

In summary:



Multi-Head Attention is a vital component within Transformers.

An example:



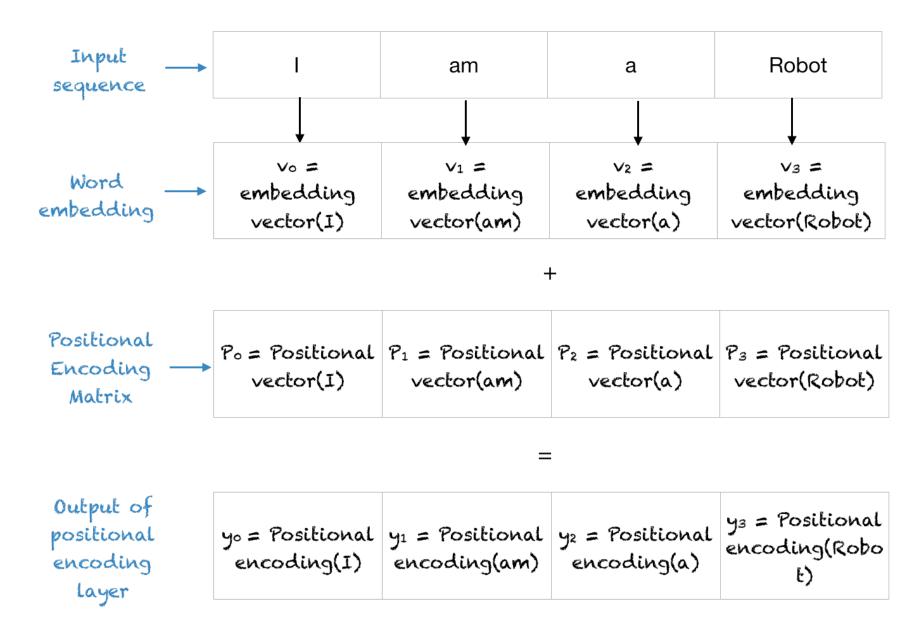
Positive:

I loved the restaurant even though the pasta was mediocre

Negative:

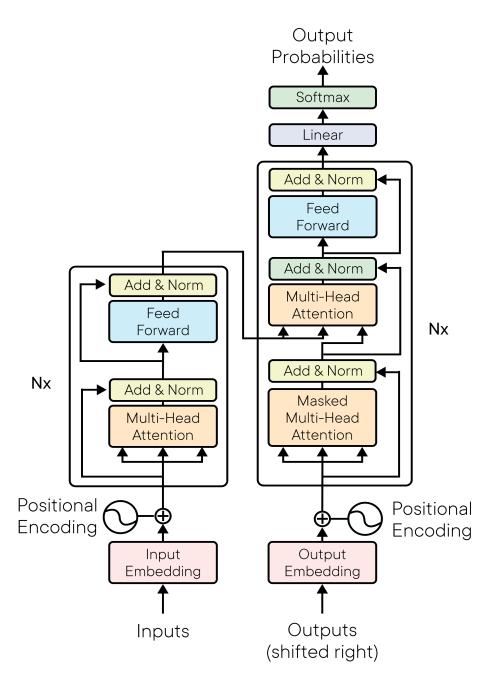
I hate this place even though it has fantastic steak





