 Instituto Infnet	Avaliação	Nota:
		Visto do Professor:
MIT em Inteligência Artificial, Machine Learning e Deep Learning		
Nome	Mateus Teixeira Ramos da Silva	
Link do repositório	https://github.com/GitMateusTeixeira/03-ml-modeling/tree/main/06_infnet_llm_pd	
Matéria	IA generativa para linguagem (Large Language Model)	
Prazo	23.06.2025	

Parte 1. Fundamentos das LLMs

1. Explique os seguintes conceitos fundamentais dos LLMs, fornecendo exemplos práticos e diagramas onde for relevante:

1.1. Pre-training

R: O Pre-training (ou pré-treinamento) é a fase em que o modelo aprende uma representação geral da linguagem. Nesta etapa, o modelo é exposto a uma grande variedade de textos (livros, artigos, sites, fóruns, ...) e tenta prever palavras, frases ou tokens, com base no que já foi lido. Em outras palavras, ele aprende padrões, dependências linguísticas, gramática, contexto, estilo e até conhecimentos gerais – mas sem tarefas específicas.

Geralmente, o objetivo de aprendizado é auto-supervisionado, o que significa que os próprios dados geram os rótulos. Dois métodos comuns são: os Modelos Causais (como o GPT) e os Modelos Mascarados (como o BERT).

Os Modelos Causais aprendem a prever o próximo token com base em uma sequência anterior. Veja o exemplo:

Input: [A girafa come frutas da]

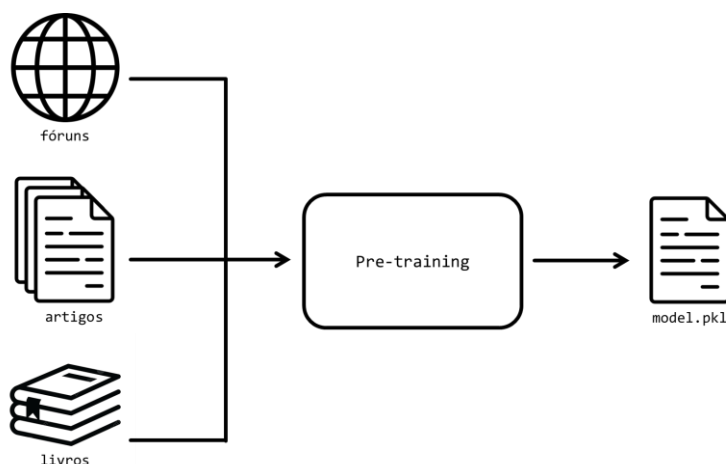
Target: 'árvore'

Já os Modelos Mascarados aprendem a prever tokens faltando (daí 'mascarados') dentro de um contexto. Como se percebe no exemplo:

Input: [O sol é uma [MASK] de energia]

Target: 'fonte'

Esses dois métodos têm papéis diferentes: enquanto os modelos causais são ideais para geração de texto (como diálogos ou resumos), os modelos mascarados são usados com mais eficiência em tarefas de compreensão e análise textual.

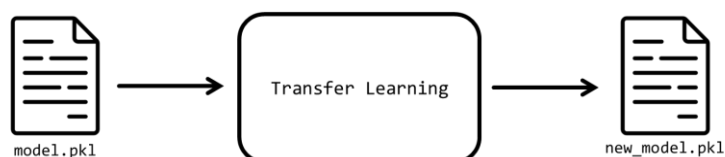


Assim, o pré-treinamento é fundamental para dar ao modelo uma “base linguística” ampla, que poderá ser posteriormente refinada por meio de fine-tuning em tarefas específicas, como tradução, classificação de sentimentos ou resposta a perguntas. Esse processo espelha, de certa forma, como, nós, humanos lemos diversos textos ao longo da vida e construímos nossa compreensão da linguagem antes de aplicá-la de maneira dirigida.

1.2. Transfer Learning

R: Transfer Learning (ou Aprendizado por Transferência) é o processo de reaproveitar o conhecimento aprendido durante o pré-treinamento de um modelo para uma nova tarefa específica. Em vez de treinar um modelo do zero, você usa um modelo já “inteligente” em linguagem para adaptá-lo a tarefas como tradução, resumo, classificação de sentimentos, etc.

As principais vantagens do Transfer Learning são: economia de tempo e computação (visto que não será necessário treinar o modelo do zero), necessidade reduzida de dados rotulados (visto que o modelo precisa de bem menos exemplos para entender e resolver novas tarefas), aproveitamento do conhecimento geral (já que o modelo aplica automaticamente fatos e raciocínio adquiridos em corpora) e facilidade para adaptar a domínios específicos (visto que o modelo já entende a base da linguagem).



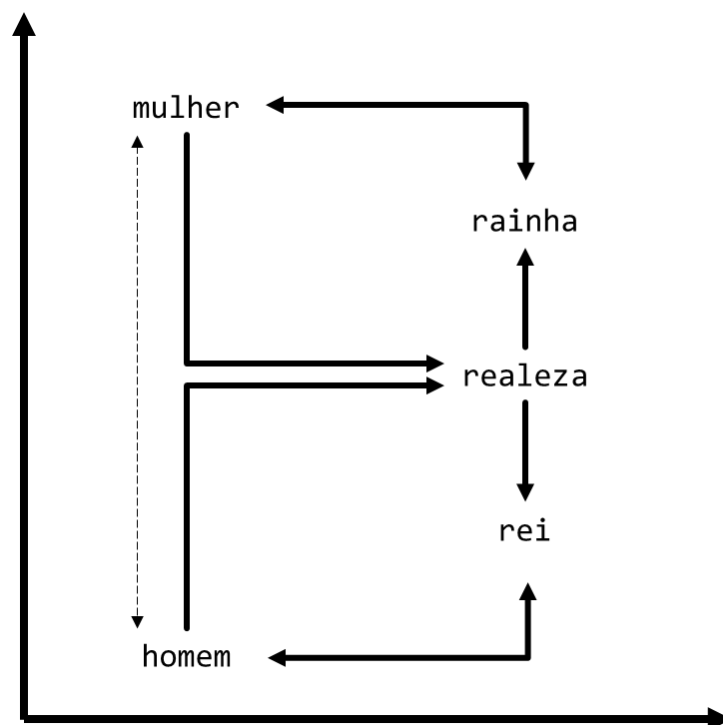
A título exemplificativo, podemos citar uma situação onde há um modelo de LLM que passou meses lendo livros, sites e artigos durante seu pré-treinamento. Agora, um escritório de advocacia deseja aplicá-lo para identificar se as decisões judiciais

favorecem o autor ou o réu. Basta adaptar esse modelo com um pequeno conjunto de exemplos anotados de decisões, e ele será capaz de generalizar essa tarefa com alguma precisão (resguardado, claro, as hipóteses de casos bem específicos) – sem a necessidade de começar do zero.

1.3. Embeddings

R: Embeddings são representações vetoriais densas de palavras, tokens ou frases, projetadas para capturar seus significados semânticos. Em vez de lidar com palavras como texto puro, os modelos as convertem em vetores numéricos de alta dimensão que preservam relações e contextos.

Imagine um mapa 3D onde há palavras semelhantes estão próximas (ex: "rei" e "rainha"), palavras opostas estão mais distantes (ex: "feliz" e "triste") e relações aparecem como direções vetoriais (ex: "rei" - "homem" + "mulher" \approx "rainha").



Como tipos de Embeddings podemos citar os Estáticos (como o Word2Vec, GloVe), onde cada palavra tem um valor fixo e os Contextuais (como o BERT e o GPT), onde o vetor da palavra muda dependendo da frase fornecida ("banco" em "banco de madeira" ou "banco financeiro").

