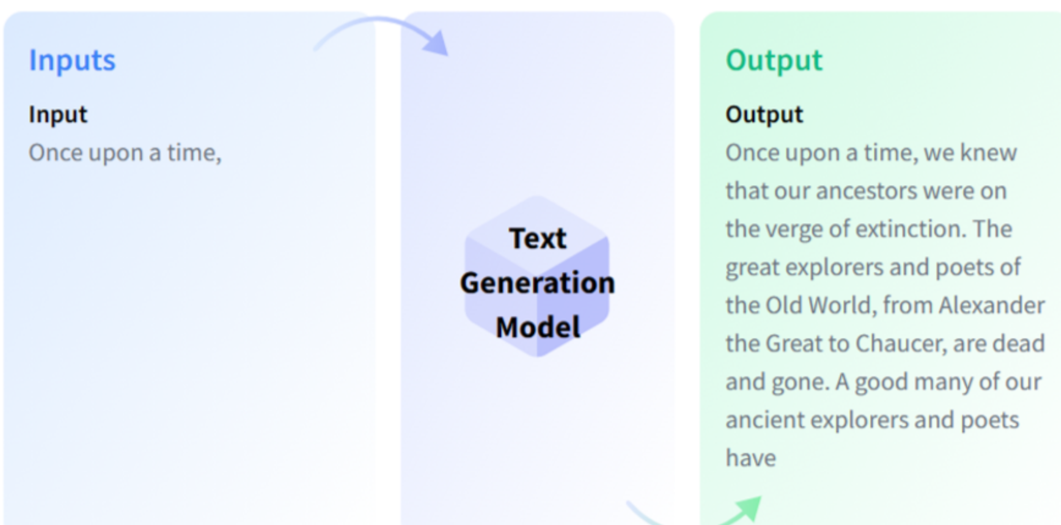
Text generation

Generating text is the task of generating new text given another text This task is more formally known as "natural language generation" in the literature.

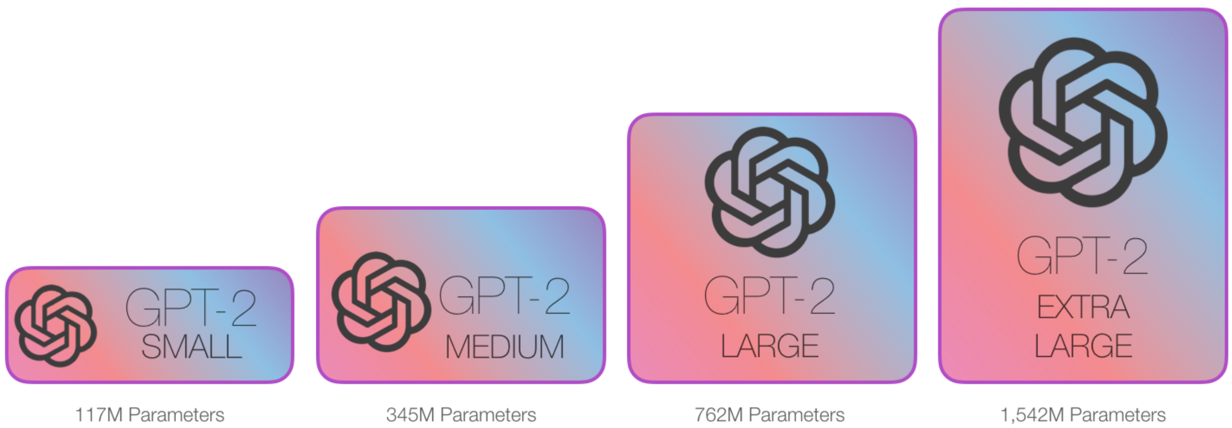
These models can, for example, fill in incomplete text or paraphrase:



Example: **GPT2, XLNet, OpenAI GPT, CTRL, TransformerXL, XLM, Bart, T5, GIT, Whisper.**

Model description: GPT2

GPT-2 is a transformers model pretrained on a very large corpus of English data in a self-supervised fashion. This means it was pretrained on the raw texts only, with no humans labelling them in any way (which is why it can use lots of publicly available data) with an automatic process to generate inputs and labels from those texts. More precisely, it was trained to guess the next word in sentences. we use `small` gpt model.



