Hands-on Hugging Face for Natural Language Processing

Prerequisites

- Comfortable programming in Python
- Basic familiarity with machine learning and deep learning
- Basic familiarity with natural language processing

Poll Question

How comfortable are you building and training machine learning models in Python?

- Never worked with ML before
- Built traditional ML models using scikit-learn
- Built ML models using TensorFlow and PyTorch

Poll Question

Have you worked with Natural Language Processing before?

- Never worked with NLP before
- Built traditional NLP models with scikit-learn
- Built NLP models with TensorFlow, PyTorch

Hugging Face

A company known for its work in the field of artificial intelligence particularly in the field of natural language processing

Hugging Face

A machine learning and data science platform and community that helps users build, deploy, and train machine learning models

Natural Language Processing with Transformers

- The original transformer architecture was introduced in the 2017 paper "**Attention Is All You Need**"

- Introduced the novel concept of attention to focus on the right parts of the input

- Revolutionized the field of NLP, leading to the powerful LLMs that we work with today

Hugging Face Transformer Library

- Offer a wide range of **pre-trained** transformer models for different use cases
- Cross-framework compatibility – all models can be used with both PyTorch and TensorFlow
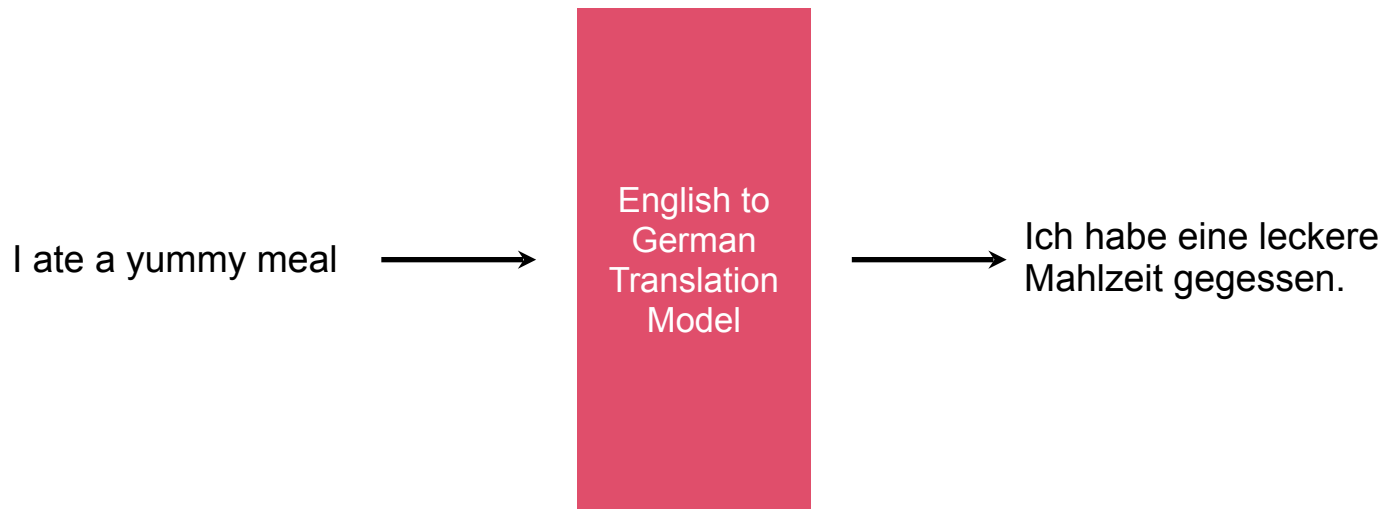- Used in a variety of applications – chatbots, enhancing search, translation, and more

Attention in Neural Networks

Mechanism that allows the network to focus on specific parts of the input data while ignoring others

Language Translation Model

I ate a yummy meal →

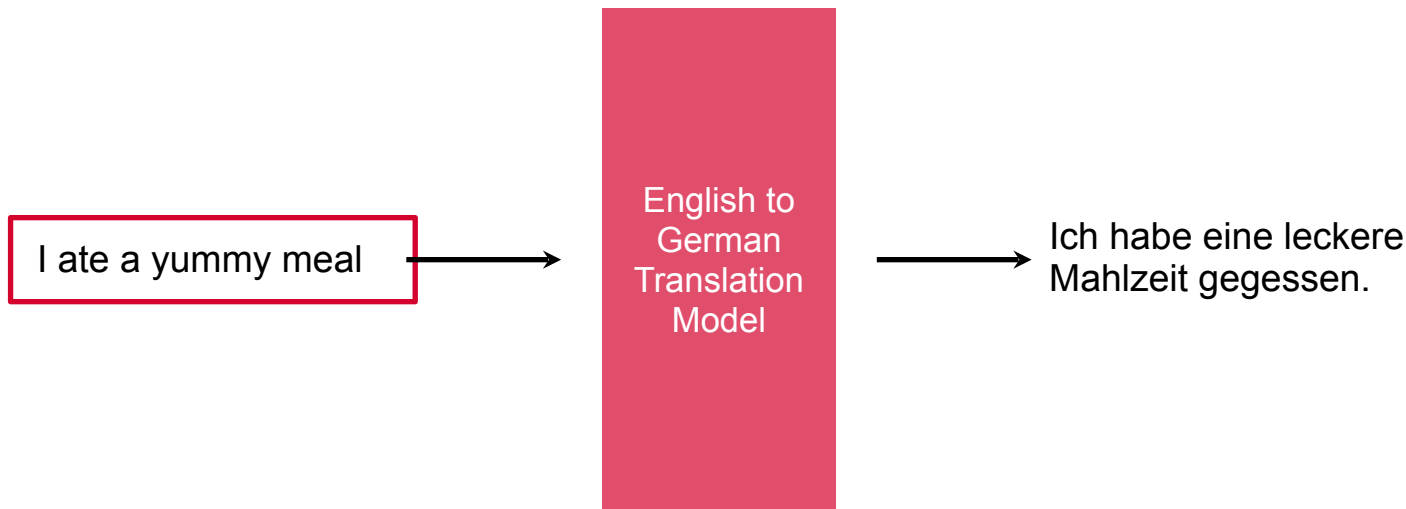English to German Translation Model
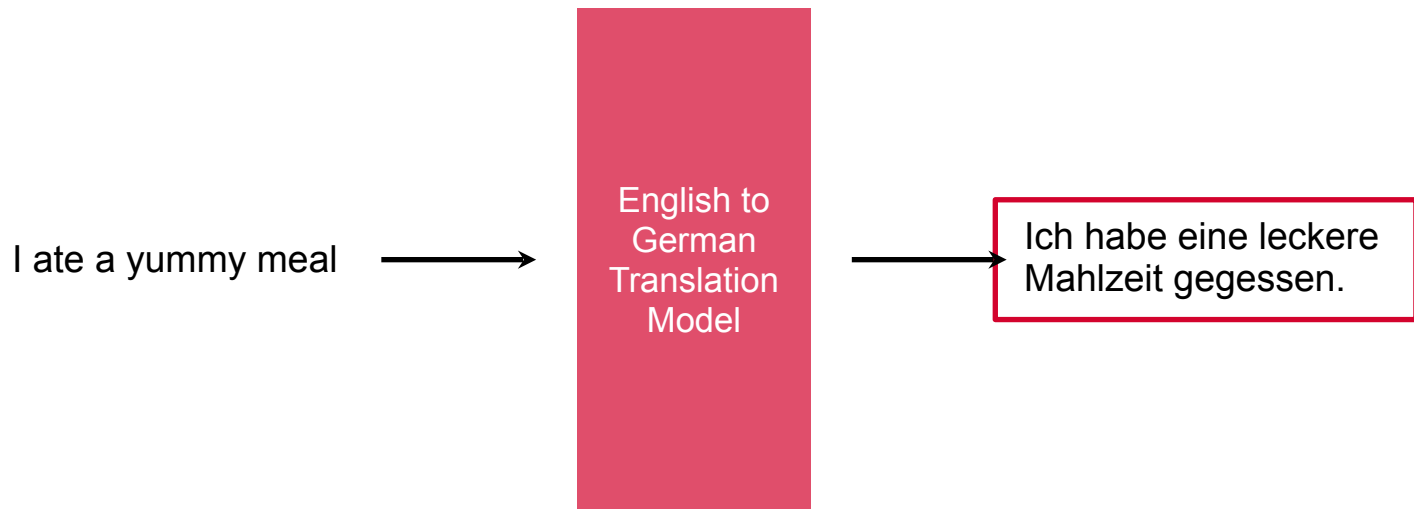
→ Ich habe eine leckere Mahlzeit gegessen.

Model Without Attention

I ate a yummy meal → English to German Translation Model → Ich habe eine leckere Mahlzeit gegessen.
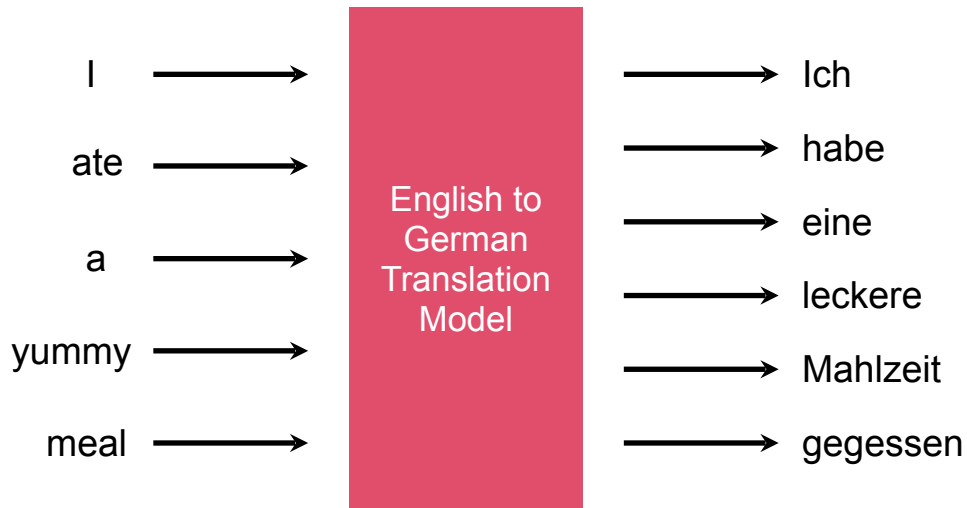
Parses the Entire Input

I ate a yummy meal → English to German Translation Model → Ich habe eine leckere Mahlzeit gegessen.

To Generate the Translation

I ate a yummy meal → **English to German Translation Model** → Ich habe eine leckere Mahlzeit gegessen.

Input and Output are Sequences

I → [English to German Translation Model] → Ich

ate → → habe

a → → eine

yummy → → leckere

meal → → Mahlzeit

→ gegessen

Input Sequence Fed In

I ⟶

ate ⟶

a ⟶

yummy ⟶

meal ⟶

English to
German
Translation
Model
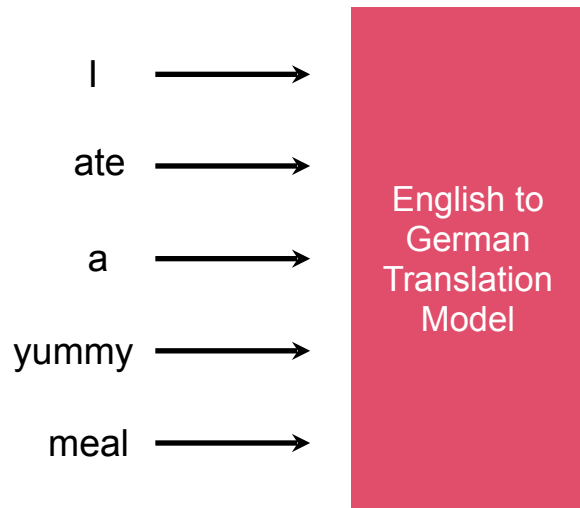
Output Sequence Produced one Word at a Time

I $\longrightarrow$

ate $\longrightarrow$

a $\longrightarrow$

yummy $\longrightarrow$

meal $\longrightarrow$

English to German Translation Model

$\longrightarrow$ Ich
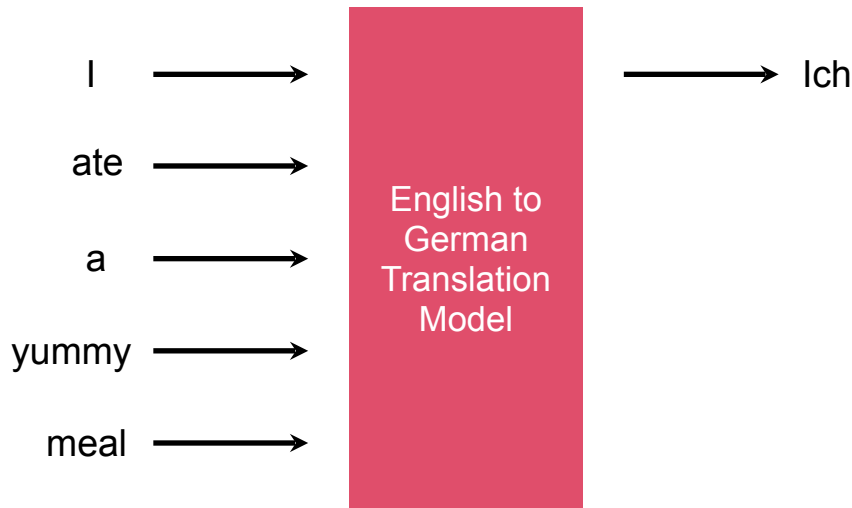
Complete Input + Words Produces So Far Used to Get Next Word in Sequence