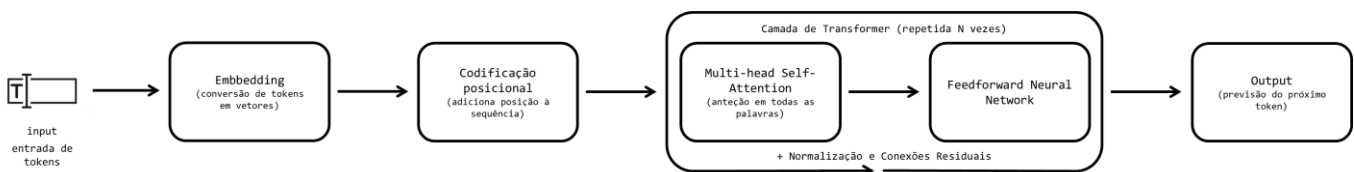
1.4. Transformers

R: No contexto da LLM, os Transformers são a arquitetura-base dos modelos de linguagem modernos, introduzida em 2017 pelo artigo “Attention is All You Need”.

Os modelos sequenciais antigos (como RNNs e LSTMs) foram substituídos por uma abordagem que permite analisar todas as palavras de uma frase simultaneamente, focando em quais são mais relevantes em cada contexto.

Isso acontece graças ao mecanismo de “self-attention”, que permite que cada palavra na entrada observe e atribua “atenção” a todas as outras, dependendo da tarefa – por exemplo, entender que em “o cachorro que correu”, o verbo “correu” está associado a “cachorro” e não a “o”.

A estrutura do Transformer inclui diversas camadas empilhadas compostas por atenção, redes neurais feedforward, normalização e conexões residuais. Como entrada, os tokens são transformados em vetores por camadas de embedding, que ainda recebem codificações de posição, já que o modelo não é sequencial. O que faz os Transformers tão poderosos é sua capacidade de capturar relações complexas em texto de maneira paralela e escalável.



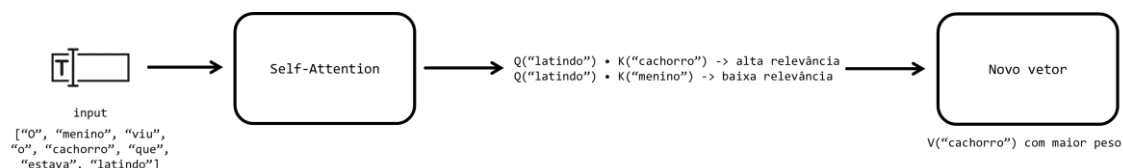
Isso permitiu o surgimento de LLMs como BERT, GPT, T5 e outros – todos baseados nessa arquitetura versátil, capaz de processar linguagem, código, imagens e muito mais, de forma contextualizada, rápida e altamente adaptável.

1.5. Attention

R: O mecanismo de attention permite que, ao processar uma palavra, o modelo “preste atenção” em outras partes da frase (ou do parágrafo inteiro), atribuindo pesos diferentes a cada uma, de acordo com sua relevância. Isso é o oposto de uma leitura sequencial cega: o modelo pode, por exemplo, entender que “ela” se refere a “Maria” no início do parágrafo, mesmo que estejam separadas por várias frases.

Internamente, a atenção funciona com três vetores por token: Query (Q), Key (K) e Value (V). A partir da similaridade entre Q e K, calcula-se um peso que é aplicado ao V correspondente. Isso permite que o modelo construa uma nova representação contextualizada daquele token, levando em conta os demais à sua volta.

Como exemplo, na frase: “O menino viu o cachorro que estava latindo.” Quando o modelo chega em “latindo”, ele usa atenção para determinar que está se referindo ao “cachorro” – e não ao “menino”. Isso é particularmente importante em frases ambíguas ou longas, onde a relação entre palavras não está explícita apenas pela ordem.



O attention também é a base do chamado self-attention, usado dentro da arquitetura dos Transformers, em que cada palavra presta atenção em todas as outras da mesma sequência. Há, ainda, o cross-attention, usado, por exemplo, para mapear frases de entrada e saída em tarefas como tradução automática.

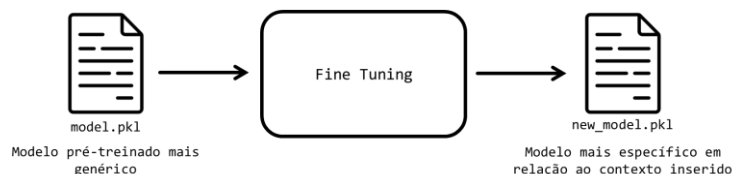
Um ponto interessante sobre esse mecanismo é que ele é totalmente diferenciável, o que significa que ele é aprendido durante o treinamento. O modelo descobre sozinho como organizar seus focos de atenção – sem regras definidas por humanos.

1.6. Fine-Tuning

R: O Fine-Tuning é a etapa em que um modelo de linguagem previamente treinado passa por um novo treinamento, agora com dados mais específicos, focado em uma tarefa concreta ou em um determinado domínio de conhecimento.

Em vez de começar o aprendizado do zero, o modelo já possui uma base sólida de compreensão da linguagem, e o fine-tuning serve para especializá-lo.

Por exemplo, podemos pegar um modelo geral e adaptá-lo para classificar sentimentos em tweets, responder dúvidas jurídicas ou atuar como assistente virtual em uma empresa. O processo envolve continuar o treinamento por algumas épocas, geralmente com uma taxa de aprendizado menor e em cima de um conjunto de dados menor e mais direcionado.



Essa adaptação pode ser feita de maneira mais leve com técnicas como o adapter tuning ou LoRA, que permitem treinar apenas pequenas partes adicionais do modelo.

No final, o modelo se torna muito mais eficaz para a tarefa em questão, ganhando vocabulário específico, estilo de escrita mais adequado ao contexto e uma compreensão mais refinada do seu novo objetivo. Em outras palavras, seria como transformar um generalista em um especialista – com tempo e dados bem escolhidos, pode-se obter ganhos bem consideráveis.

Parte 2. Quizzes do Curso de NLP da Hugging Face

2. Acesse os quizzes dos capítulos 1, 2 e 3 do curso de NLP da Hugging Face através do link: [Curso de NLP](#).

R: Requisitos salvos na pasta: https://github.com/GitMateusTeixeira/03-ml-modeling/tree/main/06_infnet_llm_pd/02_quizzes_hugging_face

2.1. Resolva os quizzes e capture screenshots dos resultados.

R: Seguem as capturas de tela do Quiz do Capítulo 1:

1. Explore the Hub and look for the roberta-large-mnli checkpoint. What task does it perform?

- ☐ Summarization
- ☒ Text classification

Correct! More precisely, it classifies if two sentences are logically linked across three labels (contradiction, neutral, entailment) — a task also called *natural language inference*.

- ☐ Text generation

Submit You got all the answers!

2. What will the following code return?

```
from transformers import pipeline

ner = pipeline("ner", grouped_entities=True)
ner("My name is Sylvain and I work at Hugging Face in Brooklyn.")
```

- ☐ It will return classification scores for this sentence, with labels "positive" or "negative".
- ☐ It will return a generated text completing this sentence.
- ☒ It will return the words representing persons, organizations or locations.

Correct! Furthermore, with `grouped_entities=True`, it will group together the words belonging to the same entity, like "Hugging Face".

Submit You got all the answers!