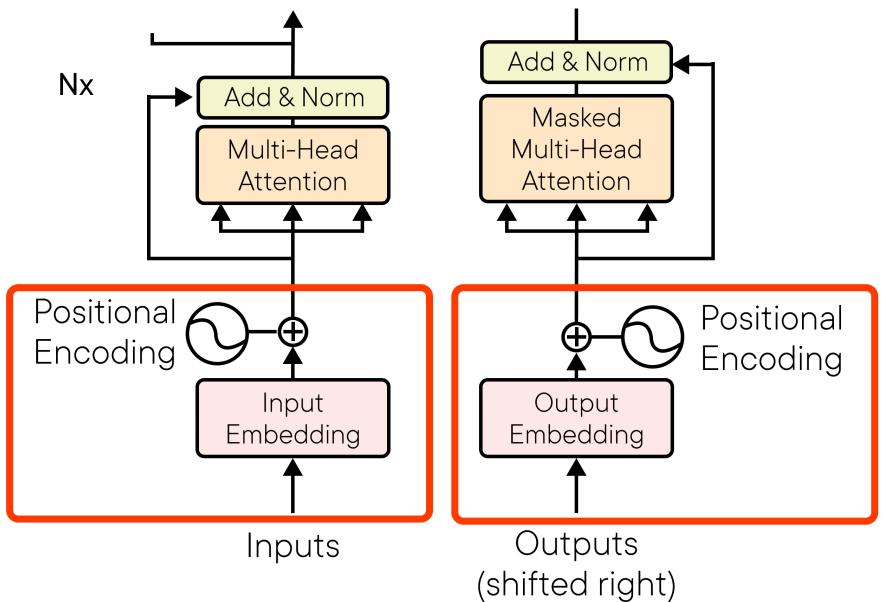




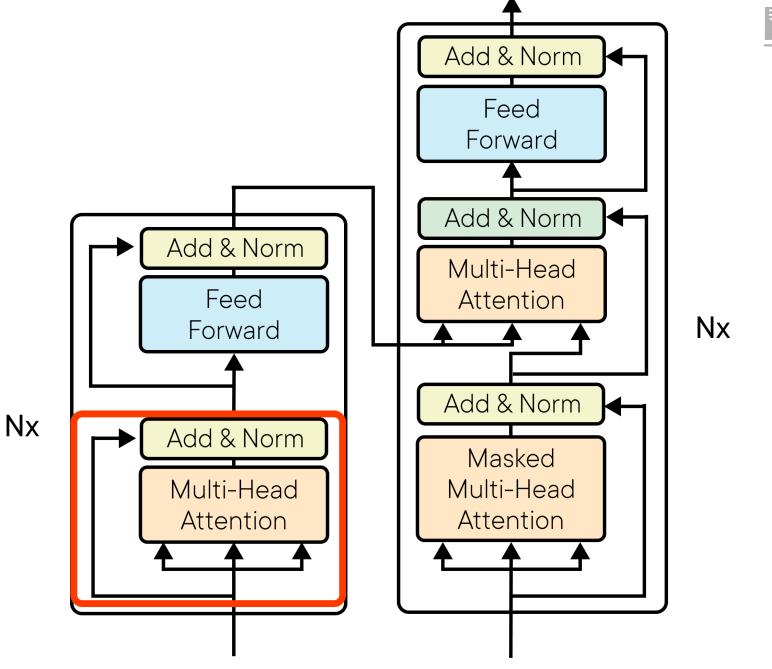




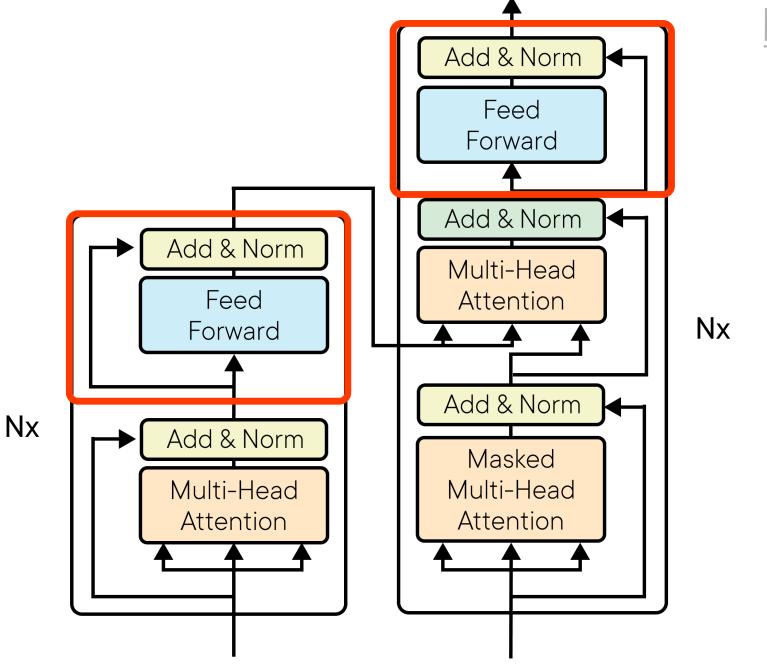






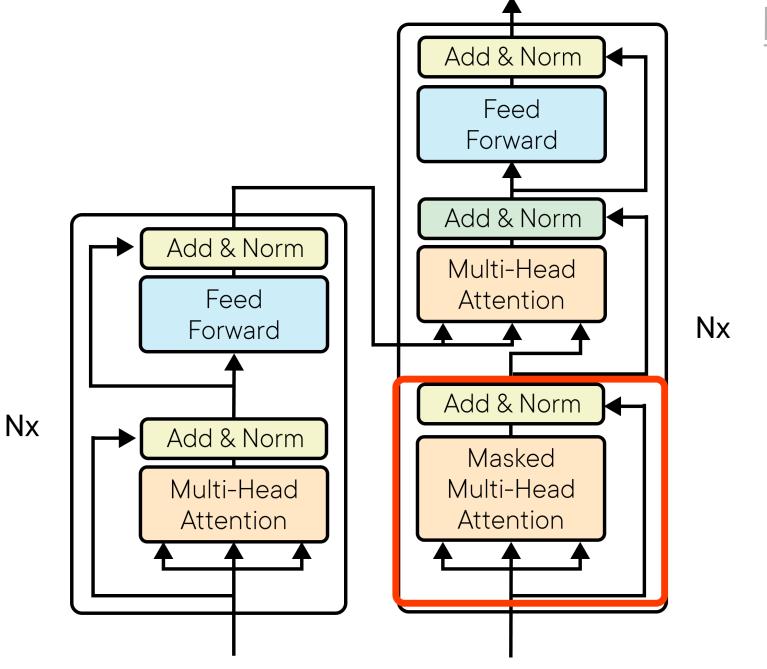




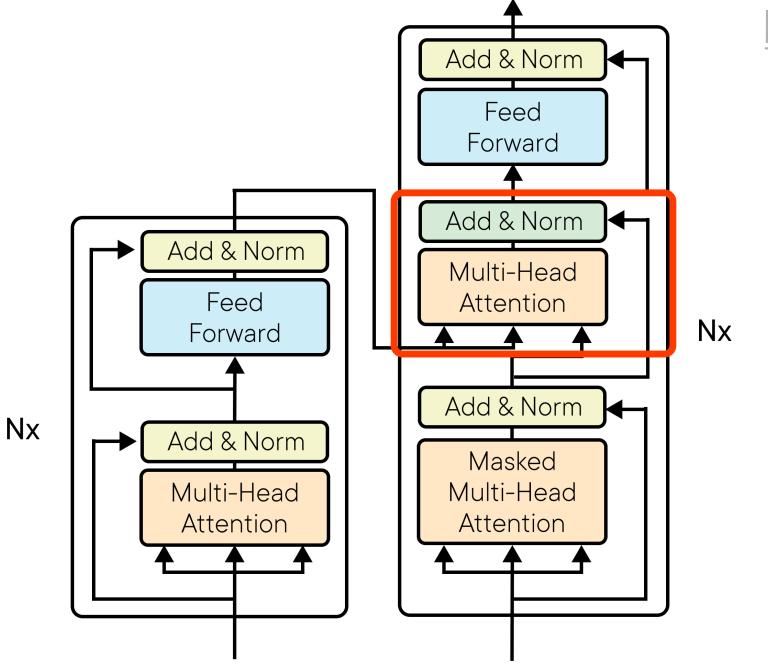






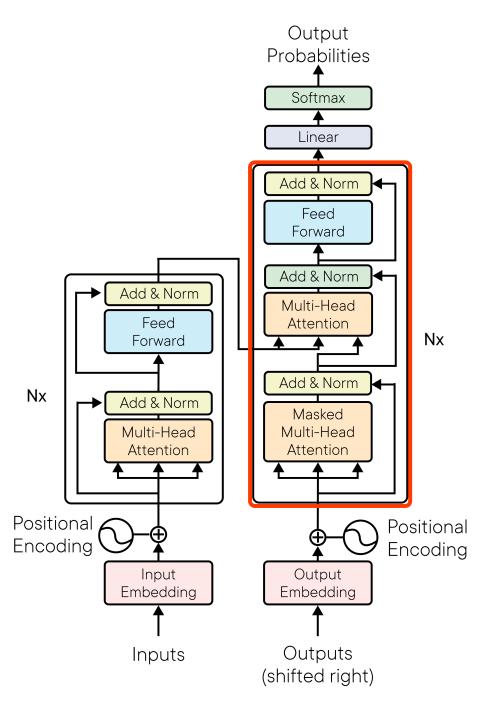


















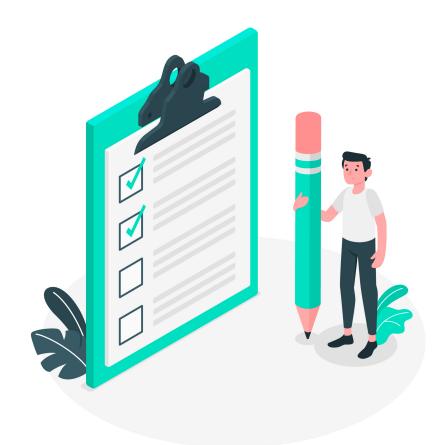
Transformers can parallelise computations across a GPU, which RNNs cannot

Transformers do not have the informational bottleneck

And Transformers have much fewer parameters for the same size of architecture than an RNN



Our first transformer







Introducing Hugging Face





Hugging Face is an open-source library and plataforma that provides a wide range of tolos and models for NLP tasks.





- Pretrained Models: Hugging Face provides a vast collection of pretrained models, including popular architectures like BERT, GPT and many more.
- **Transformers Library:** The Transformers Library, developed by Hugging Face, is a go-to resource for NLP practitioners.
- Datasets library: At your disposal, you have the datasets library to download tons of famous datasets along with their metrics to compare your models with the current state of the art.



1. Download and load the tokenizer and model



```
import tensorflow as tf