I →

ate →

a →

yummy →

meal →

English to German Translation Model

→ Ich

→ habe
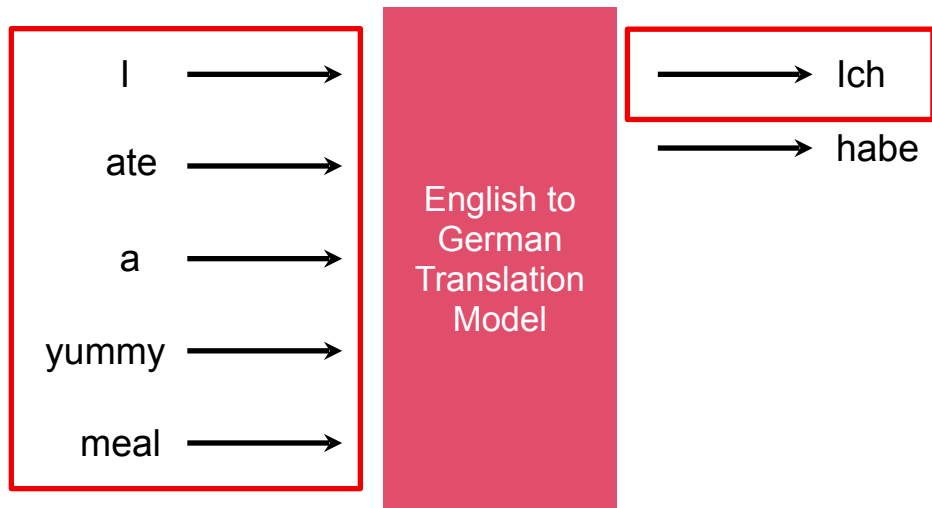
Predict One Word at a Time
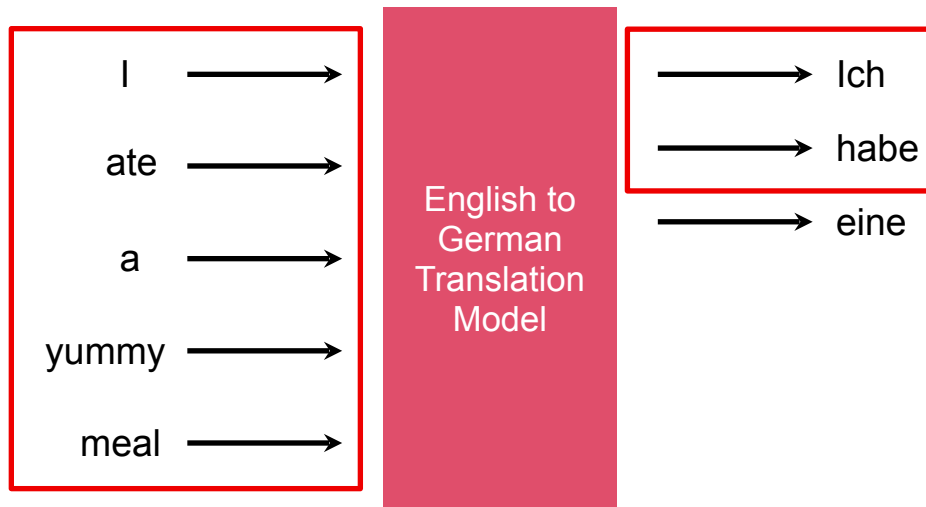
Better Predictions with Attention
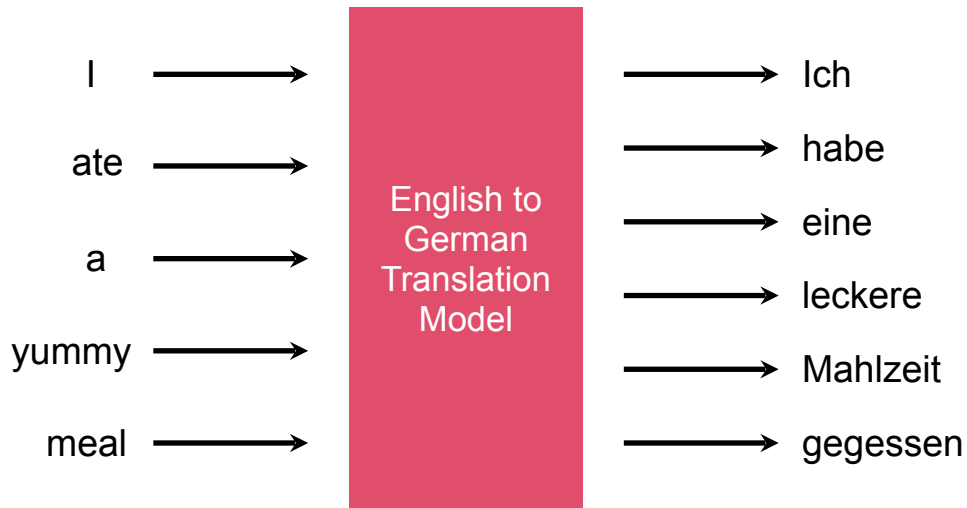
I $\rightarrow$ $\rightarrow$ Ich

ate $\rightarrow$ $\rightarrow$ habe

a $\rightarrow$ English to German Translation Model $\rightarrow$ eine

yummy $\rightarrow$ $\rightarrow$ leckere

meal $\rightarrow$ $\rightarrow$ Mahlzeit

$\rightarrow$ gegessen

Better Predictions with Attention

I → Ich

ate →

a →

yummy →

meal →

English to German Translation Model

→ Ich

→ habe

→ eine

→ leckere

→ Mahlzeit

→ gegessen

Better Predictions with Attention

I → | English to German Translation Model | → Ich
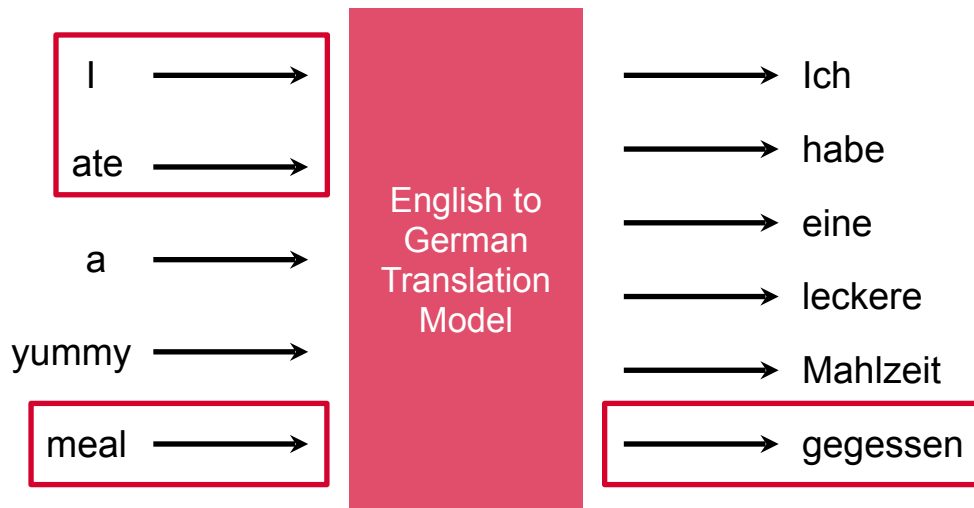
ate → habe

a → eine

yummy → leckere

meal → Mahlzeit

gegessen
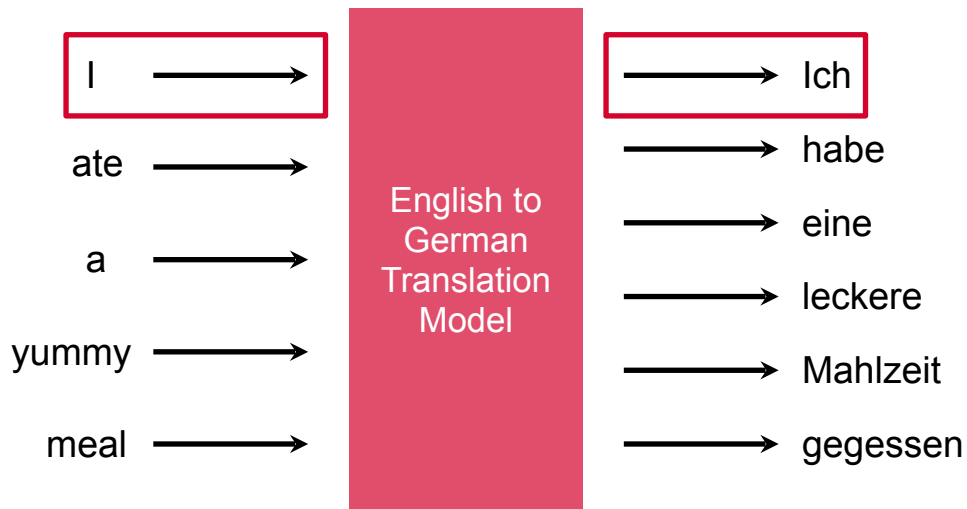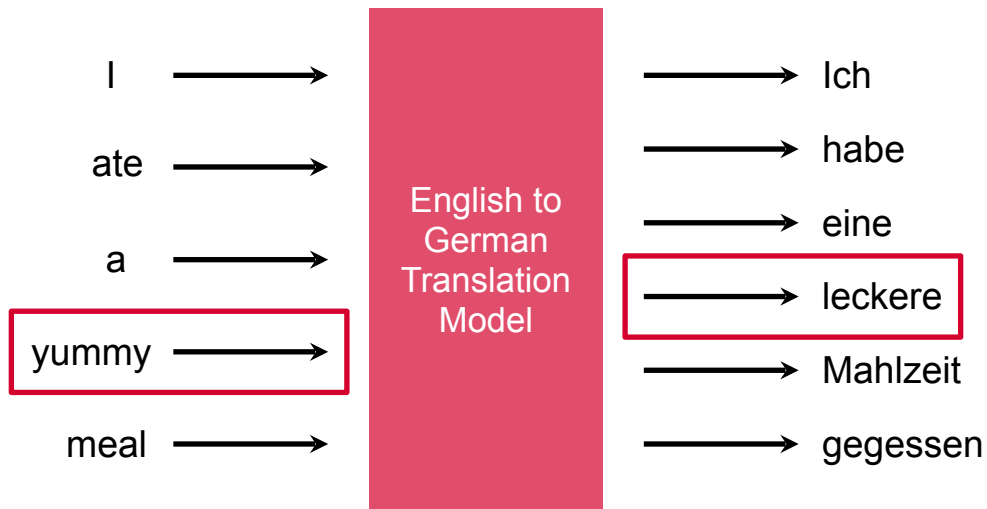
Better Predictions with Attention

Attention

- Help models focus on the right parts of the input to generate the output
- The entire input along with attention generates better outputs
- Are the foundation of transformers used to power the large language models of today
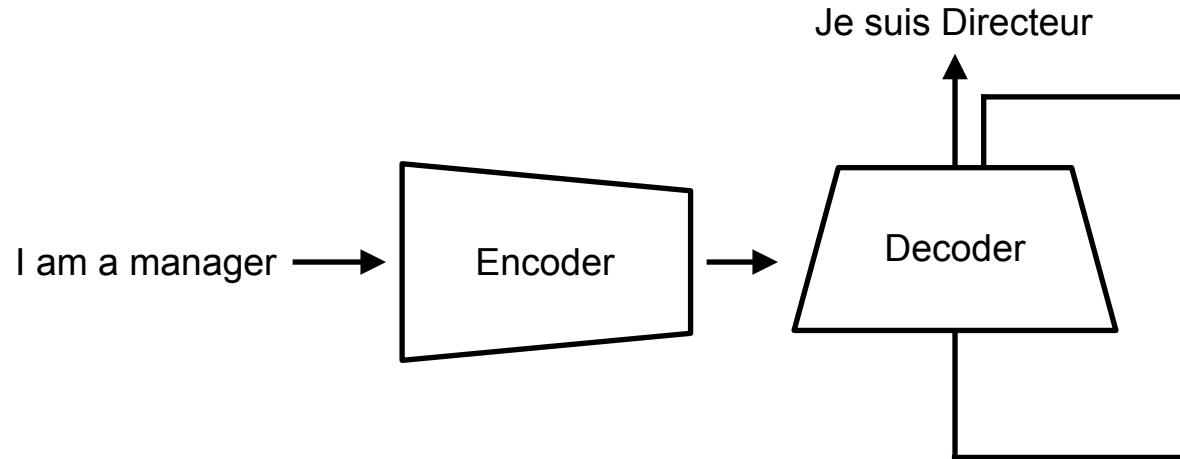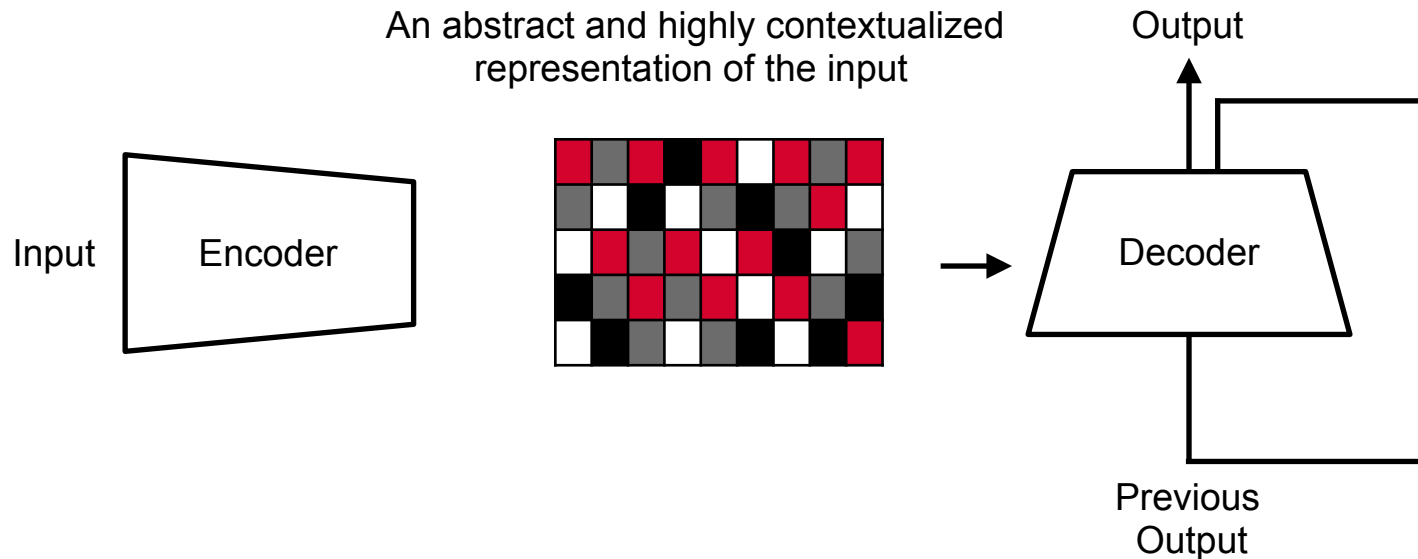
The Transformer Architecture