3. What should replace ... in this code sample?

```
from transformers import pipeline

filler = pipeline("fill-mask", model="bert-base-cased")
result = filler("...")
```

- ☐ This <mask> has been waiting for you.
- ☒ This [MASK] has been waiting for you.

Correct! This model's mask token is [MASK].

- ☐ This man has been waiting for you.

Submit You got all the answers!

4. Why will this code fail?

```
from transformers import pipeline

classifier = pipeline("zero-shot-classification")
result = classifier("This is a course about the Transformers library")
```

- ☒ This pipeline requires that labels be given to classify this text.

Correct! Right — the correct code needs to include `candidate_labels=[...]`.

- ☐ This pipeline requires several sentences, not just one.
- ☐ The 🤖 Transformers library is broken, as usual.
- ☐ This pipeline requires longer inputs; this one is too short.

Submit You got all the answers!

5. What does “transfer learning” mean?

- ☐ Transferring the knowledge of a pretrained model to a new model by training it on the same dataset.
- ☒ Transferring the knowledge of a pretrained model to a new model by initializing the second model with the first model's weights.

Correct! When the second model is trained on a new task, it *transfers* the knowledge of the first model.

- ☐ Transferring the knowledge of a pretrained model to a new model by building the second model with the same architecture as the first model.

Submit

You got all the answers!

7. Select the sentence that best describes the terms “model”, “architecture”, and “weights”.

- ☐ If a model is a building, its architecture is the blueprint and the weights are the people living inside.
- ☐ An architecture is a map to build a model and its weights are the cities represented on the map.
- ☒ An architecture is a succession of mathematical functions to build a model and its weights are those functions parameters.

Correct! The same set of mathematical functions (architecture) can be used to build different models by using different parameters (weights).

Submit

You got all the answers!

9. Which of those types of models would you use for summarizing texts?

- ☐ An encoder model
- ☐ A decoder model
- ☒ A sequence-to-sequence model

Correct! Sequence-to-sequence models are perfectly suited for a summarization task.

Submit

You got all the answers!

11. What possible source can the bias observed in a model have?

- ☒ The model is a fine-tuned version of a pretrained model and it picked up its bias from it.
- ☒ The data the model was trained on is biased.
- ☒ The metric the model was optimizing for is biased.

Correct! A less obvious source of bias is the way the model is trained. Your model will blindly optimize for whatever metric you chose, without any second thoughts.

Submit

You got all the answers!

6. True or false? A language model usually does not need labels for its pretraining.

- ☒ True

Correct! The pretraining is usually *self-supervised*, which means the labels are created automatically from the inputs (like predicting the next word or filling in some masked words).

- ☐ False

Submit

You got all the answers!

8. Which of these types of models would you use for completing prompts with generated text?

- ☐ An encoder model
- ☒ A decoder model

Correct! Decoder models are perfectly suited for text generation from a prompt.

- ☐ A sequence-to-sequence model

Submit

You got all the answers!

10. Which of these types of models would you use for classifying text inputs according to certain labels?

- ☒ An encoder model

Correct! An encoder model generates a representation of the whole sentence which is perfectly suited for a task like classification.

- ☐ A decoder model
- ☐ A sequence-to-sequence model

Submit

You got all the answers!

Seguem as capturas de tela do Quiz do Capítulo 2:

1. What is the order of the language modeling pipeline?

- ☐ First, the model, which handles text and returns raw predictions. The tokenizer then makes sense of these predictions and converts them back to text when needed.
- ☐ First, the tokenizer, which handles text and returns IDs. The model handles these IDs and outputs a prediction, which can be some text.
- ☒ The tokenizer handles text and returns IDs. The model handles these IDs and outputs a prediction. The tokenizer can then be used once again to convert these predictions back to some text.

Correct! Correct! The tokenizer can be used for both tokenizing and de-tokenizing.

Submit You got all the answers!

3. Which of the following is an example of subword tokenization?

- ☒ WordPiece

Correct! Yes, that's one example of subword tokenization!

- ☐ Character-based tokenization
- ☐ Splitting on whitespace and punctuation
- ☒ BPE

Correct! Yes, that's one example of subword tokenization!

- ☒ Unigram

Correct! Yes, that's one example of subword tokenization!

- ☐ None of the above

Submit You got all the answers!

5. What is an AutoModel?

- ☐ A model that automatically trains on your data
- ☒ An object that returns the correct architecture based on the checkpoint

Correct! Exactly: the `AutoModel` only needs to know the checkpoint from which to initialize to return the correct architecture.

- ☐ A model that automatically detects the language used for its inputs to load the correct weights

Submit You got all the answers!

7. What is the point of applying a SoftMax function to the logits output by a sequence classification model?

- ☐ It softens the logits so that they're more reliable.
- ☒ It applies a lower and upper bound so that they're understandable.

Correct! Correct! The resulting values are bound between 0 and 1. That's not the only reason we use a SoftMax function, though.

- ☒ The total sum of the output is then 1, resulting in a possible probabilistic interpretation.

Correct! Correct! That's not the only reason we use a SoftMax function, though.

Submit You got all the answers!

2. How many dimensions does the tensor output by the base Transformer model have, and what are they?

- ☐ 2: The sequence length and the batch size
- ☐ 2: The sequence length and the hidden size
- ☒ 3: The sequence length, the batch size, and the hidden size

Correct! Correct!

Submit You got all the answers!

4. What is a model head?

- ☐ A component of the base Transformer network that redirects tensors to their correct layers
- ☐ Also known as the self-attention mechanism, it adapts the representation of a token according to the other tokens of the sequence
- ☒ An additional component, usually made up of one or a few layers, to convert the transformer predictions to a task-specific output

Correct! That's right. Adaptation heads, also known simply as heads, come up in different forms: language modeling heads, question answering heads, sequence classification heads...

Submit You got all the answers!

6. What are the techniques to be aware of when batching sequences of different lengths together?

- ☒ Truncating

Correct! Yes, truncation is a correct way of evening out sequences so that they fit in a rectangular shape. Is it the only one, though?

- ☐ Returning tensors
- ☒ Padding

Correct! Yes, padding is a correct way of evening out sequences so that they fit in a rectangular shape. Is it the only one, though?

- ☒ Attention masking

Correct! Absolutely! Attention masks are of prime importance when handling sequences of different lengths. That's not the only technique to be aware of, however.

Submit You got all the answers!

8. What method is most of the tokenizer API centered around?

- ☐ `encode`, as it can encode text into IDs and IDs into predictions
- ☒ Calling the tokenizer object directly.

Correct! Exactly! The `__call__` method of the tokenizer is a very powerful method which can handle pretty much anything. It is also the method used to retrieve predictions from a model.

- ☐ `pad`
- ☐ `tokenize`

Submit You got all the answers!

9. What does the result variable contain in this code sample?

```
from transformers import AutoTokenizer

tokenizer = AutoTokenizer.from_pretrained("bert-base-cased")
result = tokenizer.tokenize("Hello!")
```

☒ A list of strings, each string being a token

Correct! Absolutely! Convert this to IDs, and send them to a model!

☐ A list of IDs

☐ A string containing all of the tokens

Submit You got all the answers!

10. Is there something wrong with the following code?

```
from transformers import AutoTokenizer, AutoModel

tokenizer = AutoTokenizer.from_pretrained("bert-base-cased")
model = AutoModel.from_pretrained("gpt2")

encoded = tokenizer("Hey!", return_tensors="pt")
result = model(**encoded)
```

☐ No, it seems correct.

☒ The tokenizer and model should always be from the same checkpoint.

Correct! Right!

☐ It's good practice to pad and truncate with the tokenizer as every input is a batch.

Submit You got all the answers!