This is relevant for models that support tasks involving multiple inputs

Input:

This is relevant for models that support tasks involving multiple inputs

`1)Tokenization.`

`2) Special Tokens trained for german language.`

`3)Padding.`

`4)Segmentation.`

`5) Conversion to Input IDs.`

Output:

The output of a GPT-2 model is typically a sequence of tokens representing the generated text. These tokens are numerical IDs that correspond to the indices of the tokens in the model's vocabulary. The output tokens can be decoded back into human-readable text using the model's tokenizer

Steps to make model text_generation

1) **Data Collection**:"**German Recipes Dataset**"

data set is german language contain: **12190 german recipes**

contant:

- Ingredients: the ingredients of the recipe as array
- Instructions: the instructions as free text
- Name: the name of the recipe
- Url: the source url
- Day: the day where the recipe was created
- Month: the month where the recipe was created
- Year: the year where the recipe was created
- Weekday: the weekday where the recipe was created

2) **Preprocessing**: Clean the data by removing irrelevant characters, formatting issues, or any other noise. This step might also include tokenization (splitting the text into words or subwords), lowercasing, and removing stopwords.

3) **Tokenization: return dictionary contain(`input_types,input_ids,_attention_mask`)**

From tokens to input IDs

The conversion to input IDs is handled by the `convert_tokens_to_ids()` tokenizer method:

Attention masks

Attention masks are tensors with the exact same shape as the input IDs `tensor`, filled with `0s` and `1s`: 1s indicate the corresponding tokens should be attended to, and **0s** indicate the corresponding tokens should not be attended to (i.e., they should be ignored by the attention layers of the model).

we use transformers "Pytorch"

```
from transformers import AutoTokenizer

tokenizer = AutoTokenizer.from_pretrained("anonymous-german-nlp/

train_path = 'train_dataset.txt'
test_path = 'test_dataset.txt'
```

4) **Model Selection**: we use model "German_GPT2" [dbmdz/german-gpt2]

5) **Training**: we used pretrained model ["anonymous-german-nlp/german-gpt2"]
we split data in (train and test)

6) **Fine-tuning**: Fine-tuning the model allows users to customize the pre-trained GPT-2 model to suit their specific needs, improving its overall performance and utility in niche domains.

steps for fine-tunning: 1-inastall "transformers" and use pytorch

2-Importing Necessary Libraries such as (Trainer,TrainingArguments,AutoModelWithLMHead,TextDataset,

DataCollatorForLanguageModeling)
3-
*a)*Loading the GPT-2 Model and Tokenizer. *b)*Loading the Training Dataset.

c). Creating Data Collator. *d)* Setting Training Arguments.

e) Initialising the Trainer. *f)* Training the Model. *g)*Saving the Fine-Tuned Model

Training the Model with Text Data

```
from transformers import Trainer, TrainingArguments,AutoModelWit

model = AutoModelWithLMHead.from_pretrained("anonymous-german-nl


training_args = TrainingArguments(
    output_dir="folder", #The output directory
    overwrite_output_dir=True, #overwrite the content of the out
    num_train_epochs=2, # number of training epochs
```

```
    per_device_train_batch_size=32, # batch size for training
    per_device_eval_batch_size=64,  # batch size for evaluation
    eval_steps = 400, # Number of update steps between two evalu
    save_steps=800, # after # steps model is saved
    warmup_steps=500,# number of warmup steps for learning rate
    prediction_loss_only=True,
    report_to="tensorboard"
    )
```

9) **Generation:** Once the model is trained, it can generate text by providing a prompt or seed input. The model then generates the subsequent text based on its learned patterns and probabilities.

```
from transformers import pipeline

generation = pipeline('text-generation',model='anonymous-german-
```