from transformers import AutoTokenizer, AutoModelForSequenceClassification
from datasets import load_dataset
from sklearn.model_selection import train_test_split
# Download and load the tokenizer and model
model_name = "distilbert-base-uncased"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForSequenceClassification.from_pretrained(model_name,
num_labels=2)
# Load the IMDb dataset from Hugging Face datasets library
dataset = load_dataset("imdb")
# Split the dataset into training and testing sets
train_dataset, test_dataset = train_test_split(dataset['train'],
test_size=0.2, random_state=42)
# Tokenize the training dataset
train_encodings = tokenizer(train_dataset['text'], truncation=True,
padding=True)
train_labels = train_dataset['label']
# Tokenize the testing dataset
```

2. Load the IMDB dataset from Hugging Face datasets library



```
import tensorflow as tf
from transformers import AutoTokenizer, AutoModelForSequenceClassification
from datasets import load_dataset
from sklearn.model_selection import train_test_split
# Download and load the tokenizer and model
model name = "distilbert-base-uncased"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForSequenceClassification.from_pretrained(model_name,
num_labels=2)
# Load the IMDb dataset from Hugging Face datasets library
dataset = load_dataset("imdb")
# Split the dataset into training and testing sets
train_dataset, test_dataset = train_test_split(dataset['train'],
test_size=0.2, random_state=42)
# Tokenize the training dataset
train_encodings = tokenizer(train_dataset['text'], truncation=True,
padding=True)
train_labels = train_dataset['label']
# Tokenize the testing dataset
```

2. Split the dataset into training and testing sets



```
import tensorflow as tf
from transformers import AutoTokenizer, AutoModelForSequenceClassification
from datasets import load_dataset
from sklearn.model_selection import train_test_split
# Download and load the tokenizer and model
model name = "distilbert-base-uncased"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForSequenceClassification.from_pretrained(model_name,
num_labels=2)
# Load the IMDb dataset from Hugging Face datasets library
dataset = load_dataset("imdb")
# Split the dataset into training and testing sets
train_dataset, test_dataset = train_test_split(dataset['train'],
test_size=0.2, random_state=42)
# Tokenize the training dataset
train_encodings = tokenizer(train_dataset['text'], truncation=True,
padding=True)
train_labels = train_dataset['label']
# Tokenize the testing dataset
```