Seguem as capturas de tela do Quiz do Capítulo 3:

1. The emotion dataset contains Twitter messages labeled with emotions. Search for it in the Hub , and read the dataset card. Which of these is not one of its basic emotions?

☐ Joy

☐ Love

☒ Confusion

Correct! Correct! Confusion is not one of the six basic emotions.

☐ Surprise

Submit You got all the answers!

2. Search for the ar_sarcasm dataset in the Hub . Which task does it support?

☒ Sentiment classification

Correct! That's right! You can tell thanks to the tags.

☐ Machine translation

☐ Named entity recognition

☐ Question answering

Submit You got all the answers!

3. How does the BERT model expect a pair of sentences to be processed?

☐ Tokens_of_sentence_1 [SEP] Tokens_of_sentence_2

☐ [CLS] Tokens_of_sentence_1 Tokens_of_sentence_2

☒ [CLS] Tokens_of_sentence_1 [SEP] Tokens_of_sentence_2 [SEP]

Correct! That's correct!

☐ [CLS] Tokens_of_sentence_1 [SEP] Tokens_of_sentence_2

Submit You got all the answers!

4. What are the benefits of the Dataset.map() method?

☒ The results of the function are cached, so it won't take any time if we re-execute the code.

Correct! That is indeed one of the neat benefits of this method! It's not the only one, though...

☒ It can apply multiprocessing to go faster than applying the function on each element of the dataset.

Correct! This is a neat feature of this method, but it's not the only one!

☒ It does not load the whole dataset into memory, saving the results as soon as one element is processed.

Correct! That's one advantage of this method. There are others, though!

Submit You got all the answers!

5. What does dynamic padding mean?

☐ It's when you pad the inputs for each batch to the maximum length in the whole dataset.

☒ It's when you pad your inputs when the batch is created, to the maximum length of the sentences inside that batch.

Correct! That's correct! The "dynamic" part comes from the fact that the size of each batch is determined at the time of creation, and all your batches might have different shapes as a result.

☐ It's when you pad your inputs so that each sentence has the same number of tokens as the previous one in the dataset.

Submit You got all the answers!

6. What is the purpose of a collate function?

☐ It ensures all the sequences in the dataset have the same length.

☒ It puts together all the samples in a batch.

Correct! Correct! You can pass the collate function as an argument of a `DataLoader`. We used the `DataCollatorWithPadding` function, which pads all items in a batch so they have the same length.

☐ It preprocesses the whole dataset.

☐ It truncates the sequences in the dataset.

Submit You got all the answers!

7. What happens when you instantiate one of the `AutoModelForXxx` classes with a pretrained language model (such as `bert-base-uncased`) that corresponds to a different task than the one for which it was trained?

☐ Nothing, but you get a warning.

☒ The head of the pretrained model is discarded and a new head suitable for the task is inserted instead.

Correct! Correct. For example, when we used `AutoModelForSequenceClassification` with `bert-base-uncased`, we got warnings when instantiating the model. The pretrained head is not used for the sequence classification task, so it's discarded and a new head is instantiated with random weights.

☐ The head of the pretrained model is discarded.

Nothing, since the model can still be fine-tuned for the different task.

Submit

You got all the answers!

9. Why should you use the 🚀 `Accelerate` library?

☐ It provides access to faster models.

☐ It provides a high-level API so I don't have to implement my own training loop.

☒ It makes our training loops work on distributed strategies.

Correct! Correct! With 🚀 `Accelerate`, your training loops will work for multiple GPUs and TPUs.

☐ It provides more optimization functions.

Submit

You got all the answers!

8. What's the purpose of `TrainingArguments` ?

☒ It contains all the hyperparameters used for training and evaluation with the `Trainer`.

Correct! Correct!

☐ It specifies the size of the model.

☐ It just contains the hyperparameters used for evaluation.

☐ It just contains the hyperparameters used for training.

Submit

You got all the answers!

2.2. Anexe as screenshots a esta avaliação e explique brevemente os conceitos abordados em cada quiz.

R: Durante os quizzes, foram abordados diversos conceitos fundamentais para trabalhar com modelos de linguagem baseados em Transformers.

Começou-se pela estrutura básica do pipeline: tokenização do texto original em IDs numéricos, processamento desses IDs pelo modelo e conversão da saída novamente em texto.

Discutiram-se também as dimensões do tensor de saída (batch size, sequence length e hidden size) e as técnicas de tokenização por subpalavras (como `WordPiece` e `BPE`), que ajudam a lidar com vocabulários extensos.

Destacou-se ainda o papel das model heads, que adaptam a saída do modelo para tarefas como classificação ou geração, e o uso da função `SoftMax` para transformar logits em probabilidades interpretáveis.

Na prática com a biblioteca Hugging Face, foi enfatizado o uso direto do tokenizer, a importância de manter compatibilidade entre tokenizer e modelo, e o formato correto de entrada para modelos como o BERT, especialmente ao lidar com pares de sentenças.

Discutiram-se também os datasets “emotion” e “ar_sarcasm”, usados em tarefas de detecção de emoções e sentimentos. Ferramentas como `map()` e `dynamic padding` foram mencionadas por sua eficiência no pré-processamento de dados, assim como o uso de funções de colagem (`collate functions`) para montagem de lotes durante o treinamento.

Por fim, explorou-se a lógica por trás da personalização de modelos com diferentes heads usando classes como `AutoModelForXxx`, e a centralização de parâmetros com `TrainingArguments`.

Destacou-se ainda a biblioteca `Accelerate`, que permite rodar loops de treinamento em múltiplos dispositivos com simplicidade, sem necessidade de reescrever o código.

De modo geral, os quizzes abordaram desde fundamentos teóricos até práticas avançadas para implementar e treinar modelos de linguagem com eficiência e clareza.

Parte 3. Análise de Dados com NER

R: Parte atendida na pasta: https://github.com/GitMateusTeixeira/03-ml-modeling/tree/main/06_infnet_llm_pd/03_analise_de_dados_com_ner

3. Baixe o conjunto de dados de notícias disponível em: Folha UOL News Dataset.

R: Os dados foram baixados na célula ‘2’ do arquivo ‘ner.ipynb’:

Parte 1. Download dos dados

[Voltar ao início](#)

```
# baixar os dados com a api do Kaggle
kaggle.api.dataset_download_files('marlesson/news-of-the-site-folhauol',
                                  path='../data/01-raw/',
                                  unzip=True)
```

[2] ✓ 10.4s

Python

... Dataset URL: <https://www.kaggle.com/datasets/marlesson/news-of-the-site-folhauol>

3.1. Utilize o modelo 'monilouise/ner_pt_br' para identificar e extrair entidades mencionadas nas notícias.

R: O modelo foi carregado na célula '14' no arquivo 'ner.ipynb':

Parte 3. Tokenização

[Voltar ao início](#)

```
# carregar modelo e tokenizer
model_name = "monilouise/ner_pt_br"
tokenizer = AutoTokenizer.from_pretrained("neuralmind/bert-base-portuguese-cased")
model = AutoModelForTokenClassification.from_pretrained(model_name)
```

[14] ✓ 2.4s

Python

... Some weights of the model checkpoint at monilouise/ner_pt_br were not used when initializing BertForTokenClassification: ['bert.pooler.dense.bias',
- This IS expected if you are initializing BertForTokenClassification from the checkpoint of a model trained on another task or with another architec
- This IS NOT expected if you are initializing BertForTokenClassification from the checkpoint of a model that you expect to be exactly identical (i