Internal structure for attention for text generation

The Encoder-Decoder Framework

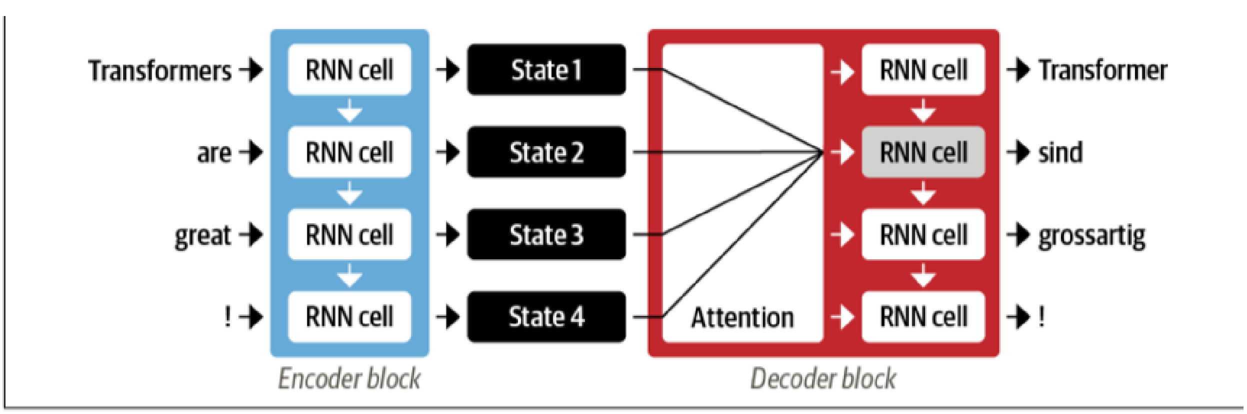


Figure 1-4. An encoder-decoder architecture with an attention mechanism for a pair of RNNs

Decoding

Decoding is going the other way around: from vocabulary indices, we want to get a string.

This can be done with the `decode()` method as follows:

```
decoded_string = tokenizer.decode([7993, 170, 11303, 1200, 2443, 1110, 3014])
print(decoded_string)
```