The Transformer Architecture

A model architecture primarily used in natural language processing that relies on self-attention mechanisms to process sequential data more effectively than previous models
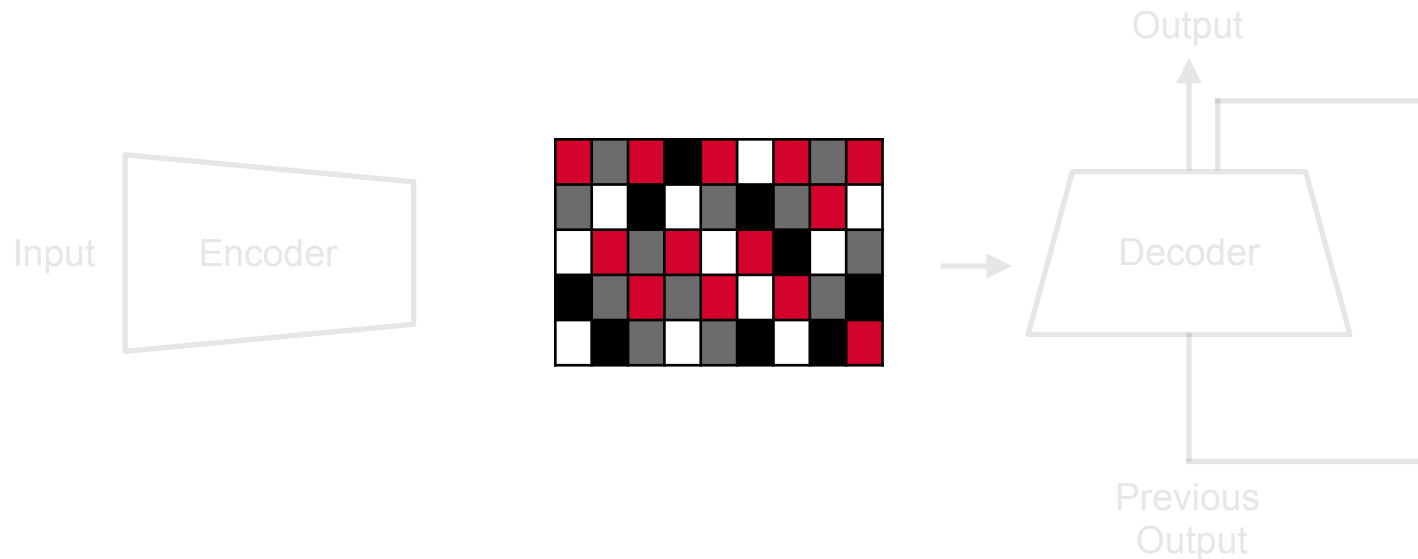
High-level Architecture of Transformers

Je suis Directeur

I am a manager → Encoder → Decoder

Encoder Creates Abstract Representation Fed to the Decoder

An abstract and highly contextualized representation of the input

Output

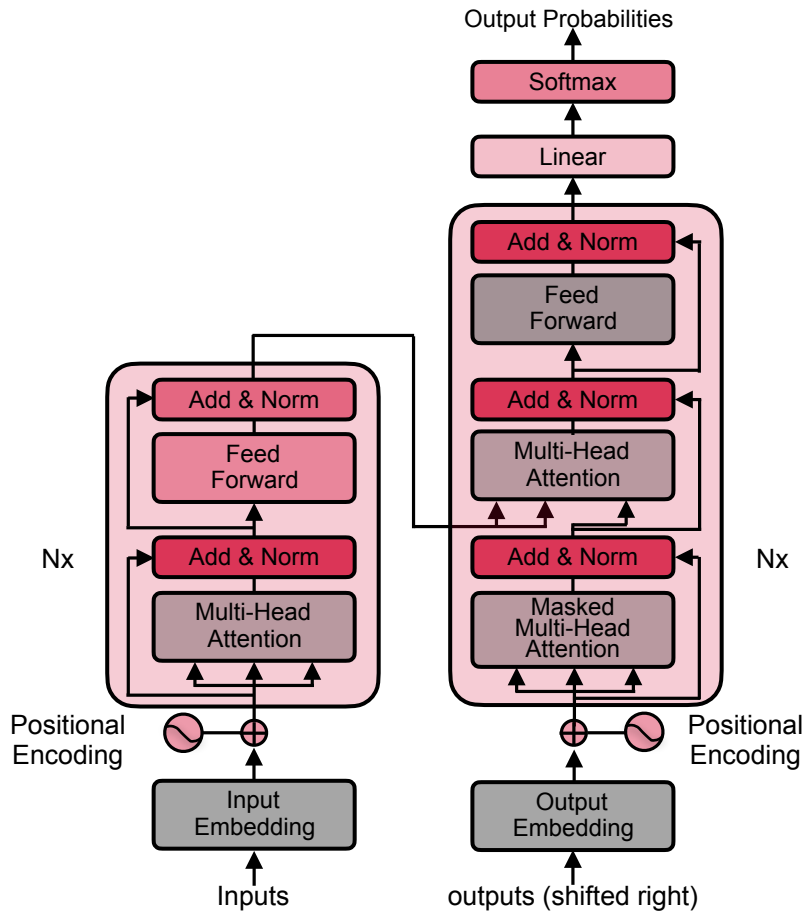Input

Encoder

Decoder

Previous Output

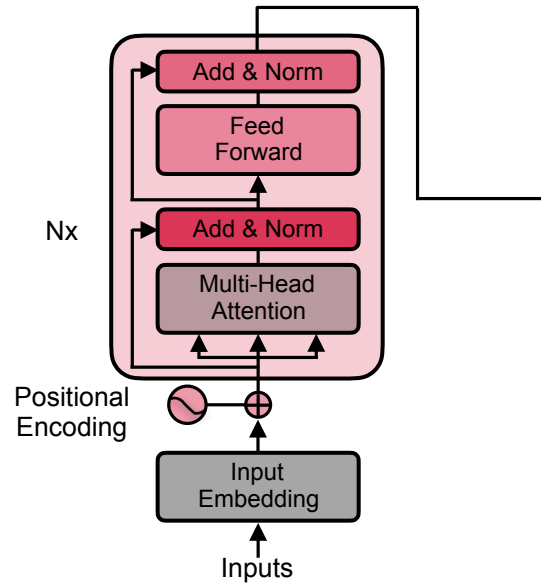# Encoder Creates Abstract Representation Fed to the Decoder

This representation encodes the importance of every word with respect to other words in the same input sequence
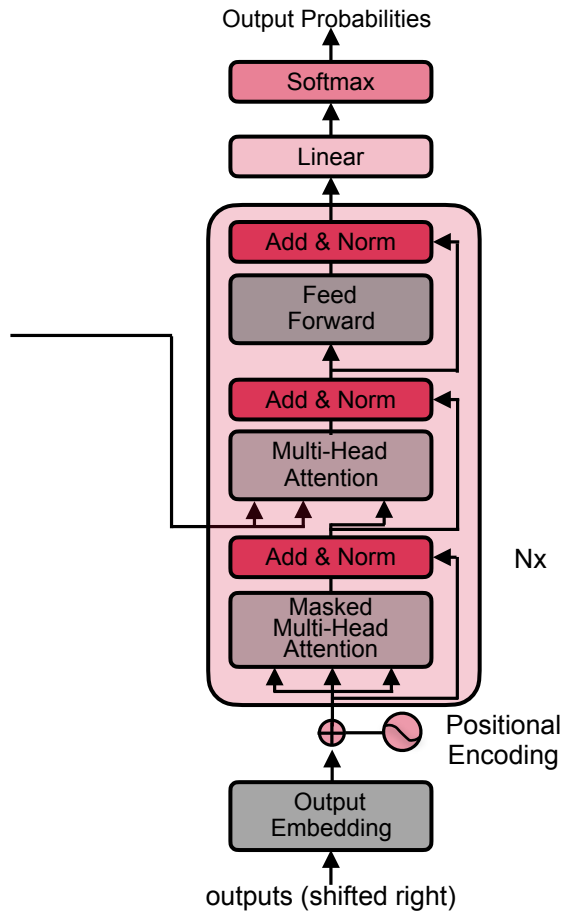
Transformer Architecture

Encoder

Add & Norm

Feed
Forward

Nx

Add & Norm

Multi-Head
Attention

Positional
Encoding

Input
Embedding

Inputs

Decoder

Output Probabilities

Softmax

Linear

Add & Norm

Feed Forward

Add & Norm

Multi-Head Attention

Add & Norm

Masked Multi-Head Attention

Nx

Positional Encoding

Output Embedding

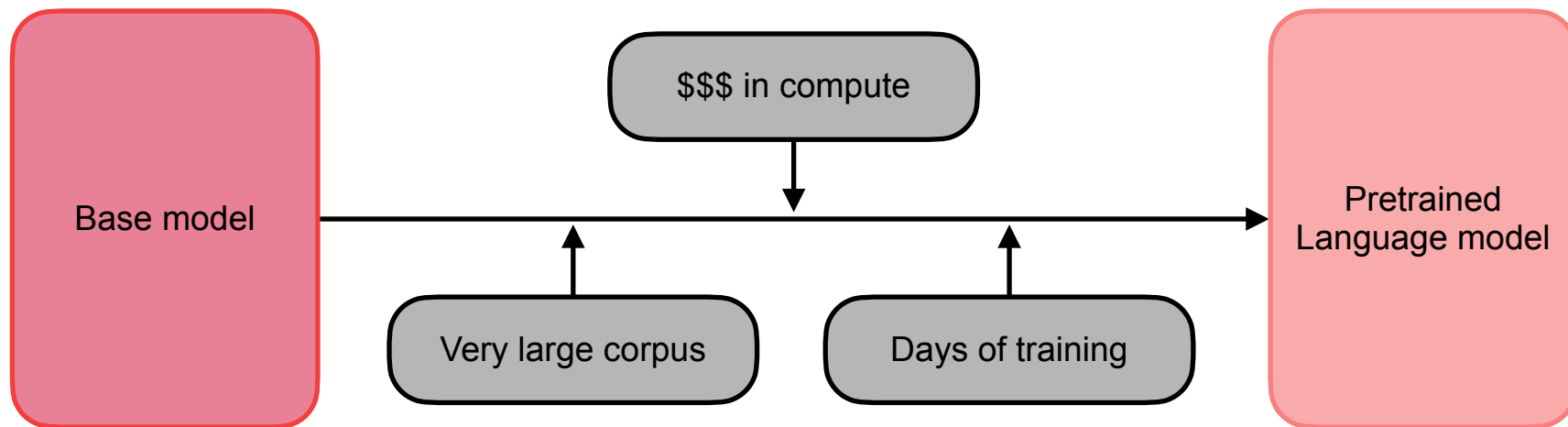outputs (shifted right)