As entidades foram extraídas nas células '15' a '18' do arquivo 'ner.ipynb':

```
# criar pipeline de NER
ner_pipeline = pipeline("ner", model=model, tokenizer=tokenizer, grouped_entities=True)
```

[15] ✓ 0.0s

Python

... Device set to use cpu

Parte 4. NER

[Voltar ao início](#)

```
# aplicar NER nos textos
def extrair_entidades(texto):
    try:
        return ner_pipeline(texto)
    except:
        return []
```

[17] ✓ 0.0s

Python

```
# aplicar na coluna de texto
df['entidades'] = df['text'].progress_apply(extrair_entidades)
```

[18] ✓ 10m 39.0s

Python

... 0%| | 0/2111 [00:00<?, ?it/s]Asking to truncate to max_length but no maximum length is provided and the model has no predefined maximum
100%|██████████| 2111/2111 [10:38<00:00, 3.30it/s]

3.2. Crie um ranking das organizações que mais apareceram na seção "Mercado" no primeiro trimestre de 2015.

R: O dataframe foi filtrado nas células '7' a '12':

```
[7] ✓ 0.0s # converter a coluna 'date' para datetime para filtrar posteriormente
df['date'] = pd.to_datetime(df['date'], errors='coerce')

[8] ✓ 0.2s # filtrar a coluna 'category' por 'Mercado'
df = df[df['category'].str.lower() == 'mercado']

[9] ✓ 0.0s # verificar o tamanho após o primeiro filtro
df.shape

... (20970, 6)
```

```
[11] ✓ 0.0s # aplicar o segundo filtro. filtrar as notícias do 1º trimestre de 2015
start = datetime(2015, 1, 1)
end = datetime(2015, 3, 31)
df = df[(df['date'] >= start) & (df['date'] <= end)]

[12] ✓ 0.0s # verificar o tamanho após o segundo filtro
df.shape

... (2111, 6)
```

Após isso, foi retirado as organizações do dataframe nas células '22' e '23':

Parte 4.1. Extrair as organizações de cada texto

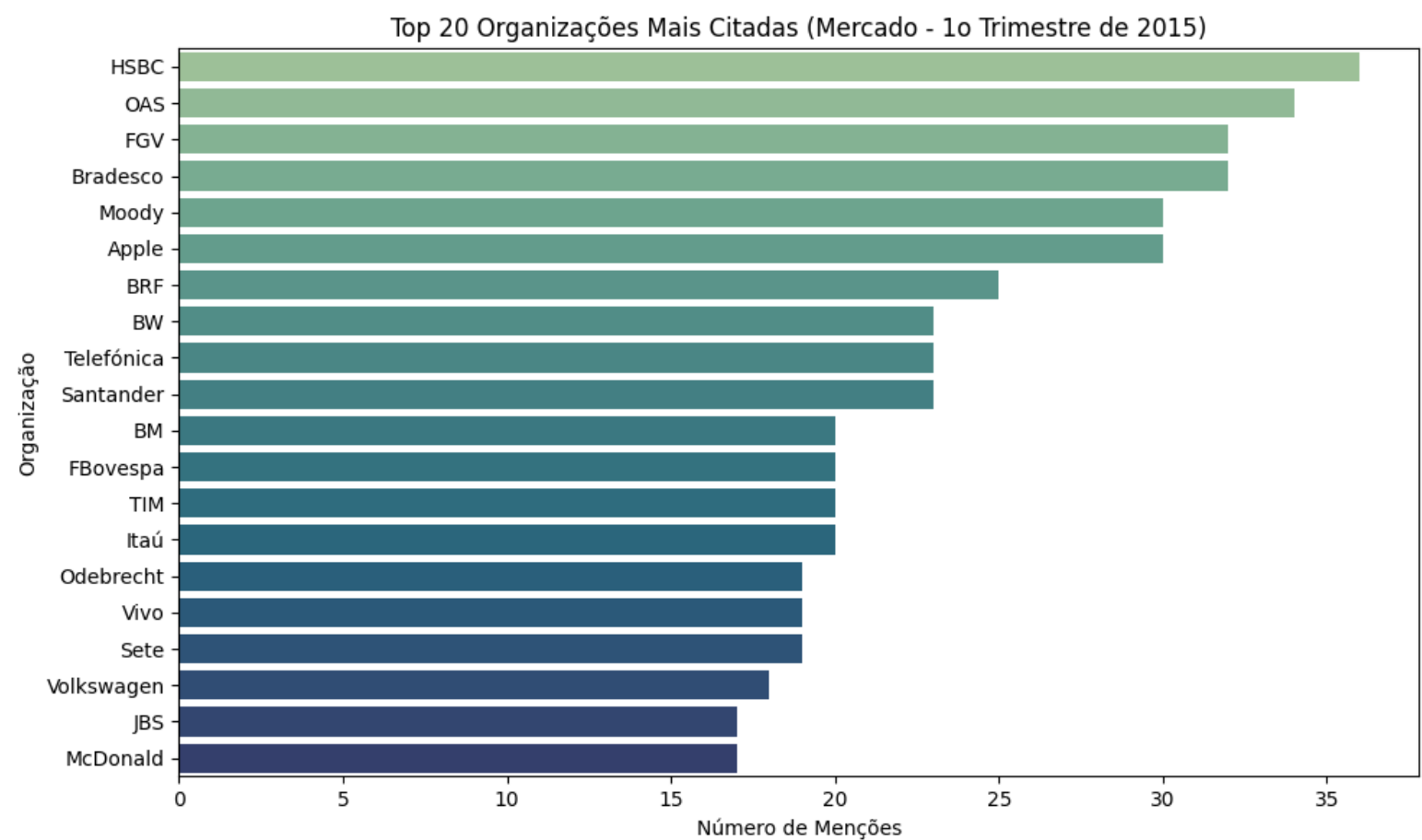
[Voltar ao início](#)

```
[22] ✓ 0.0s # extrair apenas as organizações de cada texto
def extrair_organizacoes(entidades):
    return [ent['word'] for ent in entidades if 'ORG' in ent['entity_group'].upper()]

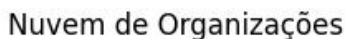
[23] ✓ 0.0s # aplicar a função
df['organizacoes'] = df['entidades'].progress_apply(extrair_organizacoes)

... 100%|██████████| 2111/2111 [00:00<00:00, 46470.32it/s]
```

Após alguns tratamentos, foi exibido o gráfico com as empresas que mais aparecem nas notícias em questão. O gráfico foi plotado na célula '31' do arquivo 'ner.ipynb':



Na célula ‘47’ foi plotada uma Nuvem de Organizações que mostram as principais empresas citadas no corpus:



3.3. Apresente os resultados em um relatório detalhado, incluindo a metodologia utilizada e visualizações para apoiar a análise.

R: Segundo a Nuvem de Palavras plotada na célula '45', as notícias tratam sobre tema de investimentos em empresas no Brasil:

Nuvem de Palavras



A nuvem de palavras foi plotada após a lematização e tratamento de palavras sem valor gramatical e remoção de preposições com `<token.pos != 'ADP'>` :

```
# lematizar o texto
def lematizar_texto(texto):
    doc = nlp(texto)
    lemas = [
        token.lemma_.lower()
        for token in doc
        if not token.is_punct
        and not token.is_space
        and token.pos_ != 'ADP' # remover preposições
        and token.lemma_.lower() not in termos_excluir
    ]
    return ' '.join(lemas)
```

[42]



0.0s

Python

```
# aplicar a lematização
df['texto_lematizado'] = df['text'].fillna('').progress_apply(lematizar_texto)
```

[43]



3m 1.9s

Python

... 100%| 2111/2111 [03:01<00:00, 11.61it/s]

Para exibir as organizações foi excluído alguns termos que não possuem muita relevância como 'Folha' (eis que é o nome do Jornal) e 'Brasil'. Desse modo, pode-se observar melhor o tema principal do corpus analisados e quais as empresas mais citadas, como HSBC, FGV e Apple.