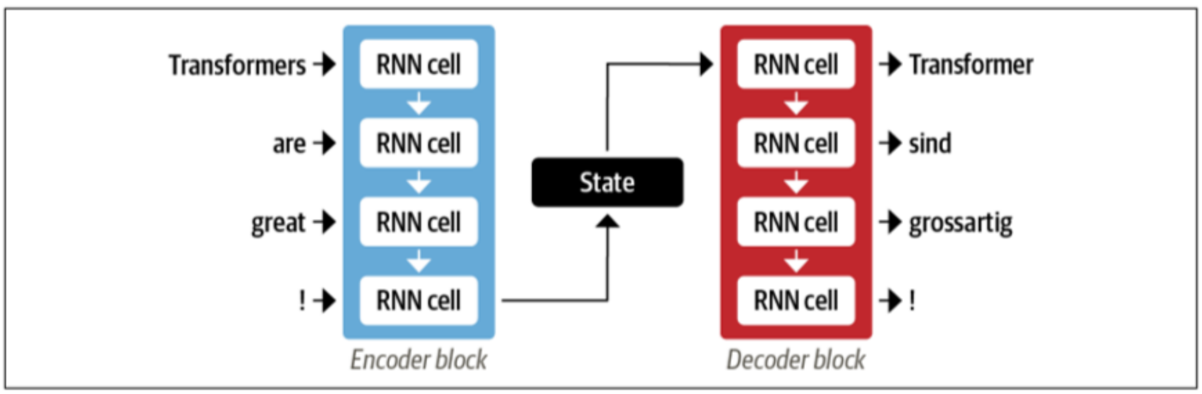


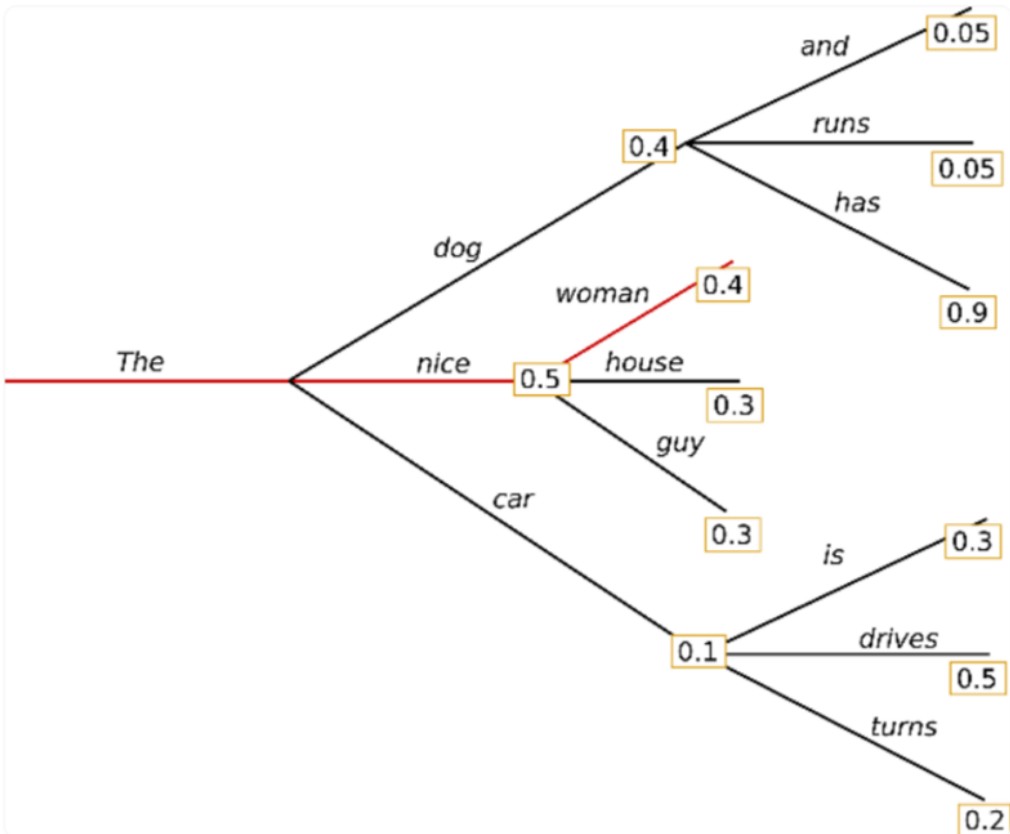
Figure 1-3. An encoder-decoder architecture with a pair of RNNs (in general, there are many more recurrent layers than those shown here)

Greedy Search

`generate` uses greedy search decoding by default so you don't have to pass any parameters to enable it. This means the parameters `num_beams` is set to 1 and `do_sample=False`.

Greedy Search

Greedy search is the simplest decoding method. It selects the word with the highest probability as its next word: $w_t = argmax_w P(w|w_{1:t-1})$ at each timestep t. The following sketch shows greedy search.



Starting from the word "The", the algorithm greedily chooses the next word of highest

Masked self_Attention

An attention mask is **a binary mask that designates which tokens should be attended to (assigned non-zero weights) and which should be ignored (assigned zero weights)**

such as(**Full Attention,Scaled Dot-Product Attention,Masked Attention**)

