Language Models (Part 2)

Intro to ML

24 November 2022

Note

- Modern language models can be quite complex
- This lecture is to give a general introduction into where we are “now”
- This means a lot of details have to be skipped over
- I want to convey general understanding

If something is unclear, **please ask for a clarification**.

You’re probably not the only one in the class wondering the same thing.

What is a Language Model?

A probability distribution over strings according to a model

```
P(string | model)
```

Corpus (plural ***corpora***): a collection of written texts

String:

- Word sequence
- Character sequence
- Letter triplets
- Can be most things!

```
P("Edmonton winters are enjoyable" | model)
P("Enjoyable Edmonton are winters" | model)
P("Les hivers d'Edmonton sont plaisants" | model)
```

Language models are defined according to their model and associated model parameters.
Calculation of this value depends on the model, here it's not specified.

Review Quiz

Which, out of a bigram and trigram model, would you expect to have the *highest* **perplexity** on a text corpus?

What problems do word vectors suffer from when the same written word can have multiple meanings?

What is the “**Distributional Hypothesis**”?

What applications do Language Models power?

- Text classification (i.e. semantic analysis)
- Named Entity Recognition (NER)
- Question Answering (QA)
- Text Summarisation & Translation
- Language Understanding (Alexa, Google Home, Cortana etc.)
- Text generation
- Search (i.e. Google)
- Image captioning
- Image generation (DALLE2, Stable Diffusion, Midjourney)

Where are language models used?

Where are language models used?

BLOG ›

Recent Advances in Google Translate

MONDAY, JUNE 08, 2020

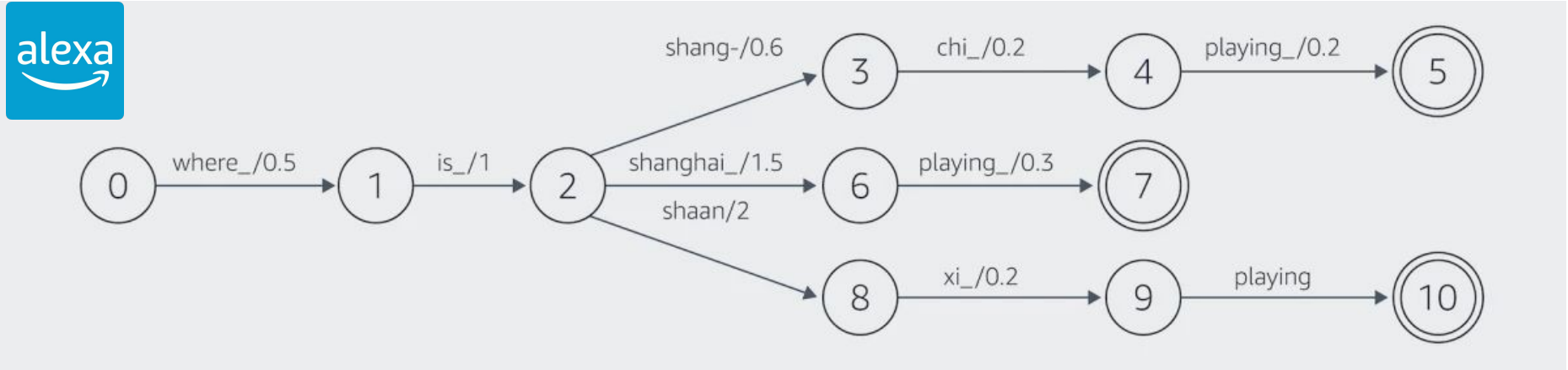
Posted by Isaac Caswell and Bowen Liang, Software Engineers, Google Research

Hybrid Model Architecture

Four years ago we introduced the RNN-based GNMT model, which yielded large quality improvements and enabled Translate to cover many more languages. Following our work decoupling different aspects of model performance, we have replaced the original GNMT system, instead training models with a transformer encoder and an RNN decoder, implemented in Lingvo (a TensorFlow framework). **Transformer models have been demonstrated to be generally more effective at machine translation than RNN models**, but our work suggested that **most of these quality gains were from the transformer *encoder*, and that the transformer *decoder* was not significantly better than the RNN decoder.** Since the RNN decoder is much faster at inference time, we applied a variety of optimizations before coupling it with the transformer encoder. The resulting hybrid models are higher-quality, more stable in training, and exhibit lower latency.

Where are language models used?