Parte 4. Engenharia de Prompts

4. Analise os seguintes prompts e identifique por que eles poderiam gerar respostas insatisfatórias ou irrelevantes:

Exemplo 1: "Escreva sobre cachorros."

Exemplo 2: "Explique física."

R: O primeiro prompt é vago e excessivamente amplo. Cachorros podem ser abordados sob diversos aspectos: comportamento, raças, alimentação, saúde, papel histórico, interação com humanos, entre outros. Ao não delimitar um foco, o modelo de linguagem pode gerar uma resposta genérica, superficial ou abordar aspectos que não correspondem à expectativa do usuário. A falta de contexto, público-alvo e objetivo textual dificulta uma resposta realmente útil.

O segundo prompt sofre do mesmo problema: é extremamente genérico. Física é uma ciência com múltiplas ramificações — como mecânica, termodinâmica, óptica, eletromagnetismo, relatividade ou física quântica. Sem direcionamento, o modelo pode abordar qualquer uma dessas áreas de forma aleatória ou genérica, o que tende a gerar um conteúdo desorganizado ou com baixa profundidade.

Em ambos os casos, a ausência de escopo, detalhamento e contexto compromete a qualidade da resposta.

4.1. Reformule cada prompt utilizando técnicas de engenharia de prompts para torná-los mais específicos e direcionados.

R: Os prompts apresentados revelam-se excessivamente amplos e carentes de direcionamento, o que tende a resultar em respostas genéricas ou desconectadas do objetivo pretendido. Para aumentar sua eficácia, é recomendável aplicar técnicas de engenharia de prompts que incluam delimitação de escopo, definição de público-alvo, indicação do formato textual e, sempre que possível, a inclusão de exemplos.

No caso do primeiro prompt, pode-se reformulá-lo da seguinte maneira:

“Escrever um texto explicativo de aproximadamente 300 palavras comparando o comportamento de cães das raças Border Collie e Golden Retriever, voltado para pessoas que consideram adotar um cão de companhia ativo.”

Com essa reformulação, observa-se a definição clara do foco (comparação entre raças específicas), o tipo textual desejado (explicativo), o tamanho estimado da resposta e o perfil do público-alvo (futuros adotantes), o que contribui para a geração de um conteúdo mais direcionado e relevante.

Já no segundo exemplo, recomenda-se a seguinte reformulação:

“Explicar o conceito de entropia dentro da termodinâmica clássica, utilizando exemplos cotidianos como o derretimento de gelo ou a desorganização de um quarto.”

Essa versão guia o modelo a tratar de um tema específico dentro de uma área ampla da física, ao mesmo tempo em que solicita analogias do cotidiano que favorecem a compreensão de públicos não especializados.

Essas reformulações demonstram como o aprimoramento do enunciado pode aumentar significativamente a clareza, utilidade e qualidade das respostas geradas por modelos de linguagem. Ao tornar mais explícito se espera obter um resultado mais preciso, aplicável e alinhado aos objetivos de quem formula o prompt.

4.2. Explique as melhorias feitas em cada caso e os motivos por trás das reformulações.

R: No primeiro caso, a reformulação do prompt introduz uma delimitação temática clara, o que contribui para restringir o escopo e orientar a geração do conteúdo. Ao especificar o tipo de texto (explicativo), o tema (comparação comportamental entre duas raças de cães) e o público-alvo (pessoas interessadas em adoção), permite-se ao modelo adaptar a linguagem, o nível de profundidade e os exemplos utilizados, tornando a resposta mais assertiva.

No segundo caso, a melhoria está na precisão conceitual e na contextualização prática. Ao selecionar um conceito específico – entropia – e inseri-lo dentro de uma subárea da física (termodinâmica clássica), garante-se foco e relevância. A exigência por exemplos cotidianos ainda amplia a acessibilidade da explicação, favorecendo a compreensão por parte de leitores com pouca familiaridade com o tema.

Em ambos os casos, o refinamento do prompt reduz a ambiguidade, aumenta a coerência temática e permite que o modelo utilize melhor seus recursos linguísticos e informativos, o que se traduz em respostas mais consistentes, contextualizadas e com maior valor para quem consulta.

5. O prompt "Descreva a história da internet." foi mal formulado. Aplique técnicas de engenharia de prompts para melhorá-lo. Reformule o prompt para melhorar a especificidade e a qualidade da resposta. Justifique as mudanças feitas e explique como elas contribuem para obter uma resposta mais eficaz e relevante.

R: O prompt descrito apresenta um escopo muito amplo e indefinido, o que pode resultar em respostas genéricas, desorganizadas ou excessivamente longas, dificultando a compreensão e o aproveitamento do conteúdo. Para torná-lo mais eficiente, aplicam-se técnicas de engenharia de prompts que visam delimitar o tema, ajustar a linguagem ao público-alvo e definir o formato e a profundidade da resposta.

No caso em tela, propõe-se a seguinte reformulação:

“Elabore uma linha do tempo concisa, com até 10 eventos marcantes, que sintetize a evolução da internet desde os experimentos com ARPANET até a popularização da Web 2.0, voltada para estudantes do ensino médio”

Essa abordagem oferece direcionamento claro quanto ao conteúdo esperado. Além disso, estabelece um recorte histórico específico e sugere um formato organizado, ao mesmo tempo em que adapta o grau de complexidade da resposta ao perfil do leitor.

Com isso, facilita-se a geração de uma resposta objetiva, bem estruturada e mais útil, ao mesmo tempo em que se evita dispersão e ambiguidade. Essa abordagem permite que o modelo de linguagem concentre-se nos aspectos mais relevantes, entregando uma explicação coerente, educativa e acessível.

Aplicar esse tipo de refinamento no enunciado é uma boa prática para quem deseja obter respostas mais eficazes e alinhadas às suas intenções informacionais. A mesma lógica pode ser replicada para outros temas amplos, como a história da computação ou a evolução da imprensa, sempre com ganhos de clareza, foco e aplicabilidade.

6. Aplique a técnica de Chain of Thought (CoT) para melhorar o prompt "Explique como funciona a energia solar.", detalhando o raciocínio necessário para que o modelo forneça uma resposta completa e coerente. Explique como a aplicação da técnica CoT melhora a resposta do modelo.

R: A técnica conhecida como Chain of Thought (CoT), ou “cadeia de raciocínio”, é uma abordagem que orienta o modelo de linguagem a construir sua resposta por meio de um encadeamento lógico de ideias, passo a passo.

Em vez de gerar uma explicação direta e muitas vezes superficial, o modelo é estimulado a “pensar em voz alta”, organizando o caminho entre a pergunta e a resposta final de maneira estruturada. Essa estratégia é particularmente eficaz

quando se lida com temas técnicos, complexos ou compostos por várias etapas interdependentes – como é o caso do funcionamento da energia solar.

O prompt original “Explique como funciona a energia solar” é excessivamente amplo e não delimita os aspectos mais importantes a serem abordados, o que pode fazer com que o modelo forneça respostas vagas, incompletas ou centradas em apenas uma parte do processo.

Ao aplicar a técnica de CoT, pode-se reformular esse prompt para algo mais orientado, como:

“Explique como funciona a energia solar descrevendo, primeiro, como a radiação solar é captada pelos painéis fotovoltaicos; em seguida, como essa energia é convertida em eletricidade; e por fim, como essa eletricidade é utilizada ou armazenada em uma residência.”