Transformers in Hugging Face

Pretrained Transformers in Hugging Face

Transformers are very larger models and very expensive to train – in terms of data, time, and resources
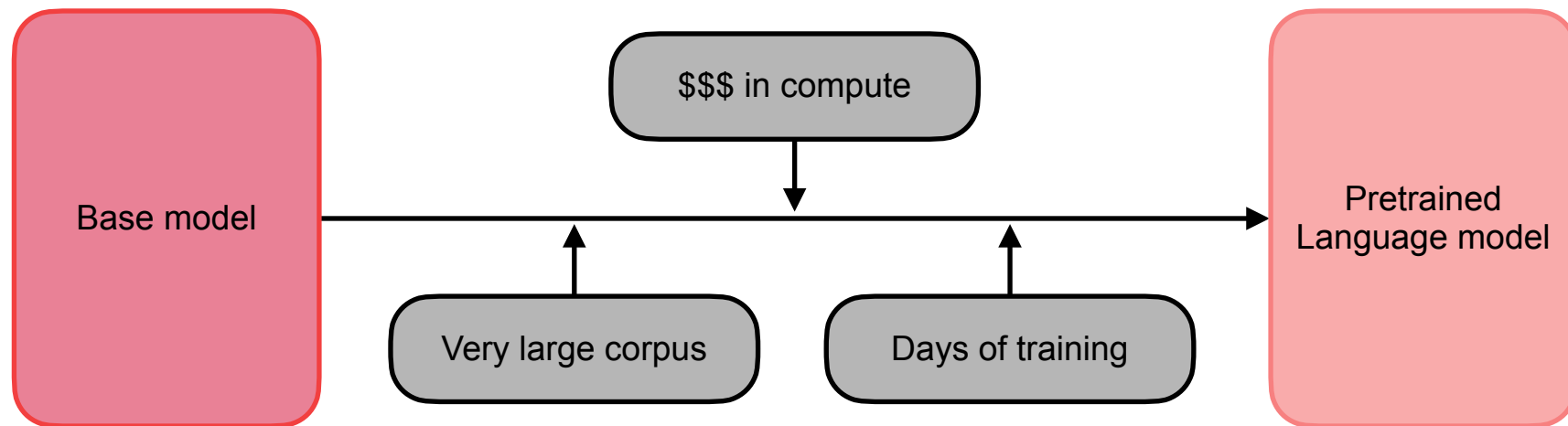
Training Transformers from Scratch

Base model → Pretrained Language model

$$$ in compute

Very large corpus

Days of training

Training transformer models from scratch (pretraining) - model weights are initialized at random and the model has no prior knowledge embedded within it
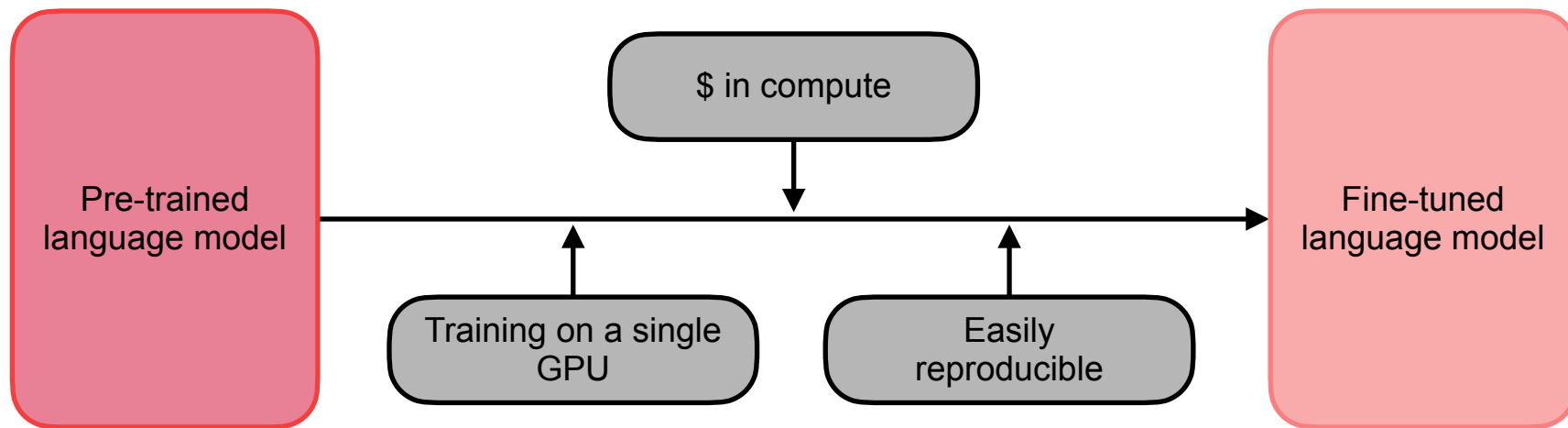
Training Transformers from Scratch

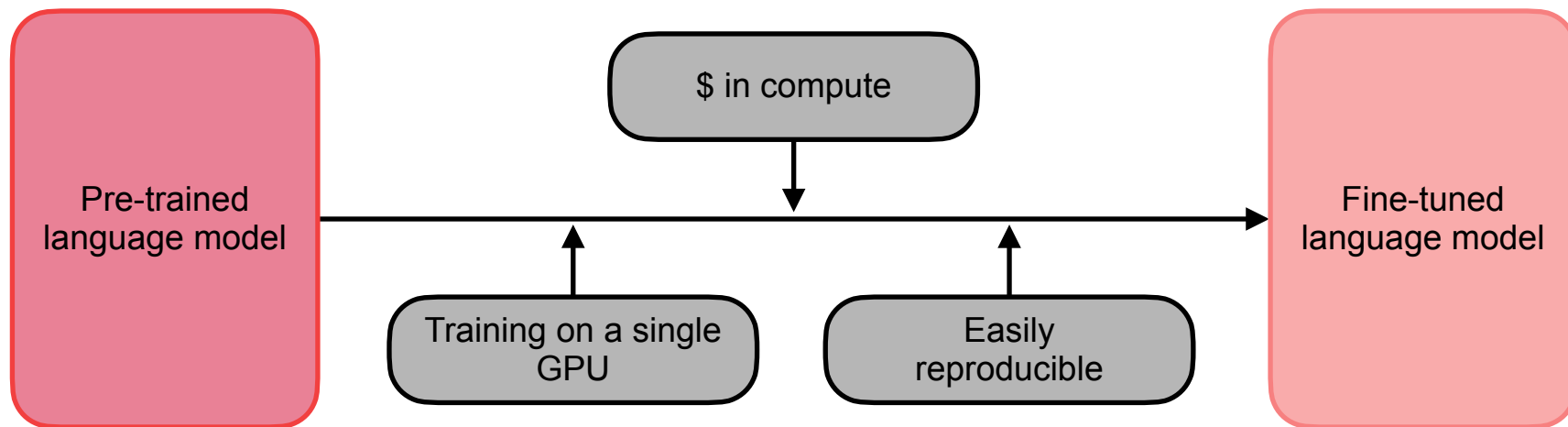Needs huge amounts of data and compute resources - can be very expensive

Fine-tuning Transformers

Pre-trained language model

$ in compute

Training on a single GPU

Easily reproducible

Fine-tuned language model

Fine-tuning models involve additional training on an already pretrained model

Fine-tuning Transformers

```
┌──────────────┐                                                    ┌──────────────┐
│              │              ┌──────────────┐                      │              │
│  Pre-trained │              │  $ in compute│                      │  Fine-tuned  │
│   language   │──────────────│──────────────│──────────────────────▶   language   │
│     model    │          ▲   └──────────────┘          ▲           │     model    │
│              │          │                              │           │              │
└──────────────┘   ┌──────────────┐          ┌──────────────┐       └──────────────┘
                   │  Training on │          │    Easily    │
                   │   a single   │          │ reproducible │
                   │     GPU      │          │              │
                   └──────────────┘          └──────────────┘
```

Use less data, time, and resources to get good results - overall costs and environment
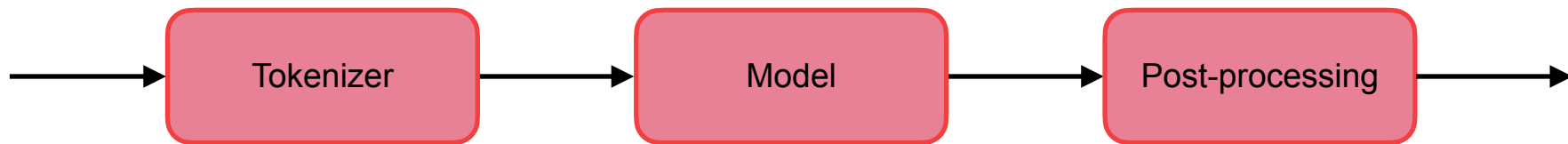impact of training is much lower than training from scratch

Hugging Face Pipelines

A high-level abstraction that makes it easy for users to perform NLP tasks using pre-trained models
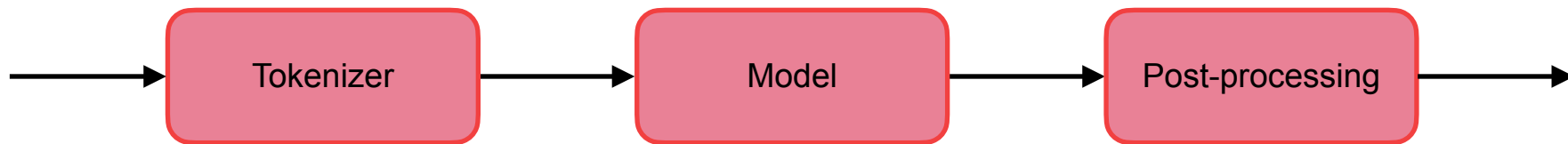
Pipeline Components

Hugging Face Tokenizers

O'REILLY®

Pipeline Components

Tokenizer

Tokenizing Text

Don't you know what time it is?

Split by Spaces

["Don't", "you", "know", "what", "time", "it", "is?"]

Consider Punctuation

["Don", """, "t", "you", "know", "what", "time", "it", "is", "?"]

Split Words Like Won't and Don't

["Do", "n't", "you", "know", "what", "time", "it", "is", "?"]

Tokenizers

- Rule-based tokenizers with different rules split data in different ways
- Pretrained models only work well with **inputs tokenized in the same manner as its training data**

Categories of Tokenizers

Word Tokenizers

Character Tokenizers

Subword Tokenizers

Word and Character Tokenizers

- Split text into tokens in meaningful ways
- "swim" and "swimming" should be identified as related
- Reduce the number of unknown words
- Word tokenizers
  - Too large a vocabulary
  - Will not identify related words
- Character tokenizers
  - Large number of tokens for each sentence
  - Intuitively less meaningful

Subword Tokenization

- Relies on the principle that frequently used words should not be split but rare words should be split into subwords

- "emotionally" could be a rare word, split into "emotional" and "ly"

- Allow models to have a reasonable vocabulary size and yet learn meaningful context independent representations
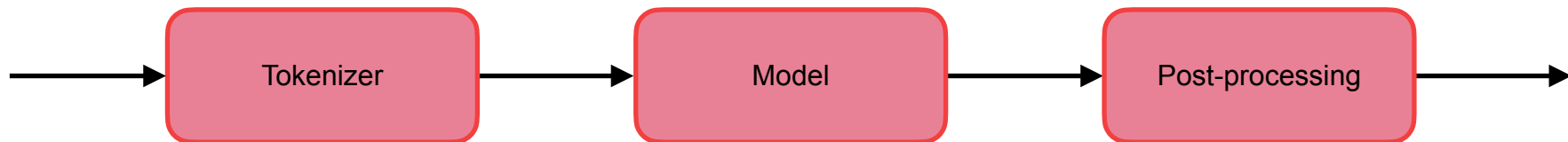
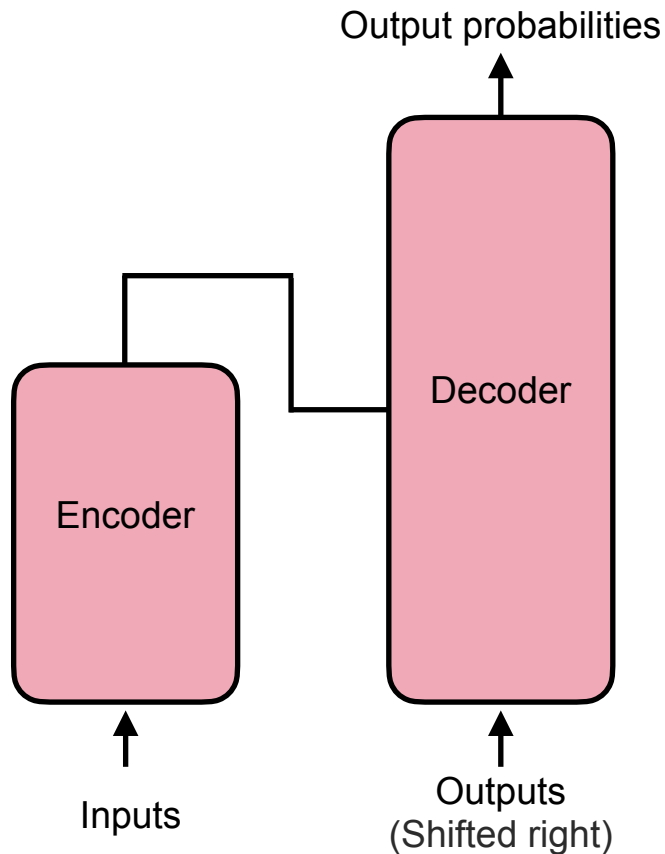# Hugging Face Pipelines

Pipeline Components

Hugging Face Transformer Pipeline

```
          ┌──────────────┐      ┌──────────────┐      ┌──────────────────┐
───────▶  │  Tokenizer   │ ───▶ │    Model     │ ───▶ │ Post-processing  │ ───▶
          └──────────────┘      └──────────────┘      └──────────────────┘
```

Raw text	⇒	Input IDs	⇒	Logits	⇒	Predictions

This course is amazing	⇒	[101, 2023, 2607, 2003, 6429, 999, 102]	⇒	[-4.3630, 4.6859]	⇒	POSITIVE: 99.89% NEGATIVE: 0.,11%

Hugging Face Transformer Pipeline

| Raw text | → | Input IDs | → | Logits | → | Predictions |

| This course is amazing | → | [101, 2023, 2607, 2003, 6429, 999, 102] | → | [-4.3630, 4.6859] | → | POSITIVE: 99.89% NEGATIVE: 0.,11% |

General Transformer Architecture

Output probabilities

Decoder

Encoder

Inputs

Outputs
(Shifted right)

Encoder

Output probabilities

Decoder

Encoder

Inputs

Outputs
(Shifted right)

Decoder

Output probabilities

Decoder

Encoder

Inputs

Outputs
(Shifted right)

Encoder-only Models

- Attention layers in the encoder can access all the words in the input sentence called "bi-directional" or "auto-encoding" models

- Pre-training involves corrupting the initial sentence (masking) and having the model predict the masked word

- Suited for tasks such as **sentence classification, named entity recognition, and question answering**

Masked Word Prediction

Let's have some fun doing _____ programming!

