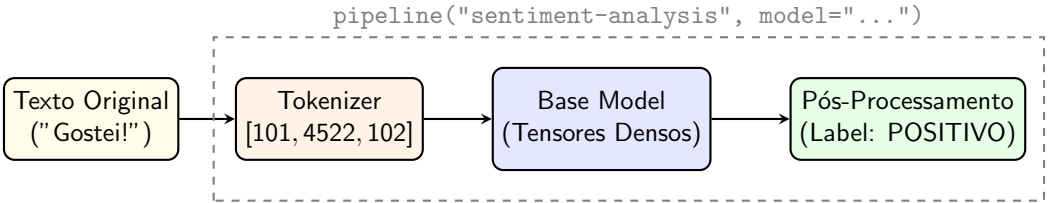
RAG versus Fine-Tuning no Ecosssistema

- Muitos pesquisadores e empresas se deparam com a pergunta fundamental: *"Devo treinar um modelo base com meus dados (Fine-Tuning) ou usar RAG?"*
- **Fine-Tuning (Ajuste Fino):** Utilizado apenas quando você precisa ensinar ao modelo um novo **estilo, sintaxe ou comportamento** (ex: falar como um pirata ou aprender a escrever em uma linguagem de programação obscura). Não deve ser usado para memorizar fatos, pois ele "alucina" com frequência.

RAG versus Fine-Tuning no Ecosssistema

- Muitos pesquisadores e empresas se deparam com a pergunta fundamental: *"Devo treinar um modelo base com meus dados (Fine-Tuning) ou usar RAG?"*
- **Fine-Tuning (Ajuste Fino):** Utilizado apenas quando você precisa ensinar ao modelo um novo **estilo, sintaxe ou comportamento** (ex: falar como um pirata ou aprender a escrever em uma linguagem de programação obscura). Não deve ser usado para memorizar fatos, pois ele "alucina" com frequência.
- **RAG (Retrieval-Augmented Generation):** Utilizado quando você precisa que o modelo tenha acesso a **conhecimento factual corporativo** e atualizado. É mais barato, seguro e permite rastrear a fonte da informação.

Representação Visual: Pipeline Interno



Vantagem da Abstração

- Em *PyTorch* bruto, escrever esses três estágios garantiria facilmente 150 linhas de código altamente propenso a erros de dimensionamento (os famosos erros `Tensor shape mismatch`).

Vantagem da Abstração

- Em *PyTorch* bruto, escrever esses três estágios garantiria facilmente 150 linhas de código altamente propenso a erros de dimensionamento (os famosos erros `Tensor shape mismatch`).
- Com o método pipeline, o programador só se preocupa com a escolha e a performance do modelo em si para a sua tarefa. E tudo é executado na própria máquina (Local). Sem APIs fechadas e sem custo de nuvem.

O Conceito de Tasks Específicas

- O ecossistema é dividido estritamente em Tarefas *(Tasks)*.

O Conceito de Tasks Específicas

- O ecossistema é dividido estritamente em Tarefas *(Tasks)*.
- Cada classe de problema que o modelo foi treinado para realizar tem um *endpoint* específico na biblioteca. Se o seu modelo foi treinado (fez *Fine-tuning*) apenas para traduzir textos, ele falhará catastroficamente se você tentar chamá-lo num pipeline de detecção de entidades.

O Conceito de Tasks Específicas

- O ecossistema é dividido estritamente em Tarefas *(Tasks)*.
- Cada classe de problema que o modelo foi treinado para realizar tem um *endpoint* específico na biblioteca. Se o seu modelo foi treinado (fez *Fine-tuning*) apenas para traduzir textos, ele falhará catastroficamente se você tentar chamá-lo num pipeline de detecção de entidades.
- Tarefas comuns: `text-classification`, `token-classification`, `translation`, `question-answering`, `summarization`.

Código 1: Análise de Sentimentos

- *Text Classification* puro. Uma das tarefas corporativas mais utilizadas para rastrear triagem automática em tickets do Zendesk ou no Twitter/X.