Essa estrutura conduz o modelo por uma sequência lógica, promovendo um raciocínio em camadas: da fonte energética à aplicação prática.

Ao seguir essa cadeia, o modelo é incentivado a explorar todas as etapas relevantes do processo de forma coerente, o que resulta em uma explicação mais detalhada, fluida e compreensível. A aplicação do CoT, portanto, melhora significativamente a qualidade da resposta, garantindo que ela seja não apenas informativa, mas também pedagogicamente eficaz, conduzindo o leitor por uma jornada de entendimento passo a passo. Essa técnica é um recurso poderoso para quem deseja obter respostas mais precisas, completas e alinhadas ao raciocínio humano.

Parte 5. Projeto Prático com Streamlit, LLM e LangChain

7. Escolha uma aplicação para desenvolver utilizando Streamlit, LLM e LangChain. Crie um aplicativo interativo que demonstre o uso de LLMs para resolver um problema específico.

Exemplos de Aplicação:

- **Sumarizador de Artigos:** Desenvolva um aplicativo que permita ao usuário inserir o texto de um artigo e obter um resumo conciso do conteúdo.
- **Sistema de Perguntas e Respostas:** Crie um sistema que permita ao usuário fazer perguntas sobre um tópico específico e receba respostas precisas e relevantes.
- **Agente de Viagem:** Desenvolva um agente virtual que possa ajudar usuários a planejar suas viagens, fornecendo informações sobre destinos, itinerários, e dicas de viagem.

- **App de Auxílio em Aprendizagem:** Crie um aplicativo que auxilie estudantes a aprender um novo assunto, fornecendo explicações, exemplos e quizzes interativos.

R: Requisitos atendidos na pasta: https://github.com/GitMateusTeixeira/03-ml-modeling/tree/main/06_infnet_llm_pd/05_perguntas_cdc

7.1. Descreva a aplicação escolhida e os objetivos principais do projeto.

R: Foi escolhido um aplicativo de perguntas e respostas focada no Direito do Consumidor. A aplicação (construída com o Streamlit) possui um layout simples e confortável para o usuário.

The screenshot displays the 'Specter - Assistente Jurídico (AIJ)' web application. The interface is dark-themed. On the left, a sidebar contains the title 'Specter - Assistente Jurídico' with a scales icon, a description 'Converse com o Specter, assistente de IA especializado em Direito do Consumidor.', and a section 'Temas sugeridos:' listing topics like 'Direito de arrendimento', 'Responsabilidade pelo produto', 'Publicidade enganosa', and 'Garantia e assistência técnica'. Below this is a 'Conceitos Básicos' section with examples and buttons for 'Direitos básicos do consumidor' and 'Publicidade enganosa'. The main area features the title 'Specter - Assistente Inteligente Jurídico (AIJ)' and a 'Deploy' button. It includes a 'Nova Conversa' button and a chat history entry with a question about consumer rights and an AI response explaining the Consumer Defense Code (Lei nº 8.078/90) and listing rights to information and withdrawal. At the bottom is a text input field with a placeholder 'Digite sua dúvida jurídica aqui...' and a submit arrow.

Specter - Assistente Jurídico

Converse com o Specter, assistente de IA especializado em Direito do Consumidor.

Temas sugeridos:

- Direito de arrendimento
- Responsabilidade pelo produto
- Publicidade enganosa
- Garantia e assistência técnica

Conceitos Básicos

Exemplos sobre conceitos fundamentais sobre o Direito do Consumidor.

Direitos básicos do consumidor

Publicidade enganosa

Specter - Assistente Inteligente Jurídico (AIJ)

Deploy

Converse com o assistente de IA sobre temas jurídicos, com base em documentos técnicos confiáveis.

Nova Conversa

Quais são os direitos básicos do consumidor, como o direito à informação, ao arrendimento, à proteção contra práticas abusivas, etc.?

Excelente pergunta!

De acordo com o Código de Defesa do Consumidor (Lei nº 8.078/90), os direitos básicos do consumidor incluem:

1. **Direito à Informação:** O consumidor tem direito a receber informações claras, precisas e ostensivas sobre os produtos e serviços que adquire, incluindo características, qualidade, preço, garantia, prazo de entrega e outros dados relevantes (Art. 6º, inciso III, do CDC).
2. **Direito ao Arrendimento:** O consumidor tem direito a desistir do contrato ou do serviço, no prazo de 7 dias, contados a partir da data de recebimento do produto ou do serviço, sem


Digite sua dúvida jurídica aqui...


Specter – Assistente Jurídico


Converse com o **Specter**, assistente de IA especializado em Direito do Consumidor.

Temas sugeridos:

- Direito de arrependimento
- Responsabilidade pelo produto
- Publicidade enganosa
- Garantia e assistência técnica

 Conceitos Básicos

 Principais dúvidas

 Fontes e Estudo


Adicionar Documento

Envie um arquivo (PDF, CSV, Excel)

Drag and drop file here

Limit 200MB per file • PDF, CSV, XLSX


Browse files

 Exportar Histórico


Feito por



Mateus Teixeira
Cientista de Dados
Pós-graduando em Inteligência Artificial pela INFNET

 GMAIL

 LINKEDIN

 INSTAGRAM

Foi implementada algumas funcionalidades na ‘sidebar’ como temas sugeridos, algumas pergunta-modelo básicas para teste.

temas sugeridos:

- Direito de arrependimento
- Responsabilidade pelo produto
- Publicidade enganosa
- Garantia e assistência técnica

Conceitos Básicos

Exemplos sobre conceitos fundamentais sobre o Direito do Consumidor.

Direitos básicos do consumidor

Publicidade enganosa

Prazos de garantia

Principais dúvidas

Specter - Assistente Inteligente Jurídico (AIJ)

Deploy

Converse com o assistente de IA sobre temas jurídicos, com base em documentos técnicos confiáveis.

Nova Conversa

Quais são os prazos de garantia para os produtos e serviços e quais são as implicações para o consumidor?

Boa pergunta!

De acordo com o Código de Defesa do Consumidor (CDC), Lei nº 8.078/90, os prazos de garantia para produtos e serviços são os seguintes:

- Prazo de garantia legal: 90 dias para produtos e serviços não duráveis e 30 dias para produtos e serviços duráveis. Nesse prazo, o consumidor pode exigir a reparação ou substituição do

Digite sua dúvida jurídica aqui...

Ainda foi implementada uma opção onde o usuário pode verificar quais são as fontes que modelo está utilizando.

Além disso, caso necessário, o usuário tem a possibilidade de enviar um arquivo acessório para realizar alguma consulta com o arquivo em questão.

Ao final da sidebar, o usuário ainda tem a possibilidade de exportar o histórico de conversa em alguns formatos.

7.2. Explique a arquitetura do aplicativo, incluindo como o Streamlit, LLM e LangChain são utilizados.

R: Inicialmente, o arquivo ‘agent.py’ recebe os arquivos ‘config.yaml’, onde se encontra o token do modelo GROQ:

groqcloud

Create API Key

API Keys

Manage your API keys. Remember to keep your API keys safe to prevent unauthorized access.