Python?
Java?
C++?

Decoder-only Models

- Attention layers in the decoder can only access words before the current word in the sentence – called "auto-regressive" models

- Pre-training of these models involve predicting the next word in the sentence, given all words seen so far

- Models are best suited for natural language tasks such as **text generation**

Next Word Prediction

It's a nice sunny afternoon. Let's go _____

outside
swimming
running
dancing

Encoder-Decoder Models

- Called sequence-to-sequence models
- Attention layers in the encoder can access all parts of the input sentence,
- Attention in the decoder can only access words positioned before the given word
- Pre-training usually done using the objectives of the encoder and decoder (masked word prediction and next word prediction)
- Suited for tasks such as **summarization, translation, and generative question answering**

Text Generation by LLMs

Model Responses to Prompts

Prompt
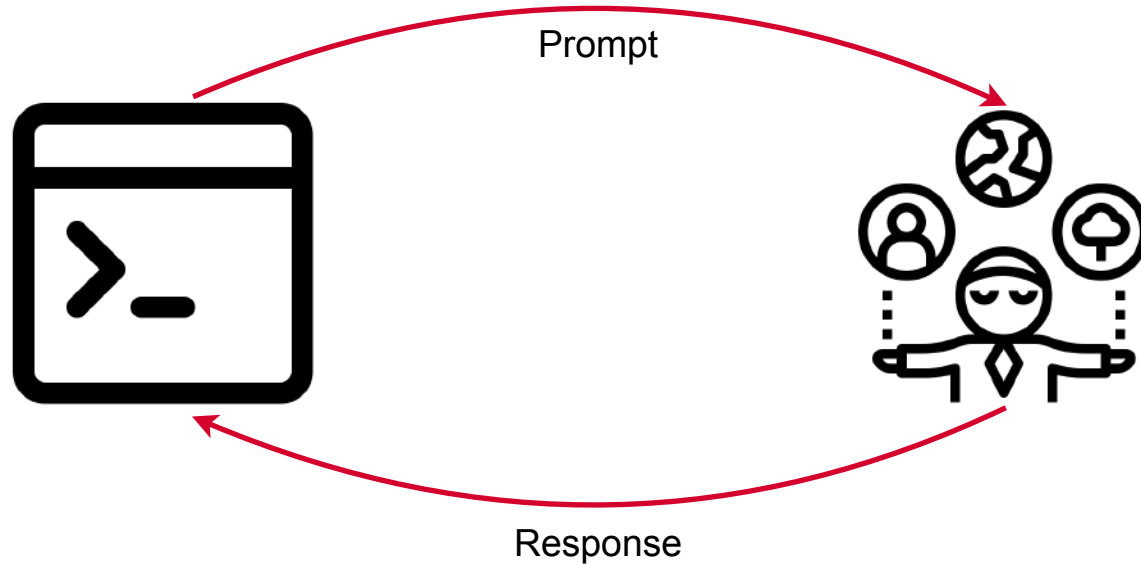
Response

Response a Sequence of Words

She speaks French quite well

Model Generates One Word at a Time

She ➡ speaks ➡ French ➡ quite ➡ well

Many Possible Words at Each Step

She → speaks →

English

French

Spanish

Hindi

Italian

Chinese

Python
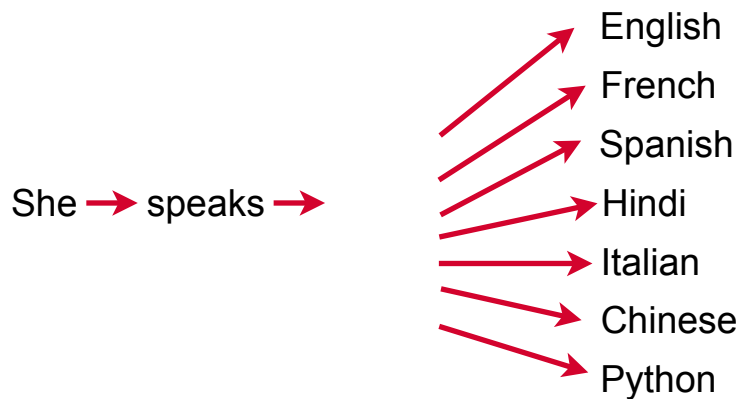
Each Possible Next Word is Assigned a Probability

She → speaks →

English - 0.4
French - 0.3
Spanish - 0.1
Hindi - 0.08
Italian - 0.05
Chinese - 0.04
Python - 0.03

The Model Picks One Word from Possible Next Words

She → speaks →

English - 0.4
French - 0.3
Spanish - 0.1
Hindi - 0.08
Italian - 0.05
Chinese - 0.04
Python - 0.03
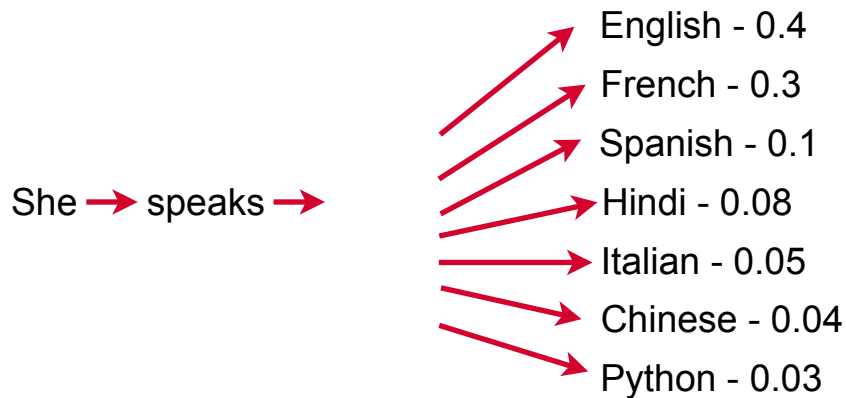
Higher Probability Words are More Likely to be Picked

She → speaks →

English - 0.4
French - 0.3
Spanish - 0.1
Hindi - 0.08
Italian - 0.05
Chinese - 0.04
Python - 0.03
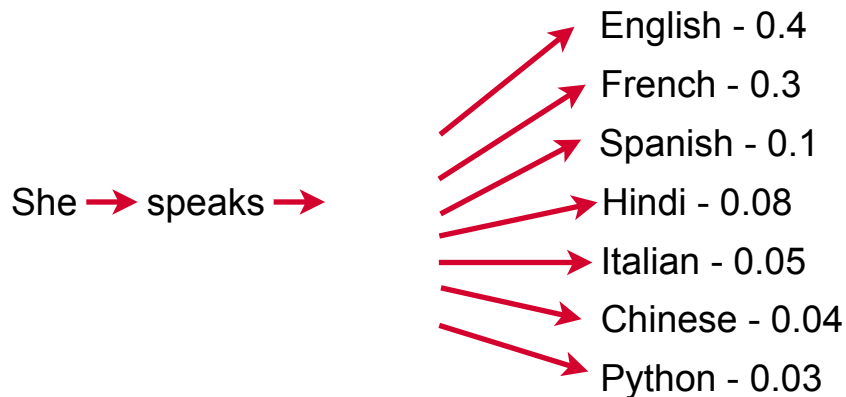
Response Generated by Picking Words at Each Step

She ➡ speaks ➡ French ➡ quite ➡ well

Model Settings for Tweaking Generated Text

Large language models offer settings that you can tweak to make the generated text **more creative and diverse or more predictable and deterministic**

Creativity vs. Predictability in Text Generation

- High creativity will produce more diverse and unexpected results making the text more engaging

- High predictability generates more consistent and reliable text – useful when you need precise responses

- Striking a balance can produce text that is both interesting and coherent

Model Settings to Control Creativity and Predictability

- Temperature
- Top-p (Nucleus Sampling)
- Top-k

Temperature

- Values range between 0 and 1 (both inclusive)
- Higher values closer to 1 results in more creative output
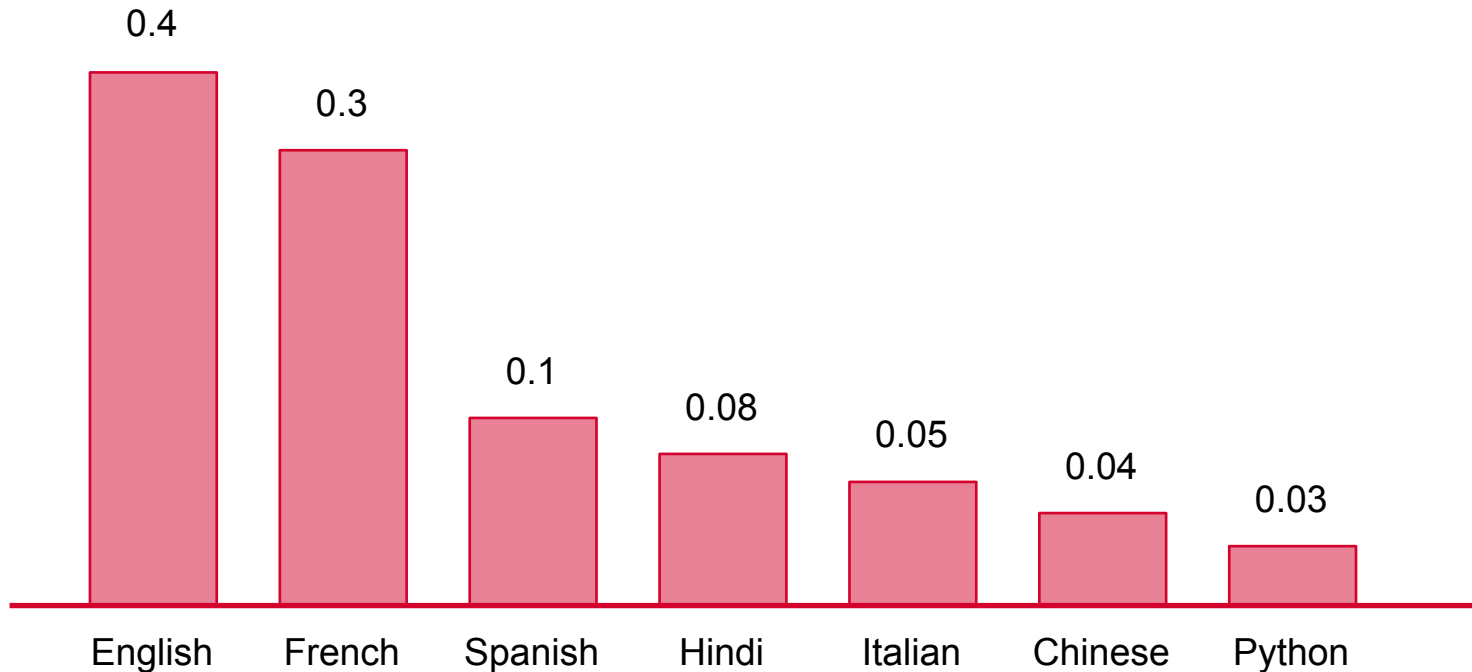- Lower values closer to 0 results in more predictable output