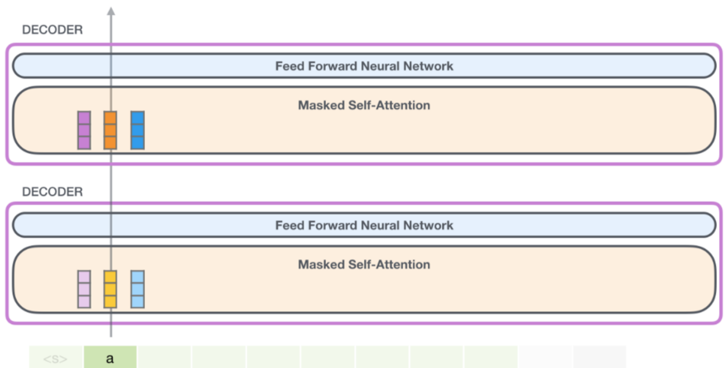
GPT-2 Masked Self-Attention

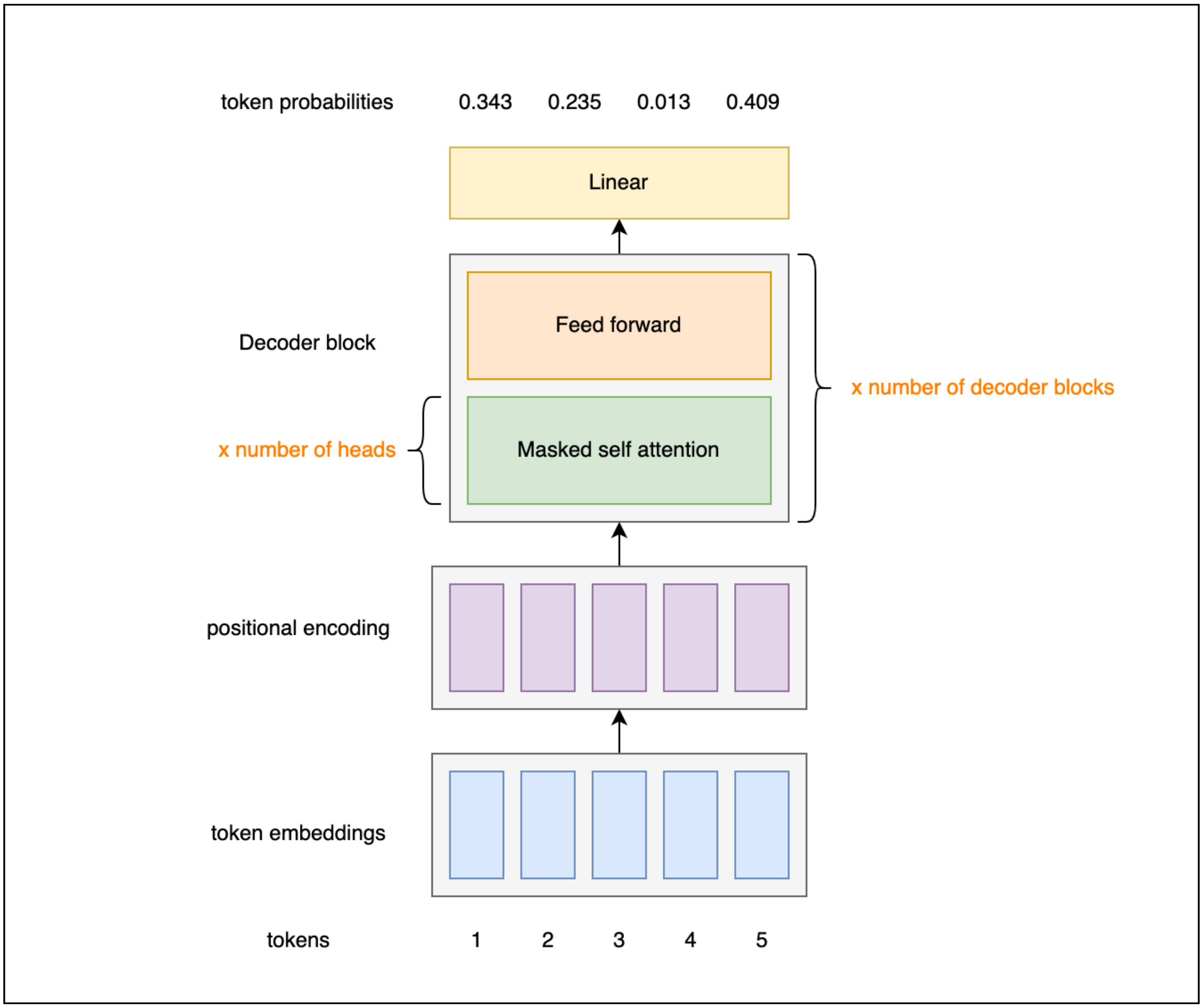
Let's get into more detail on GPT-2's masked attention.

Evaluation Time: Processing One Token at a Time

We can make the GPT-2 operate exactly as masked self-attention works. But during evaluation, when our model is only adding one new word after each iteration, it would be inefficient to recalculate self-attention along earlier paths for tokens that have already been processed.

In this case, we process the first token (ignoring `<s>` for now).