"*Our on-device ASR model takes in an acoustic speech signal and outputs a set of hypotheses about what the speaker said, ranked according to probability.*"



"*With on-device ASR [Automatic Speech Recognition], only the lattice is sent to the cloud, where a large and powerful neural **language model** reranks the hypotheses.*"

https://www.amazon.science/blog/on-device-speech-processing-makes-alexa-faster-lower-bandwidth

Where are language models used?

- Trained on open source code
- Built on top of OpenAI's **Codex** model
- OpenAI & Microsoft's joint product
- Codex is a **descendent of GPT-3**
- Type a description in English of what function / code you want to write as a comment and GHCP will create it

GitHub Copilot

Where are language models used?

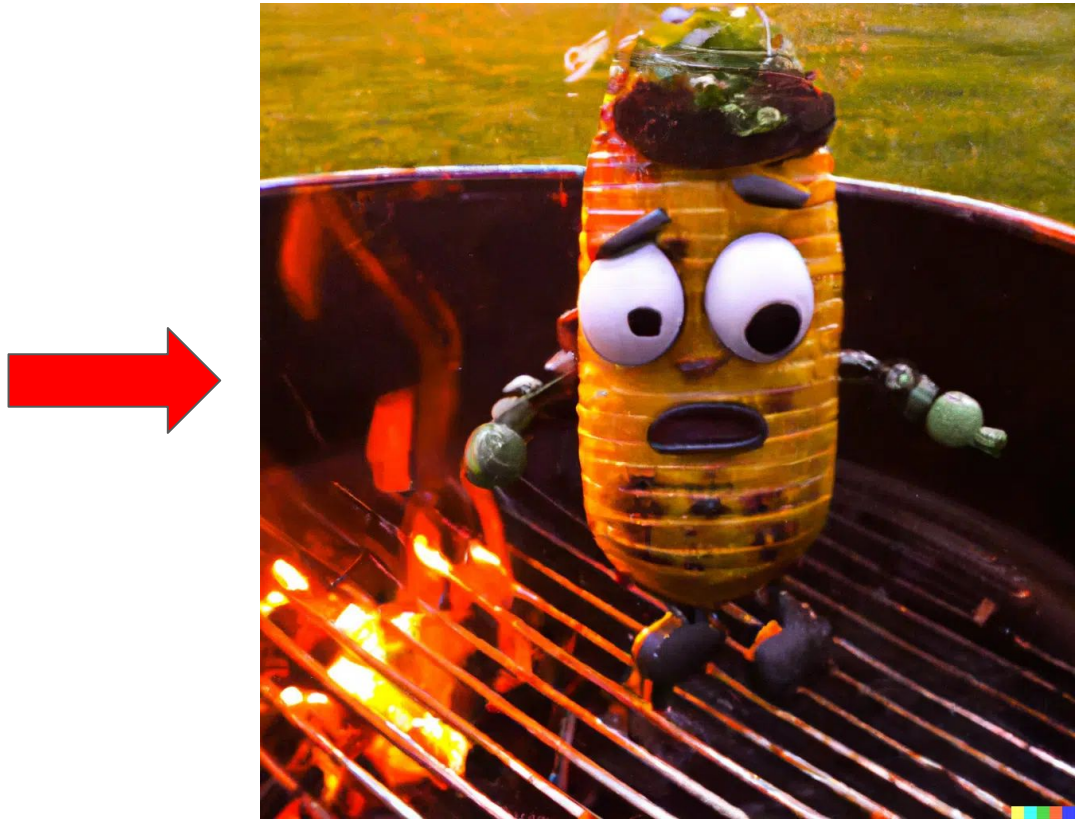
In tandem with image generation methods such as:

- DALLE-2
- Stable Diffusion
- Midjourney

Where are language models used?

Input prompt:

- A BBQ that is alive, in the style of a Pixar animated movie



Where are language models used?

Input prompt:

- A hamburger in the shape of a Rubik's cube, professional food photography

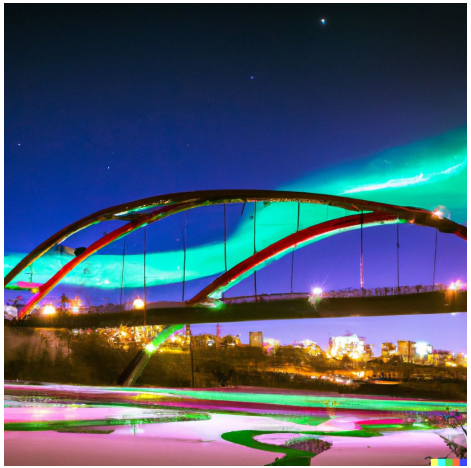
Where are language models used?