Código 1: Análise de Sentimentos

- *Text Classification* puro. Uma das tarefas corporativas mais utilizadas para rastrear triagem automática em tickets do Zendesk ou no Twitter/X.

```
from transformers import pipeline

# Se nao passarmos o parametro "model", ele baixa o default da HuggingFace
classificador = pipeline("sentiment-analysis")

texto_cliente = "A tela do meu celular quebrou no primeiro dia, odiei."
res = classificador(texto_cliente)

print(res)
# Saida provavel: [{'label': 'NEGATIVE', 'score': 0.98}]
```

Código 2: Reconhecimento de Entidades (NER)

- O *Named Entity Recognition (Token Classification)* lê o texto e mapeia os tensores que representam Organizações (ORG), Pessoas (PER) ou Locais (LOC). Essencial para extração de LGPD (Anonimização de nomes) e metadados RAG.

Código 2: Reconhecimento de Entidades (NER)

- O *Named Entity Recognition (Token Classification)* lê o texto e mapeia os tensores que representam Organizações (ORG), Pessoas (PER) ou Locais (LOC). Essencial para extração de LGPD (Anonimização de nomes) e metadados RAG.

```
from transformers import pipeline

ner_pipe = pipeline("ner", grouped_entities=True)

texto = "O CEO da Microsoft, Satya Nadella, anunciou data centers no
    Brasil."
entidades = ner_pipe(texto)

for e in entidades:
    print(f"Entidade: {e['word']} | Tipo: {e['entity_group']}")

# Saida: Microsoft (ORG), Satya Nadella (PER), Brasil (LOC)
```

Código 3: Zero-Shot Classification

- **A Magia Absoluta:** Antes dos modelos maciços, um triador de problemas exigia meses curando e anotando dados, montando arquivos CSV gigantes.
- Modelos *Zero-Shot* deduzem a taxonomia das categorias mesmo sem nunca ter visto o seu conjunto de dados interno da empresa antes!

Código 3: Zero-Shot Classification

- **A Magia Absoluta:** Antes dos modelos maciços, um triador de problemas exigia meses curando e anotando dados, montando arquivos CSV gigantes.
- Modelos *Zero-Shot* deduzem a taxonomia das categorias mesmo sem nunca ter visto o seu conjunto de dados interno da empresa antes!

```
from transformers import pipeline

classif_zero = pipeline("zero-shot-classification")

ticket = "Preciso do PDF com meu extrato de rendimentos."
opcoes = ["vendas", "devolução", "suporte hardware", "financeiro"]

resultado = classif_zero(ticket, candidate_labels=opcoes)

print(resultado["labels"][0]) # Resposta: financeiro
```

Além do Pipeline: AutoModel e AutoTokenizer

- Quando um Engenheiro de IA atinge a maturidade no uso da arquitetura (ou deseja publicar em um serviço escalável FastApi), ele abandona a conveniência do método `pipeline()`.

Além do Pipeline: AutoModel e AutoTokenizer

- Quando um Engenheiro de IA atinge a maturidade no uso da arquitetura (ou deseja publicar em um serviço escalável FastApi), ele abandona a conveniência do método `pipeline()`.
- Ele instancia estritamente os componentes usando as classes mães do framework: `AutoTokenizer` e `AutoModelForSequenceClassification`.

Além do Pipeline: AutoModel e AutoTokenizer

- Quando um Engenheiro de IA atinge a maturidade no uso da arquitetura (ou deseja publicar em um serviço escalável FastApi), ele abandona a conveniência do método `pipeline()`.
- Ele instancia estritamente os componentes usando as classes mães do framework: `AutoTokenizer` e `AutoModelForSequenceClassification`.
- **Por quê?** O controle preciso de alocação (enviar o tensor pro Cuda / Placa de vídeo usando `tensor.to('cuda')`), gerenciamento de batchs de tensores em paralelo e injeção do cache da rede para acelerar a geração auteregressiva do *Transformer*.

O Novo Perfil Profissional

- O ecossistema Hugging Face transformou violentamente o mercado de carreiras em Inteligência Artificial.