The Model Picks One Word from Possible Next Words

She → speaks →

English
French
Spanish
Hindi
Italian
Chinese
Python
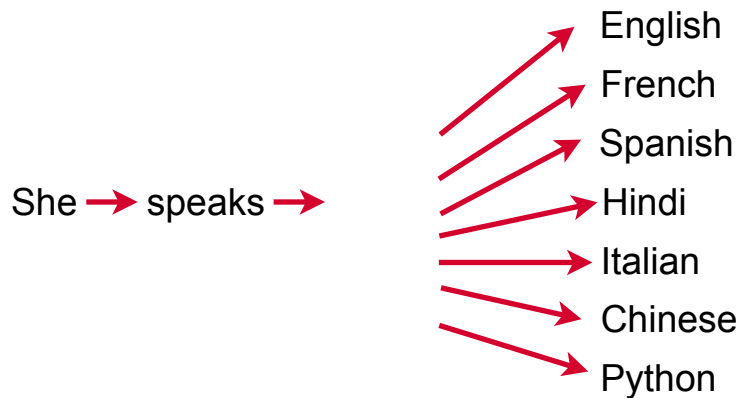
Original Probabilities of Possible Next Words
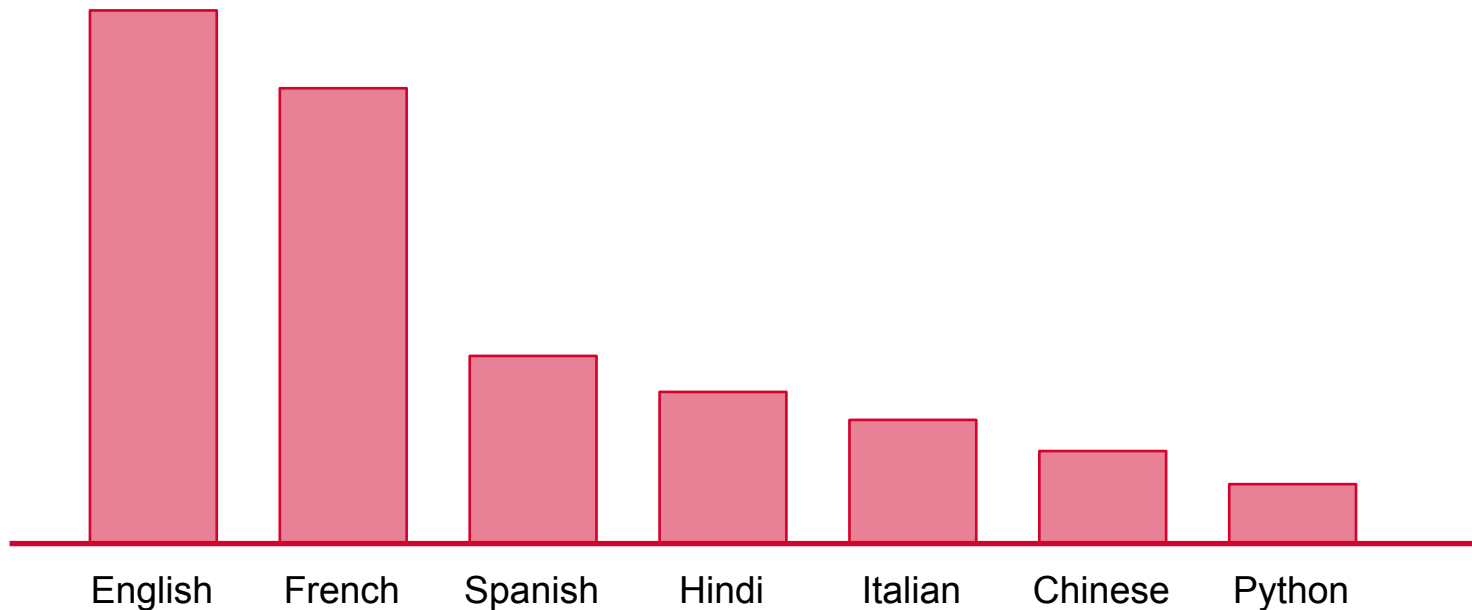
Higher Values of Temperature (Closer to 1)
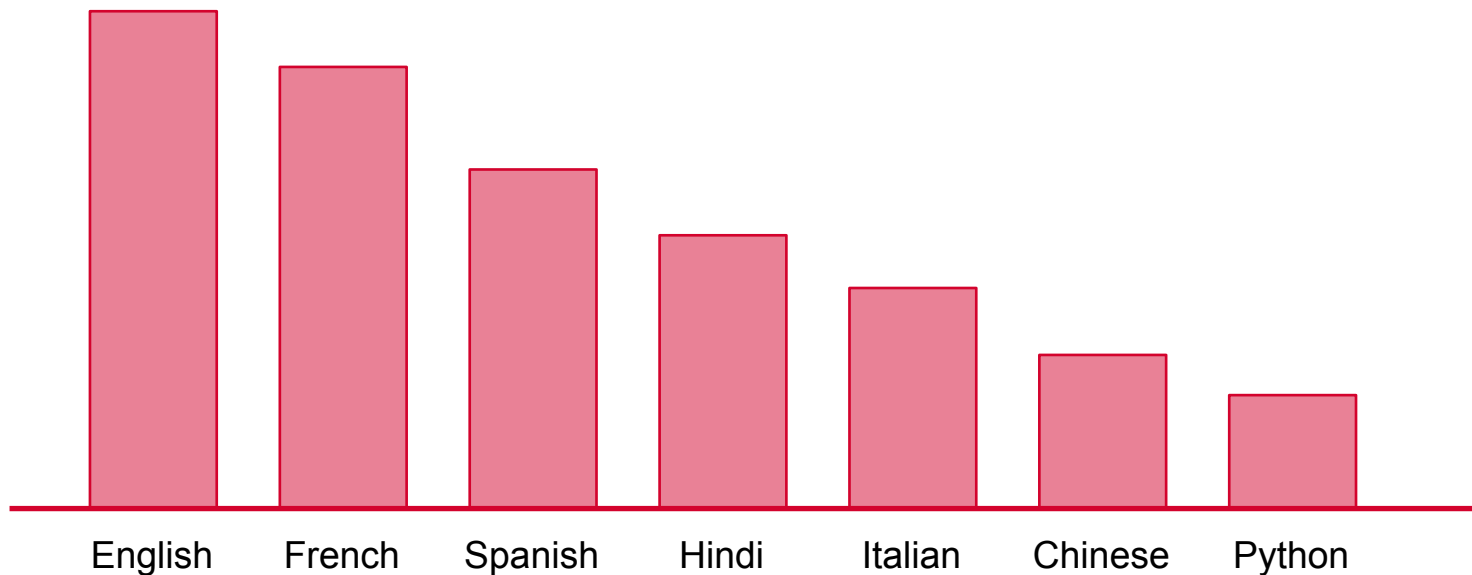
The probability distribution over the next possible word becomes **flatter.**

Original Probabilities of Possible Next Words
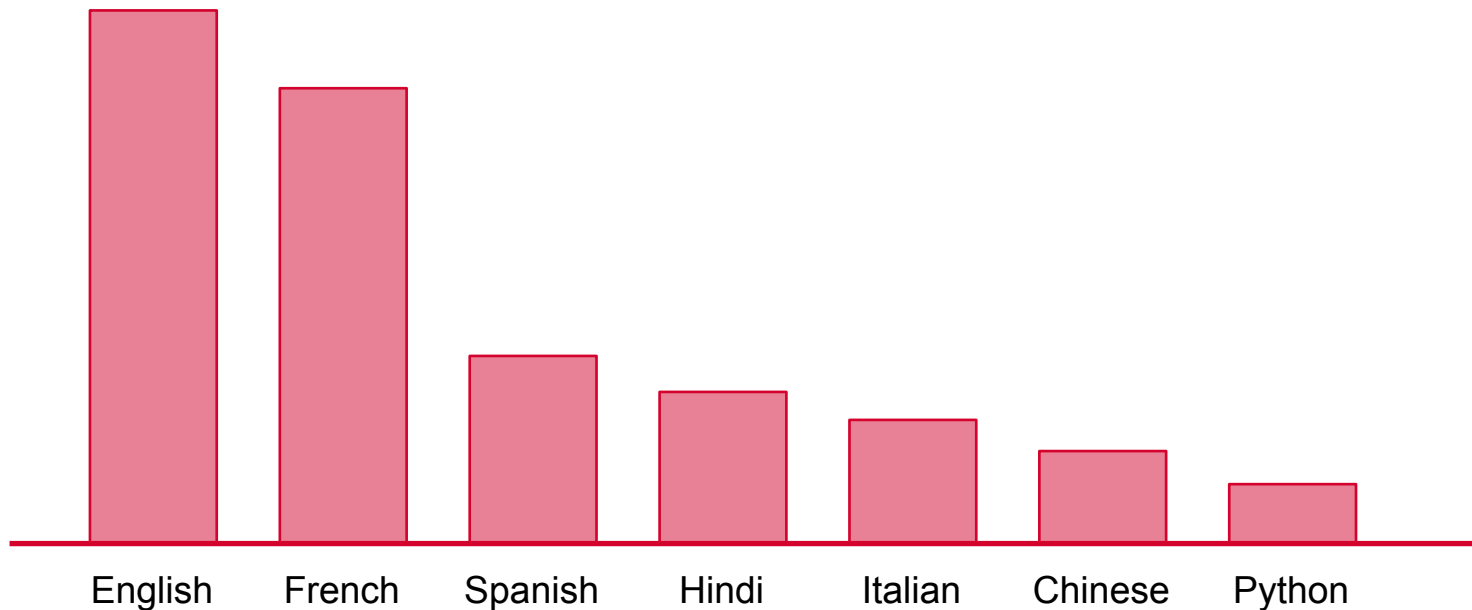
Lower Values of Temperature (Closer to 0)
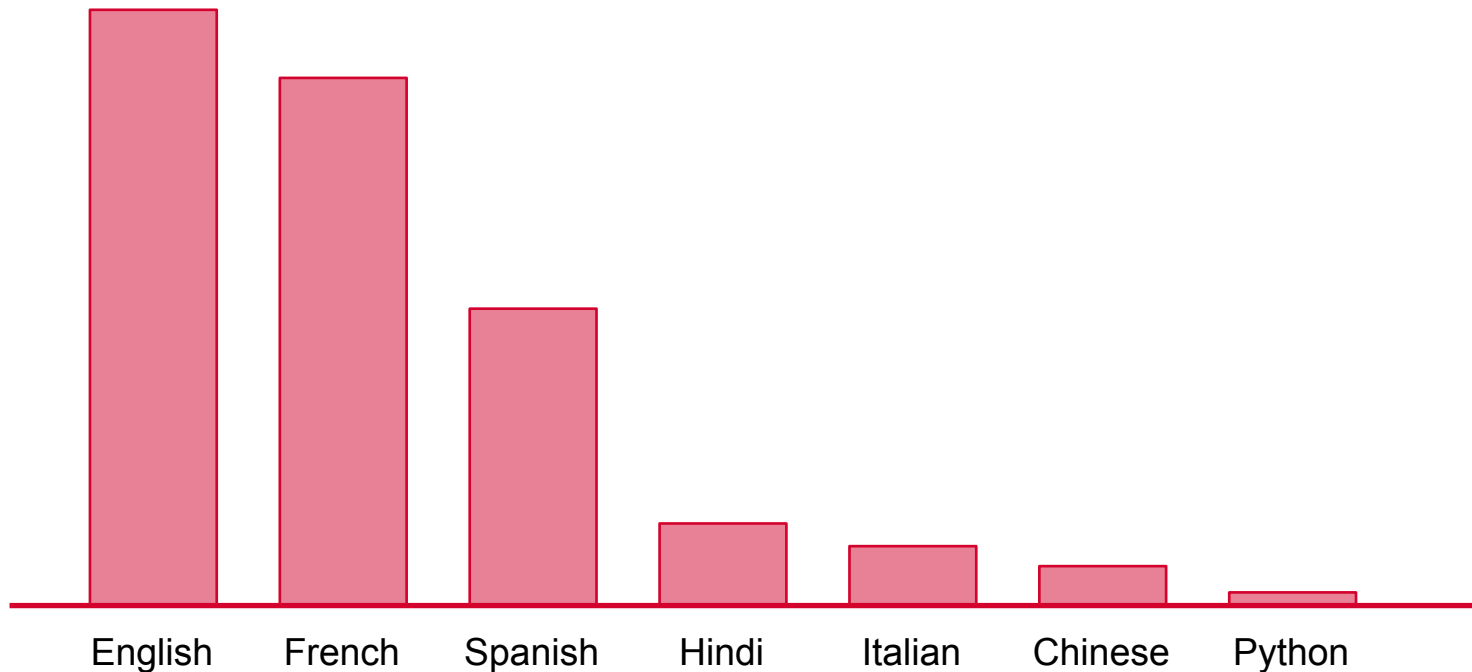
The probability distribution over the next possible word becomes **sharper.**

Top-p (Nucleus Sampling)

- Values range between 0 and 1 (both inclusive)
- Values close to 1 result in more diverse and creative output
- Values close to 0 result in more predictable output

Top-p (Nucleus Sampling)

Top-p sampling, also known as nucleus sampling, works by selecting the **smallest set of top candidate words whose cumulative probability exceeds a given threshold p.**