Comparison between models :

1) GPT-2 Model:

Description:

The GPT-2 (Generative Pre-trained Transformer 2) model is a type of language model developed by OpenAI. It is based on the transformer architecture and is trained on a large corpus of text data to generate human-like text. The model is capable of generating coherent and contextually relevant text given a prompt. It does this by predicting the next word in a sequence of text based on the words that have come before it. and it's ##usage is instantiated the same way as in the Transformers library. The only difference is that there are a few new training arguments specific to HPUs.

Definition:

> **Output Hidden States:** If set to True, the model will return all hidden states. This can be useful for tasks like fine-tuning or analysis.
>
> **Output Attentions:** If set to True, the model will return attention weights. This is helpful for understanding which parts of the input the model pays attention to.
>
> **Output Cross-Attention:** If set to True, the model will return cross-attention weights. This is relevant for models that support tasks involving multiple inputs.
>
> **Num Layers:** Specifies the number of transformer layers in the model.
>
> **Num Heads:** Specifies the number of attention heads in each transformer layer.
>
> **Hidden Size:** Specifies the size of the hidden layers in the transformer network.
>
> **Intermediate Size:** Specifies the size of the intermediate (feed-forward) layers in the transformer network.

> **Activation Function:** Specifies the activation function used in the feed-forward layers, typically "gelu" or "relu".
>
> **Dropout Rate:** Specifies the dropout probability for dropout layers within the model.
>
> **Attention Dropout Rate:** Specifies the dropout probability for attention scores in the attention layers.
>
> **Initializer Range:** Specifies the range for random weight initialization.
> These are just some of the parameters available for configuring the GPT-2 model in Hugging Face's transformers library. Depending on your specific use case, you may need to adjust these parameters to achieve optimal performance or to suit your task requirements.

and ,

Model Size:

GPT-2 comes in different sizes, ranging from "small" (117M parameters) to "large" (1.5B parameters). You can specify the model size using the model_name_or_path parameter when initializing the model.

Context length:

The GPT-2 model has a context length of 1024 tokens for its base version and up to 8192 tokens for larger versions.

Advantages:

```
1-strength: provides "true" features for text generation and
can be used as a flexible
framework for learning text representations
2-Accuracy: GPT-Pat effectively detects machine-generated tex
t with high accuracy
3-Empowering Label Mapping: GPT-2 has strong generative power
and can enhance semantic
embedding and improve mapping to relevant labels.
4-Community Support: Hugging Face has a large community of us
ers and contributors,      providing support and resources fo
r working with GPT-2 and other models.
5-High-quality text generation: With a wide range of uses GPT
-2 is known for its
capacity to produce high-quality human-like writing.
6-Pre-trained models: The pre-trained models included with GP
T-2 may be utilized for a
variety of applications involving Natural Language Processin
g, without the need for an
additional training process.
7-Large-scale architecture: The architecture of GPT-2 is buil
t to handle enormous
volumes of data, making it appropriate for applications that
need to analyze massive
datasets.
8-Flexibility: GPT-2 is tailored to perform a range of Natura
l Languages Processing
tasks, such as question answering, text summarization, and la
nguage translation, and it can be fine-tuned on specific data
sets or tasks, making it adaptable to different
applications and domains, provides "true" features for text g
eneration and can be used
as a flexible framework for learning text representations
9-Ease of Use: GPT-2 is relatively easy to use compared to so
me other AI models,
```

```
requiring minimal input to generate text.
10-Open Source: The model and its codebase are open source, e
nabling researchers and developers to use and build upon it f
reely.
```

Disadvantages:

```
1-Computational Resources: GPT-2 is a large model that requir
es significant
computational resources for training and inference, which can
be costly and
time-consuming, especially for large-scale applications.
2-Fine-tuning Complexity: Fine-tuning the model for specific
tasks or datasets can be
complex and may require expertise in natural language process
ing (NLP) and machine
learning.
3-Limited Context Understanding: While GPT-2 can generate coh
erent text, it may
sometimes struggle with understanding complex contexts or gen
erating long-range
dependencies in text.
4-Ethical Considerations: GPT-2 has raised concerns about the
potential misuse of AI-generated text for spreading misinform
ation or generating harmful content, leading to
ethical considerations regarding its deployment.
5-Model Size: The size of the GPT-2 model can be a disadvanta
ge in some applications where resource constraints are a conc
ern, as it may not be feasible to deploy the model in environ
ments with limited resources.
6-Lack of Control: While GPT-2 can generate high-quality tex
t, it may sometimes produce outputs that are inappropriate or
unintended, requiring careful monitoring and control
```

```
mechanisms.
7-Bias: Like any model trained on human-generated data, GPT-2
can exhibit biases present in the training data.
```