4. Tokenize the datasets

index.py

import tensorflow as tf



```
from transformers import AutoTokenizer, AutoModelForSequenceClassification
from datasets import load_dataset
from sklearn.model_selection import train_test_split
# Download and load the tokenizer and model
model_name = "distilbert-base-uncased"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForSequenceClassification.from_pretrained(model_name,
num_labels=2)
# Load the IMDb dataset from Hugging Face datasets library
dataset = load_dataset("imdb")
# Split the dataset into training and testing sets
train_dataset, test_dataset = train_test_split(dataset['train'],
test_size=0.2, random_state=42)
# Tokenize the training dataset
train_encodings = tokenizer(train_dataset['text'], truncation=True,
padding=True)
train_labels = train_dataset['label']
# Tokenize the testing dataset
```

Create datasets

index.py

Train the model

for epoch in range(epochs):

. . . # Training loop



```
# Create TensorFlow datasets
train_ds = TensorDataset(train_encodings, train_labels)
test_ds = TensorDataset(test_encodings, test_labels)
train_dl = DataLoader(train_ds, 256, shuffle=True, drop_last=True)
test_dl = DataLoader(test_ds, 256, shuffle=True, drop_last=True)

# Configure training parameters
optimizer = optim.AdamW(learning_rate=1e-5)
loss =nn.NLLLoss()
```

Configure training parameters

index.py

Create TensorFlow datasets

Train the model

for epoch in range(epochs):

. . . # Training loop



```
test_ds = TensorDataset(test_encodings, test_labels)
train_dl = DataLoader(train_ds, 256, shuffle=True, drop_last=True)
test_dl = DataLoader(test_ds, 256, shuffle=True, drop_last=True)

# Configure training parameters
optimizer = optim.AdamW(learning_rate=1e-5)
loss =nn.NLLLoss()
```

train_ds = TensorDataset(train_encodings, train_labels)

Train the model

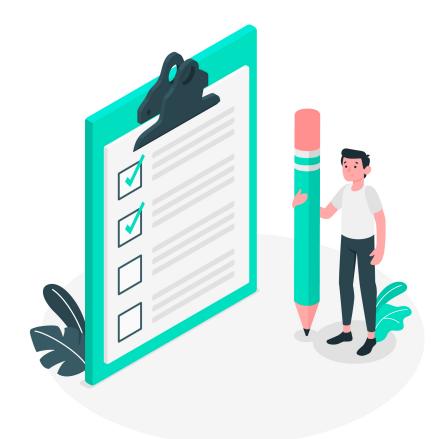


```
# Create TensorFlow datasets
train_ds = TensorDataset(train_encodings, train_labels)
test_ds = TensorDataset(test_encodings, test_labels)
train_dl = DataLoader(train_ds, 256, shuffle=True, drop_last=True)
test_dl = DataLoader(test_ds, 256, shuffle=True, drop_last=True)
# Configure training parameters
optimizer = optim.AdamW(learning_rate=1e-5)
loss =nn.NLLLoss()
```

```
# Train the model
for epoch in range(epochs):
    . . # Training loop
```

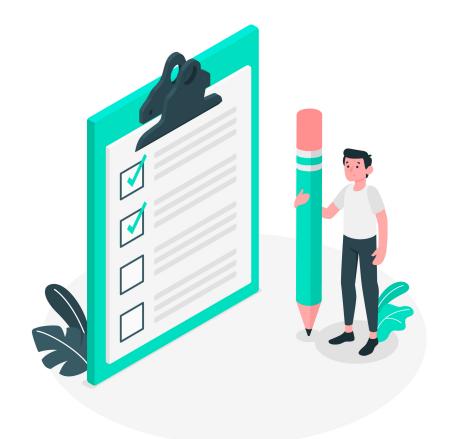


Prompt Engineering with Flan-t5





Transfer learning on Transformers with Hugging Face



Summary



- The Transformer architecture solved all of the significant challenges we had with RNNs for sequence models
- With Hugging Face, now it most straightforward tan ever just to graba checkpoint from someone and finetune it to your needs
- Some models like T5 can perform on multiple task because that's how they were trained and we can leverage it