Probabilities of Possible Next Words

Sort All Words in Order of Probability
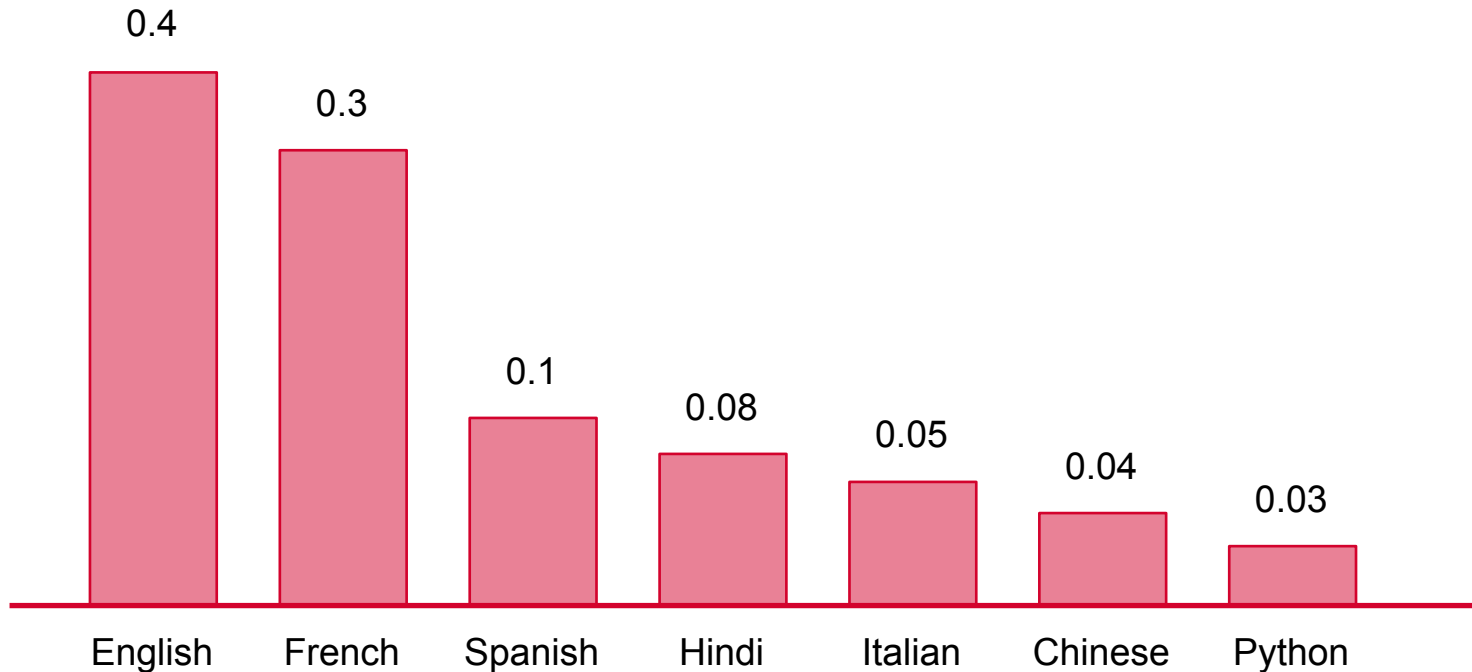
Top-p of 0.5
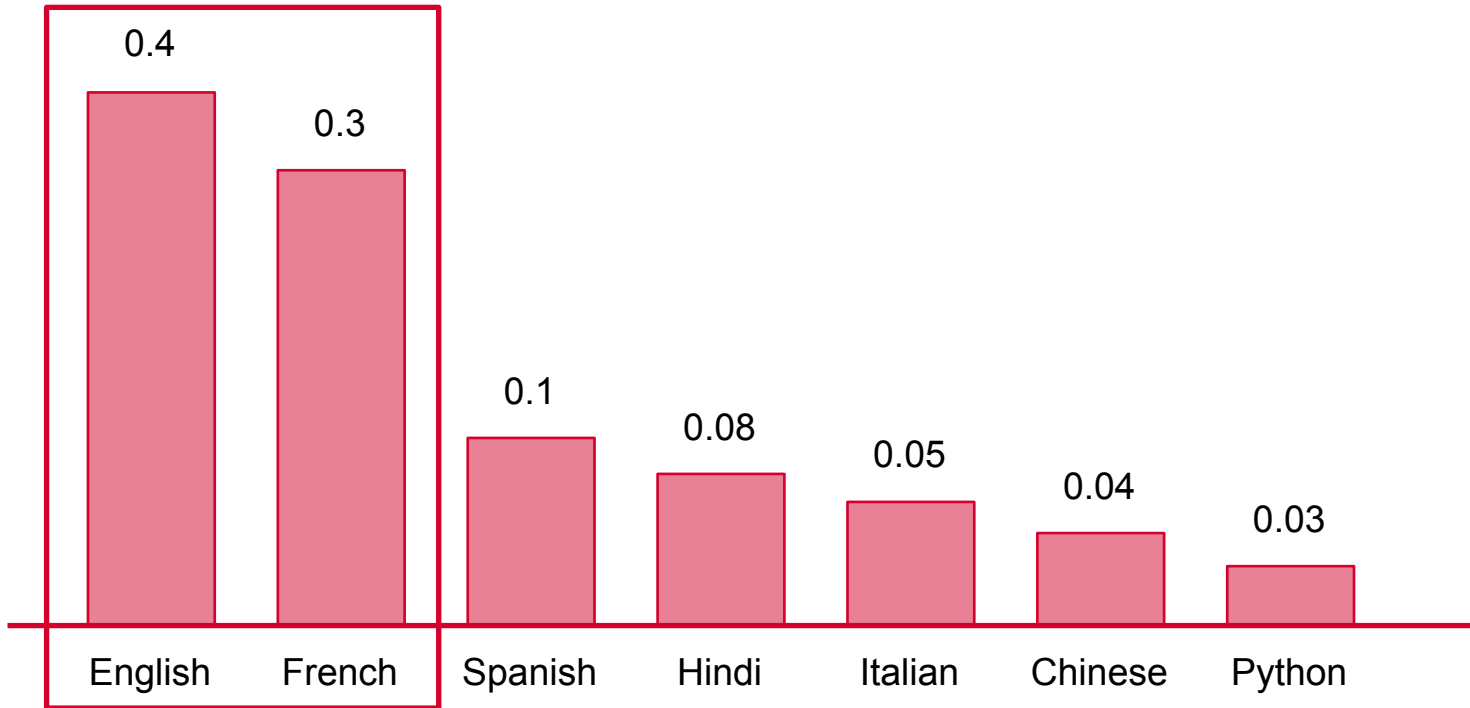
The next word selected will only choose between the smallest subset
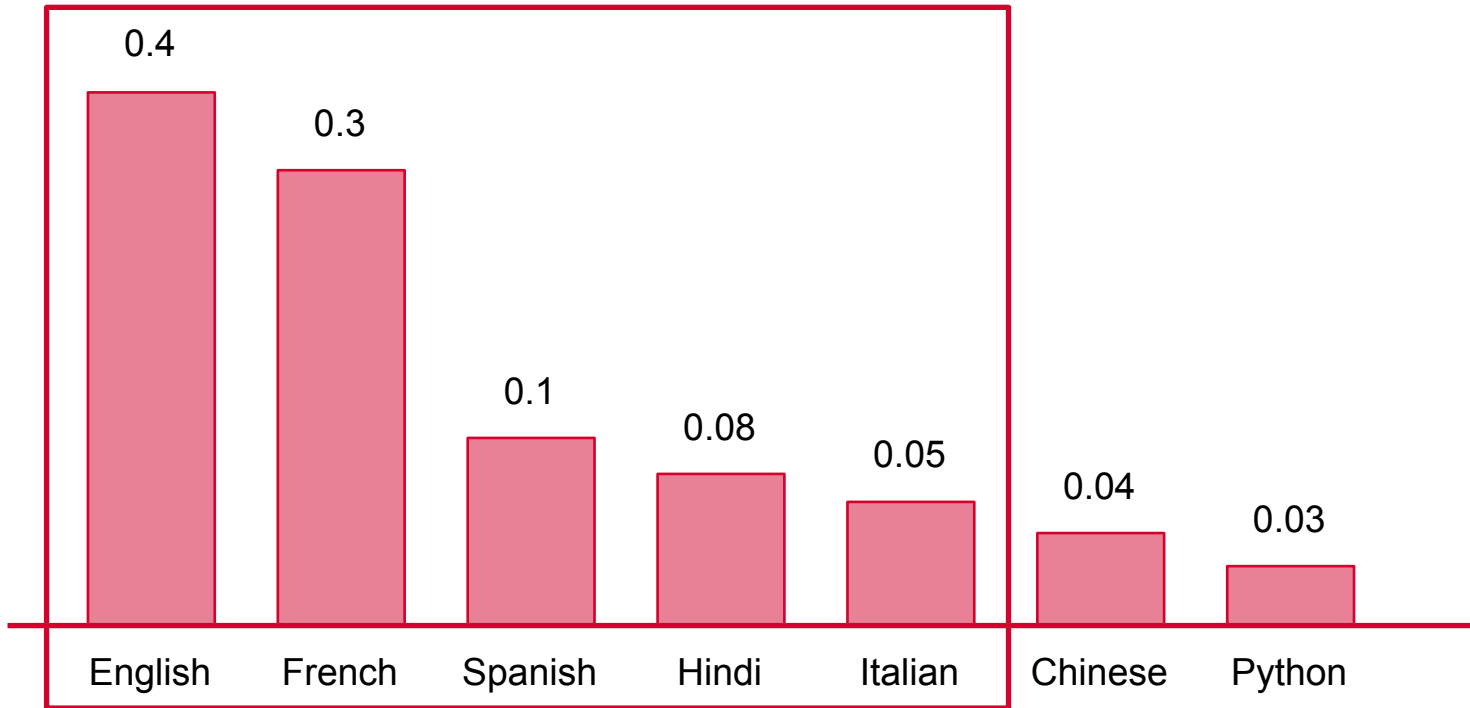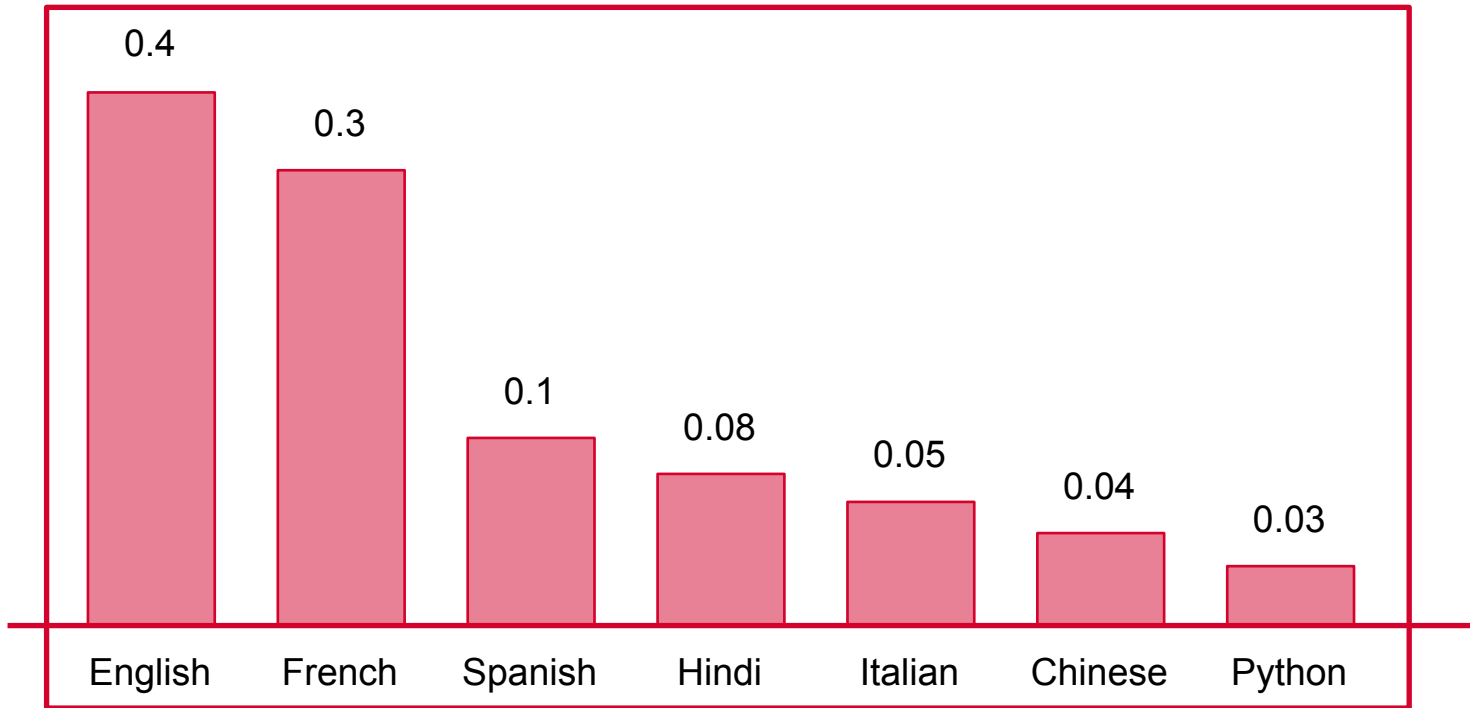of words that exceeds the cumulative probability threshold