O Novo Perfil Profissional

- O ecossistema Hugging Face transformou violentamente o mercado de carreiras em Inteligência Artificial.
- **Antes de 2020:** O engenheiro era focado no detalhe matemático infernal (90% de cálculo, derivadas de matriz e hiperparâmetros obscuros num Jupyter local).

O Novo Perfil Profissional

- O ecossistema Hugging Face transformou violentamente o mercado de carreiras em Inteligência Artificial.
- **Antes de 2020:** O engenheiro era focado no detalhe matemático infernal (90% de cálculo, derivadas de matriz e hiperparâmetros obscuros num Jupyter local).
- **Hoje (Engenheiro de IA Aplicada):** Atuamos como **Arquitetos Lógicos e Compositores**. Nós avaliamos os limites dos *Foundational Models*, encadeamos os fluxos e orquestramos componentes (Ligue um Banco Vetorial, a um modelo NER local, a um LLM na Cloud).

O Novo Perfil Profissional

- O ecossistema Hugging Face transformou violentamente o mercado de carreiras em Inteligência Artificial.
- **Antes de 2020:** O engenheiro era focado no detalhe matemático infernal (90% de cálculo, derivadas de matriz e hiperparâmetros obscuros num Jupyter local).
- **Hoje (Engenheiro de IA Aplicada):** Atuamos como **Arquitetos Lógicos e Compositores**. Nós avaliamos os limites dos *Foundational Models*, encadeamos os fluxos e orquestramos componentes (Ligue um Banco Vetorial, a um modelo NER local, a um LLM na Cloud).
- A genialidade hoje reside **exclusivamente no Encadeamento Lógico e na Curadoria Sistêmica**.

Segurança Cibernética: Safetensors vs Pickle

- Antigamente, os modelos em Python eram exportados via `Pickle` (`.bin` ou `.pkl`).
- **O Perigo:** O formato Pickle não armazena apenas matrizes, ele armazena **código executável**. Hackers faziam *upload* de modelos falsos no Hugging Face que, ao serem baixados via pipeline, executavam cavalos de tróia no seu servidor corporativo.
- Para sanar isso, a Hugging Face inventou o formato `.safetensors`. É estritamente matemático, imune a injeções de código executável e carrega as matrizes na GPU incrivelmente mais rápido!

Destilação de Modelos (Knowledge Distillation)

- Um problema grave do BERT original (110M de parâmetros) é o alto custo em milissegundos para responder um único usuário em produção.
- Para resolver isso, usamos o processo de **Destilação**. Um "Professor" (o BERT gigante) ensina e corrige um "Aluno" (uma rede neural menor).
- O resultado é o **DistilBERT**: retém 97% da capacidade de linguagem do professor, mas sendo 40% menor em tamanho e 60% mais rápido na inferência. *Perfeito para aplicações em tempo real!*

A Barreira Idiomática (O Problema PT-BR)

- Muitos iniciantes copiam o código `pipeline("sentiment-analysis")` da documentação e jogam frases em português. O modelo, que foi treinado puramente em inglês, começa a soltar classificações caóticas e aleatórias.
- Você precisa sempre vasculhar o Hub atrás de um modelo **Multilíngue** (`mBERT`) ou treinado especificamente pela comunidade brasileira (ex: `neuralmind/bert-base-portuguese-cased`, vulgo BERTimbau).
- Passar a string literal do modelo no pipeline: `pipeline("ner", model="nome-do-modelo")` resolve o problema.

Limitações das Tarefas Base (Pipelines)

- **Não fazem milagres em nichos extremos.** Se você rodar um NER (Named Entity) num prontuário médico, ele não reconhecerá nomes de remédios ou sintomas como entidades vitais (porque foi treinado apenas para reconhecer Nomes, Empresas e Locais).
- **Solução Clássica:** Você pega o modelo base do Hugging Face e aplica um *Fine-Tuning* de algumas horas usando o banco de dados do seu hospital.
- **Solução Moderna:** Você joga a extração para um LLM Generativo forte (GPT-4) resolver isso através de *Engenharia de Prompts* (assunto da próxima aula!).

Como a Comunidade Contribui?