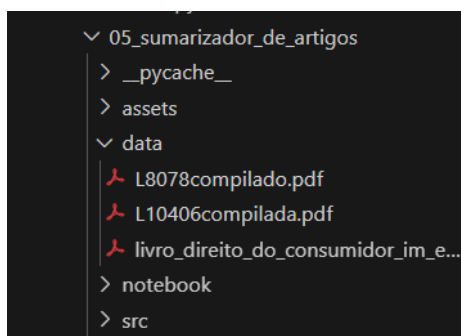
NAME

infnet_app_pd

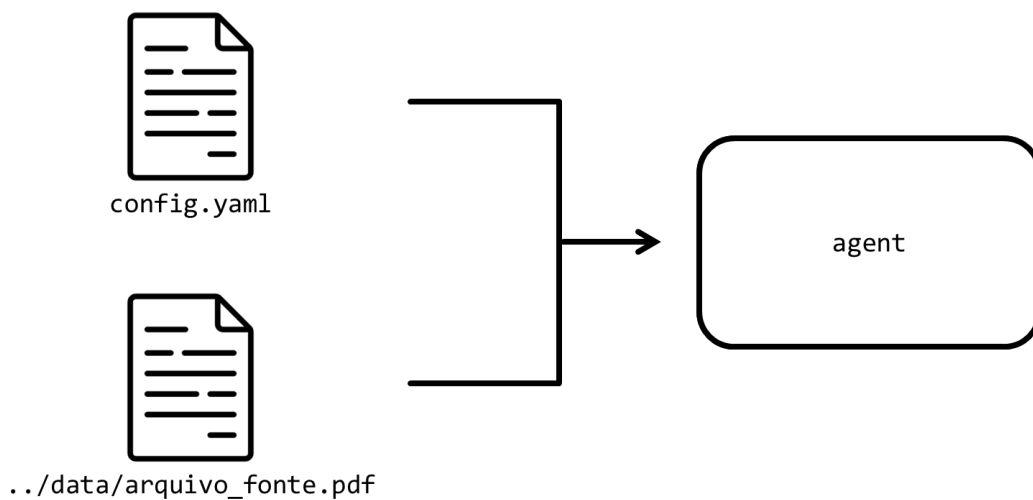
```
DevProjects > GitHubRepositories > ml_models > 06_infnet_llm_pd > 05_sumarizador_de_artigos > ! config.yaml
```

```
1  api_key:
2    key: "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"
3
4
5  model:
6    name: "llama3-70b-8192"
7    temperature: 0.3
8    frequency_penalty: 0.0
9    presence_penalty: 0.0
10   max_tokens: 1024
11   n: 1
12   seed: 42
```

Além disso, o agente ainda realiza o carregamento dos arquivos anexos: O Código de Defesa do Consumidor (Lei n. 8.078/91), o Código Civil (Lei n. 10.406/02) e uma revista pública com diversos artigos científicos sobre o tema:



Representando no fluxograma:



Esses arquivos são carregados nas funções presentes no arquivo 'agent.py' que são chamados pelo arquivo principal 'app.py', que implementa todo o layout do aplicativo.

7.3. Implemente o aplicativo e forneça o código-fonte, junto com instruções para execução.

R: Os códigos estão disponíveis no GitHub, na pasta: https://github.com/GitMateusTeixeira/03-ml-modeling/tree/main/06_infnet_llm_pd/05_sumarizador_de_artigos.

Para a instalação dos pacotes foi utilizado o pip-tools, onde o arquivo 'requirements.in' e 'requirements.txt' estão na pasta: https://github.com/GitMateusTeixeira/03-ml-modeling/tree/main/06_infnet_llm_pd.

Foi implementado assim, pois nesse trabalho estão presentes dois projetos: a análise com NER (pasta '03') e o aplicativo com o Streamlit (pasta '05'). Desse modo, os requirements podem ficar unificados e é possível instalar todos os pacotes utilizados em ambos os projetos de uma só vez.

Uma vez configurado um ambiente virtual e instaladas as bibliotecas, basta abrir um terminal dentro da pasta './05_sumarizador_de_artigos/' e executar o comando 'streamlit run app.py'. A aplicação irá abrir com os arquivos carregados e estará pronta para o uso.

7.4. Apresente evidências e exemplos de uso do aplicativo e discuta os resultados obtidos.

R: O aplicativo se propõe a responder perguntas simples sobre o Direito do Consumidor. Como seguem os exemplos:



Comprei um celular pela internet e ele veio quebrado. Qual o prazo que tenho para pedir meu dinheiro de volta?



Prezado consumidor,

De acordo com o Código de Defesa do Consumidor (Lei nº 8.078/90), o prazo para reclamar sobre um produto defeituoso é de 30 dias, contados a partir da entrega do produto. Nesse caso, você tem direito a reparo ou substituição do produto, ou ainda, a restituição do valor pago.

Além disso, é importante lembrar que o consumidor tem direito à garantia legal, que é de 90 dias, contados a partir da entrega do produto. Durante esse prazo, o consumidor pode solicitar a reparação ou substituição do produto, sem precisar comprovar a culpa do fornecedor.

Portanto, você tem até 30 dias para reclamar sobre o produto defeituoso e solicitar a restituição do valor pago. Caso não seja possível resolver a questão com o fornecedor, é recomendável buscar ajuda de órgãos de defesa do consumidor, como o PROCON.

Lembre-se de que é importante guardar todos os comprovantes de compra e entrega do produto, bem como qualquer comunicação com o fornecedor, pois esses documentos podem ser necessários para comprovar sua reclamação.

Espero que isso tenha ajudado. Se tiver mais alguma dúvida, não hesite em perguntar.

Digite sua dúvida jurídica aqui...



Em relação às funcionalidades, foi adicionado um documento com a Lei dos Juizados Especiais (Lei n. 9.099/95) e foi perguntado sobre um prazo de recurso em um caso.

O aplicativo se mostrou bem em relação à resposta, reforçando que o prazo pode variar de acordo com as circunstâncias (como feriados estaduais e municipais ou ainda algum ato do Tribunal que suspenda a contagem de prazo naquele dia), recomendando ao usuário que consulte um advogado ou defensor público para confirmar tais informações.

Fontes e Estudo

Adicionar Documento

Envie um arquivo (PDF, CSV, Excel)

Drag and drop file here

Limit 200MB per file • PDF, CSV, XLSX

Browse files

L9099.pdf

0.6MB

Documento adicionado à base de conhecimento.

Exportar Histórico

Feito por

Deploy

Em um caso de Juizado Especial, o juiz proferiu uma Sentença no dia 27 de Junho, quanto tempo eu tenho para recorrer dessa decisão?

Prezado consumidor,

De acordo com a Lei nº 9.099/95, que dispõe sobre os Juizados Especiais Cíveis e Criminais, o prazo para interpor recurso contra uma sentença é de 10 (dez) dias, contados a partir da intimação da sentença.

Portanto, considerando que a sentença foi proferida no dia 27 de junho, você tem 10 dias para recorrer dessa decisão, a contar da data em que foi intimado da sentença.

É importante lembrar que o prazo para recurso pode variar dependendo do caso específico e das circunstâncias envolvidas. É recomendável consultar um advogado ou buscar orientação de um órgão de defesa do consumidor para obter mais informações e garantir seus direitos.

Espero que isso tenha ajudado. Se tiver mais alguma dúvida, não hesite em perguntar.

Digite sua dúvida jurídica aqui...

No geral, o aplicativo lida bem com dúvidas simples e pontuais, mas as respostas não devem ser levadas como definitivas de forma alguma, pois é sabido que, no mundo real, os casos possuem suas especificidades, que podem influenciar e muito a resposta final e, ainda, que mesmo o usuário pode não saber informar isso, num primeiro momento.

Em fim, importa ressaltar que, como se trata de uma área do conhecimento onde há muito espaço para situações específicas e imprevistas, de forma alguma pretende-se aqui substituir um advogado de nenhuma forma.