Top-p of 0.7
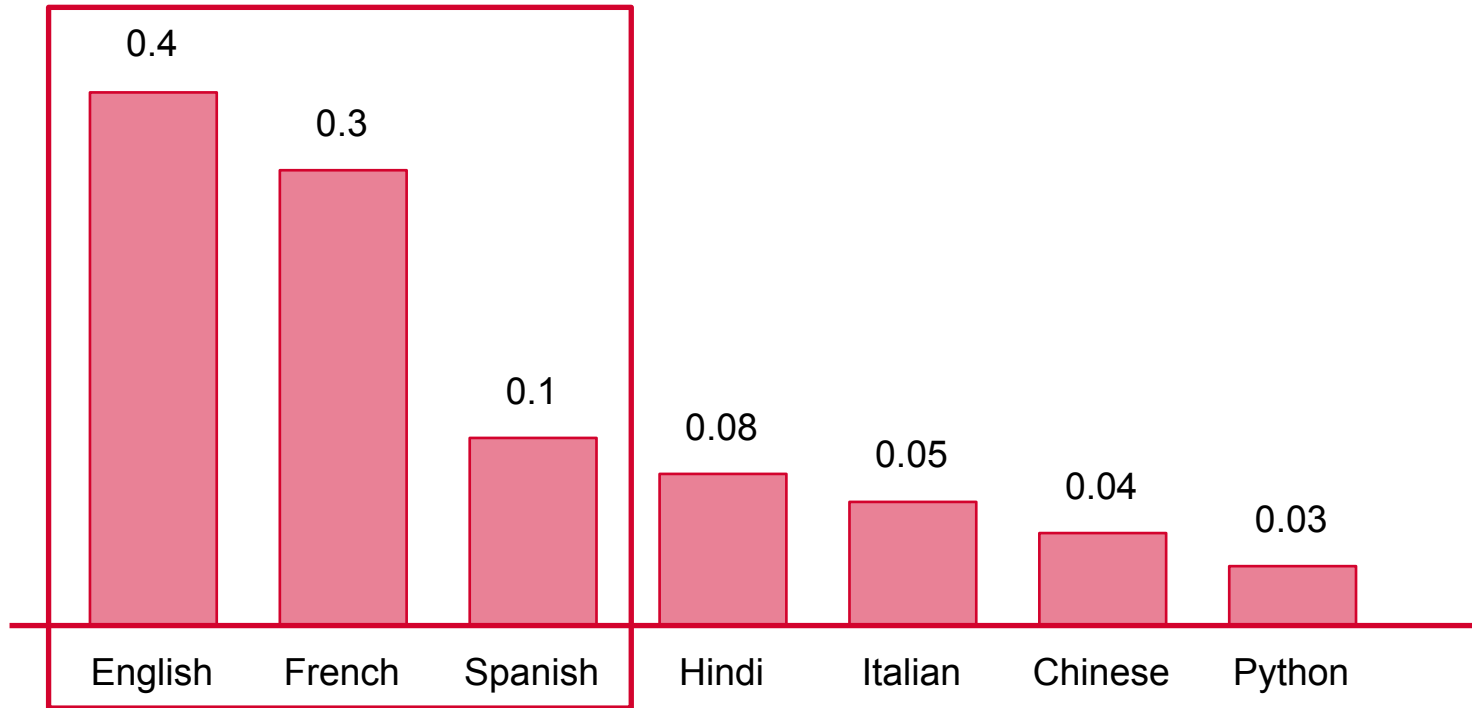
Top-p of 1

Choose from among all possible words, greatest
possible diversity

Top-k Sampling

- Words to choose from limited to the K most likely words at each step
- Value is any positive integer
- Smaller values will pick more likely words, so less diverse output
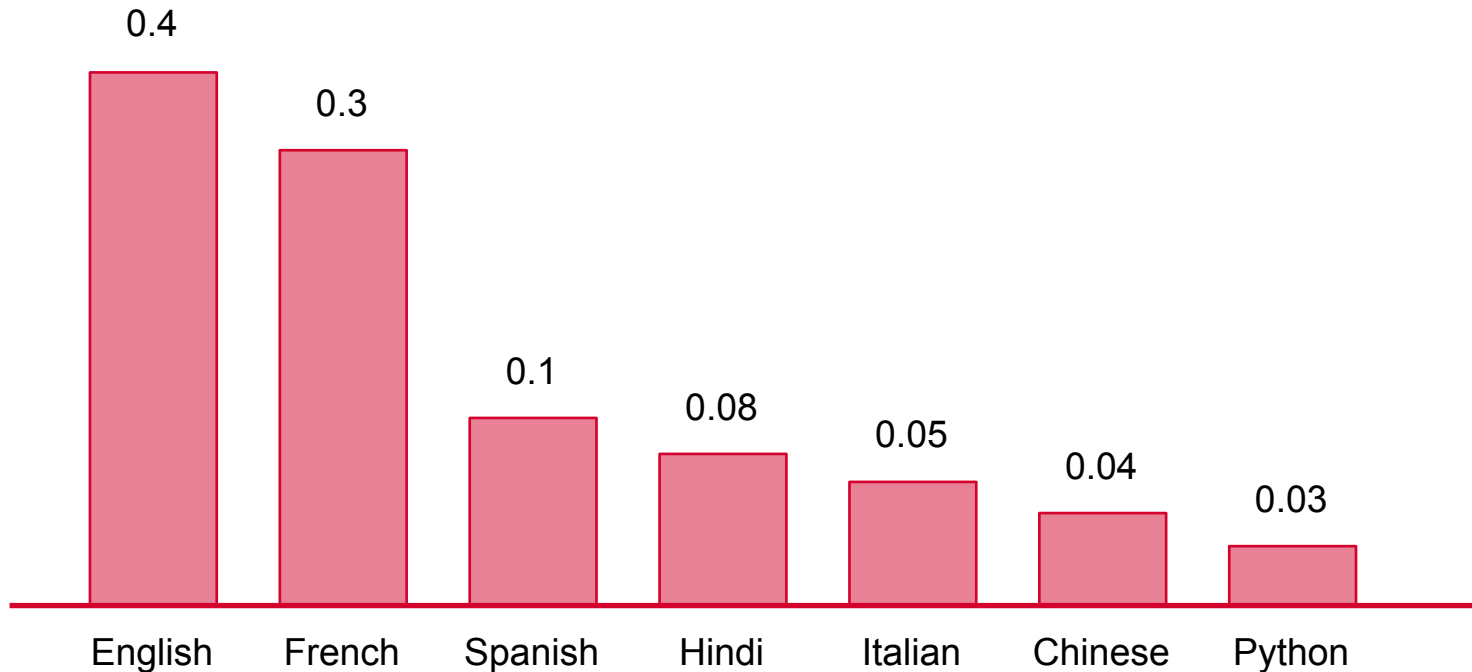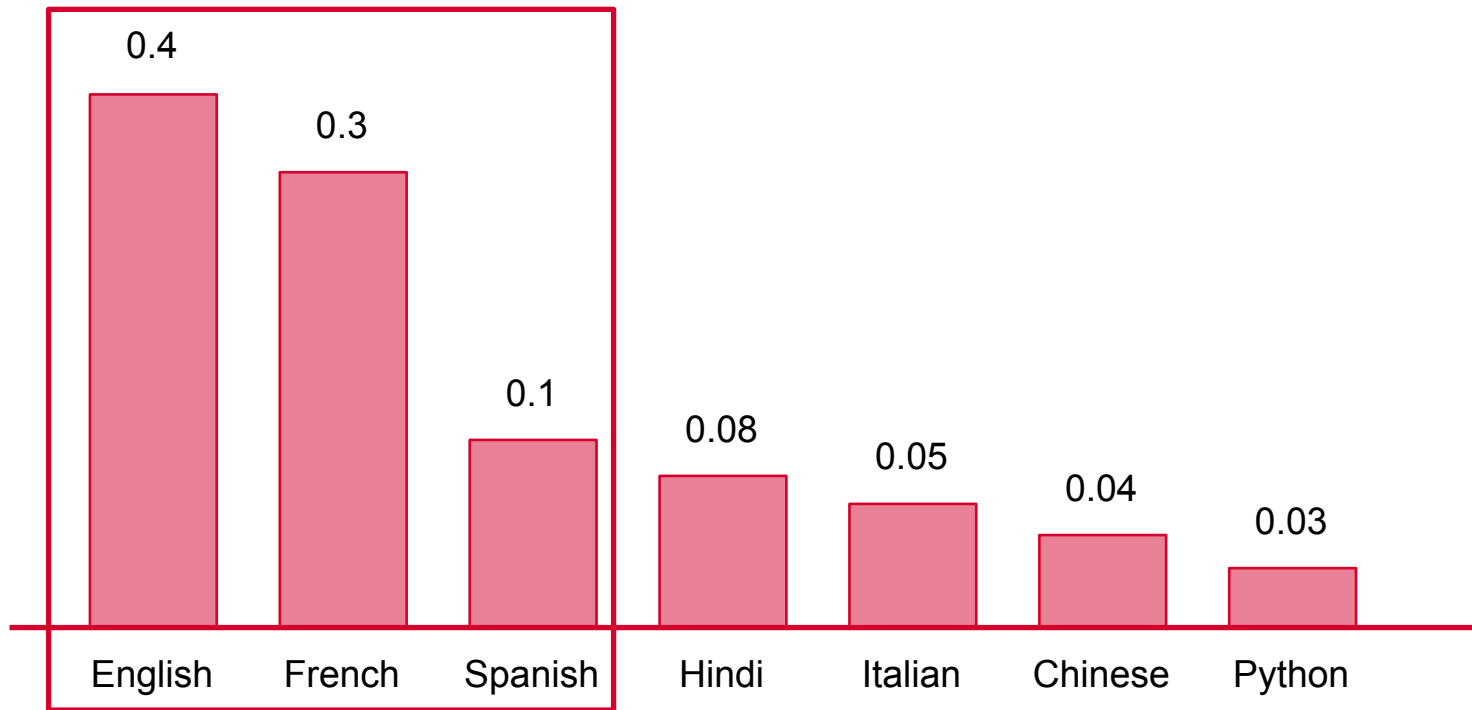- Larger values will pick less likely words, more diverse output

Top-k Sampling

Top-K sampling limits the next-word selection to the **K most likely words** predicted by the model at each generation step.
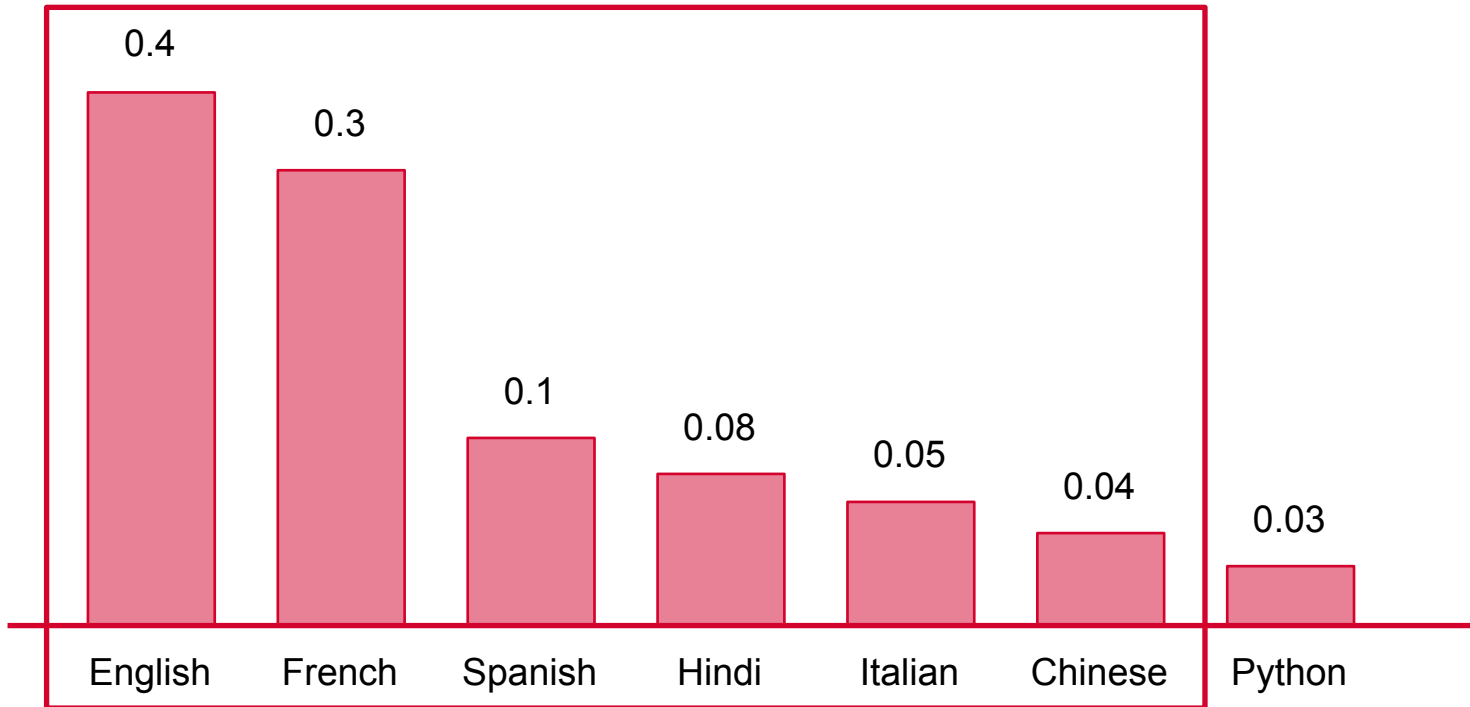
Sort All Words in Order of Probability

Top-k of 3

The next word selected will only choose between the top 3 words in terms of probability

Top-k of 6

The next word selected will only choose between the top 6 words in terms of probability