2)Mistral-7B-Instruct-v0.2 Model:

Description:

The Mistral-7B-Instruct-v0.2 model is likely a variant of a natural language processing model developed for text generation tasks. It could be based on architectures like Transformers or recurrent neural networks (RNNs). The "7B" might refer to the number of parameters in the model, indicating its size and complexity. The "Instruct" part of the name might suggest a focus on generating instructional or procedural text.

Definition :

has the same definition of GPT-2 Model

but the difference is in the model size

Model Size:

Different variants of Mistral may have different sizes or numbers of parameters. The model size can impact its performance and computational requirements.

Context length:

The context length for the Mistral-7B-Instruct-v0.2 model of text generation is 32,000 tokens , This context window is significantly larger than the 8,000-token context window of its predecessor, **Mistral-7B-v0.1**

Advantages:

```
1-Enhanced context: The larger context window enables the model
context-aware information, leading to improved responses.
2-Fine-tuning flexibility: Developers can easily fine-tune this
tasks and applications, making it versatile and adaptable.
3-Large model size (7 billion parameters): A larger model typica
capture of linguistic nuances, semantics, and context, leading t
generation compared to smaller models.
4-Adaptability: The model might be fine-tuned on specific instru
tasks,enhancing its performance and relevance to particular doma
```

Disadvantages:

```
1-Lack of moderation mechanisms: The model does not include any
features, which could be a limitation in certain use cases.
2-Computational requirements: A model with 7 billion parameters
substantial computational resources for training and inference.
accessibility to users without access to powerful hardware.
3-Data biases: Like other large language models, Mistral-7B-Inst
biases present in the training data, potentially leading to bias
instructional outputs.
4-Lack of interpretability: Large models like Mistral-7B-Instruc
describedn as "black boxes," making it challenging to understand
decision-making process or troubleshoot errors.
```

3)Google/gemma-2b Model:

Description:

Gemma is a family of lightweight, state-of-the-art open models from Google, built from the same research and technology used to create the Gemini models. They are text-to-text, decoder-only large language models, available in English, with open weights, pre-trained variants, and instruction-tuned variants. Gemma models are well-suited for a variety of text generation tasks, including question answering, summarization, and reasoning. Their relatively small size makes it possible to deploy them in environments with limited resources such as a laptop, desktop or your own cloud infrastructure, democratizing access to state of the art AI models and helping foster innovation for everyone.

Definition :

has the same definition of GPT-2 Model

but the difference is in the model size

Model Size:

Different variants of Mistral may have different sizes or numbers of parameters. The model size can impact its performance and computational requirements.

Context length:

Gemma 2B models are trained on a context length of 8192 tokens.

Advantages:

```
1-Relatively Small Size: Compared to other large language models
small. This makes it easier to deploy on devices with limited re
desktops, or even personal cloud infrastructure. This wider acce
innovation and democratizes access to advanced AI for text gene
```

```
2-State-of-the-Art: Despite their smaller size, they offer compe
to their efficient architecture.
3-Democratizing Access: By being accessible to a wider audience,
and creativity.
4-Open-Source Availability: Gemma-2b's open-source nature allows
transparency, customization, and collaboration within the AI co
the underlying code and weights, fine-tune the model for specifi
to its development.
```

Disadvantages:

```
1-Limited Capabilities: Gemma-2b might not be as powerful as sor
models on the market. This could show up in areas like handling
or generating highly nuanced creative text formats.
2-Potential for Biases: Like many AI models, Gemma-2b can inheri
data it's trained on. This can lead to outputs that are discrimi
It's crucial to be mindful of these potential biases and impleme
safeguards.
3-Fine-Tuning Expertise Required: To unlock the full potential o
tasks, some expertise in fine-tuning large language models might
limit accessibility for users who are not familiar with AI or ma
```

https://huggingface.co/google-bert/bert-base-cased