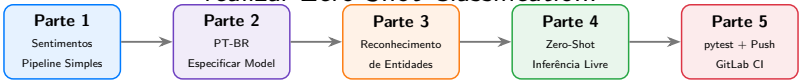
- **Você também pode publicar.** A biblioteca `transformers` possui o método mágico: `model.push_to_hub("meu_nome/modelo_medico_br")`.
- Em 5 segundos os seus pesos locais sobem para a plataforma.
- Assim que a barra de upload fechar, qualquer aluno da turma (ou pessoa no planeta) pode baixar a sua IA na máquina deles fazendo: `pipeline("text-classification", model="meu_nome/modelo_medico_br")`.

Recapitulação - Hugging Face e a Descentralização

1. **Transfer Learning**: Reutilizar os trilhões de cálculos pagos pelas Big Techs.
2. **Hugging Face Hub**: O repositório global contendo modelos, datasets e demonstrações.
3. **Biblioteca** `transformers`: Abstrai PyTorch via `pipeline()`.
4. **Tasks Específicas**: Você baixa a especialidade correta (NER, Sentimento, Resumo).
5. **Hardware e Eficiência**: A revolução de rodar IA local na CPU via `distilbert` e arquivos `.safetensors`.

Roteiro do Laboratório 14

Missão: Criar instâncias de `pipeline` para analisar sentimentos, detectar entidades (NER) e realizar Zero-Shot Classification.



O Poder do Local

Tudo rodará na sua própria máquina (CPU). Observe no terminal o log de download das arquiteturas `safetensors` sendo salvas no seu disco.

Lab Parte 1 e 2: Sentimentos e PT-BR

- O laboratório exigirá analisar textos em português do Brasil. O modelo base (*default*) explodirá ou errará.
- Vocês precisarão explorar o Hub e usar a string explícita.

```python
from transformers import pipeline

# Pipeline com modelo exato do Hugging Face Hub (finetunned no
    Twitter-BR)
nlp = pipeline(
    "sentiment-analysis",
    model="citizenlab/twitter-xlm-roberta-base-sentiment-finetunned"
)

resultado = nlp("Eu simplesmente não suporto a lentidão deste app.")
```

Lab Parte 3: Extraindo Entidades (NER)

- A seguir, utilizaremos uma pipeline multilingue para varrer um texto corporativo extraindo nomes e locais.
- O parâmetro crucial aqui é agrupar tokens (`grouped_entities=True`). Sem isso, o BPE retornaria partes quebradas de um único nome.

```
ner = pipeline("ner", model="Davlan/bert-base-multilingual-cased-ner
    -hrl", aggregation_strategy="simple")

txt = "Aléssio trabalha na Universidade de Uberaba."
print(ner(txt))
# [{'entity_group': 'PER', 'word': 'Aléssio'},
#  {'entity_group': 'ORG', 'word': 'Universidade de Uberaba'}]
```

Lab Parte 4: Classificação Zero-Shot

- Para o desafio final, vocês receberão reclamações de clientes abertas e deverão triá-las em categorias que o modelo nunca viu durante o treinamento.

```
z_shot = pipeline("zero-shot-classification", model="facebook/bart-
    large-mnli")

ticket = "A bateria do meu notebook pifou."
categorias_ficticias = ["software", "hardware", "financeiro"]

res = z_shot(ticket, candidate_labels=categorias_ficticias)
print(res["labels"][0]) # "hardware"
```

Lab Parte 5: Submissão (O Fim do Arco)

- Passem no `pytest`, realizem o `git push`.
- Percebam o alívio que é usar *Transfer Learning*: no início do século, construir essas 3 classificações exigiria o trabalho de 30 engenheiros anotando dados por um semestre inteiro.
- O GitLab CI aprovará o código e vocês encerram oficialmente a **Era do NLP Classificatório/Analítico** e se preparam para a **Geração**.

Dúvidas?

Email: alessio@cefetmg.br
Aula: Segunda e Quarta-Feira, 7h as 8:40h (Sala 39)
WhatsApp: Dúvidas rápidas

Lembrete

Confira sua Issue no GitLab!

Próxima aula: APIs de LLMs & Prompts

100% da Aula 14 Concluída!