Input prompt:

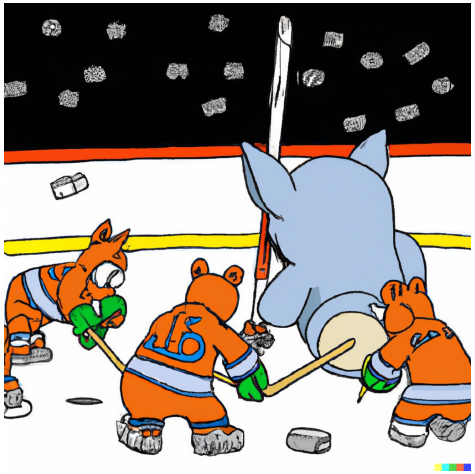
- A rabbit detective sitting on a park bench and reading a newspaper in a Victorian setting

Midjourney Generates AI Apocalyptic Images of the 'Last Selfie Ever Taken'

AUG 01, 2022 MATT GROWCOOT



“*Dramatic Northern Lights shining in the sky over the Edmonton Walterdale bridge, over a frozen river. Stylistic. Colourful.*”



“Edmonton Oilers hockey team playing a game against giant rabbits.”

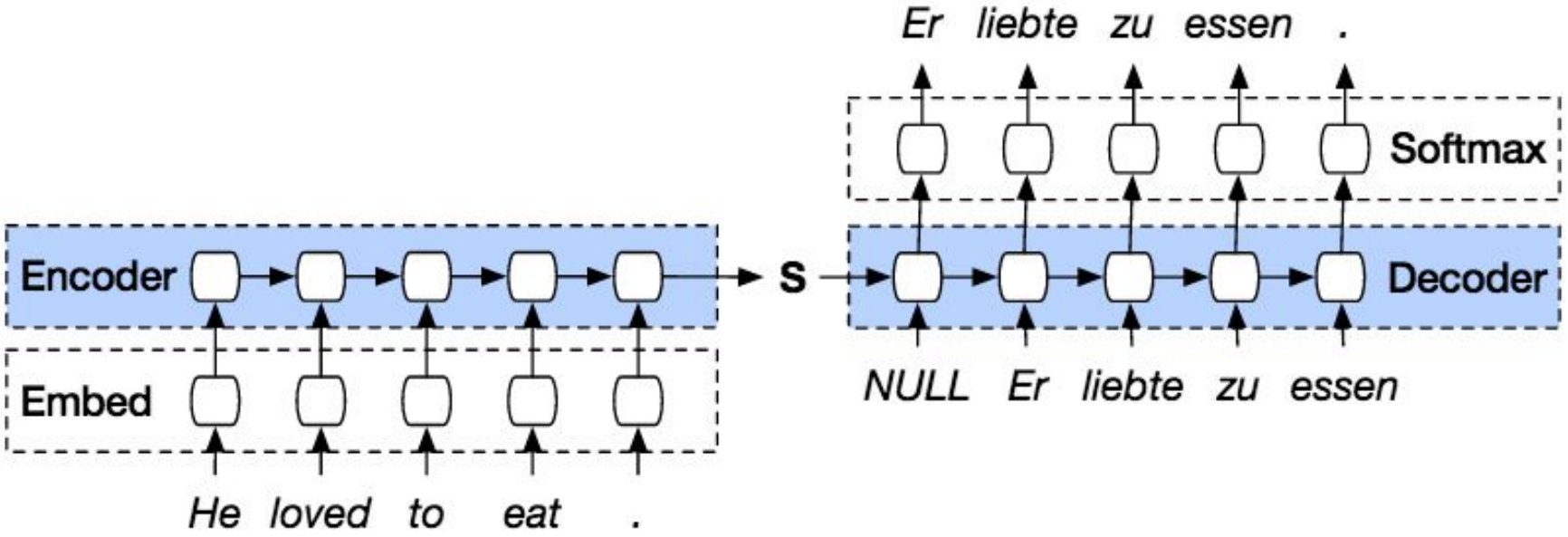
Where are language models used?

Everywhere!

The ability to represent, in numerical vectors, complex ideas that can be expressed in intricate language use has transformed our ability to interact with computers and technology.

As the last DALLE-2 images show, there is still work that needs to be done.

“Attention” (Recap)



“Attention”

Input: Hospitals are places where people get treated

Representation (in an RNN) **without attention**

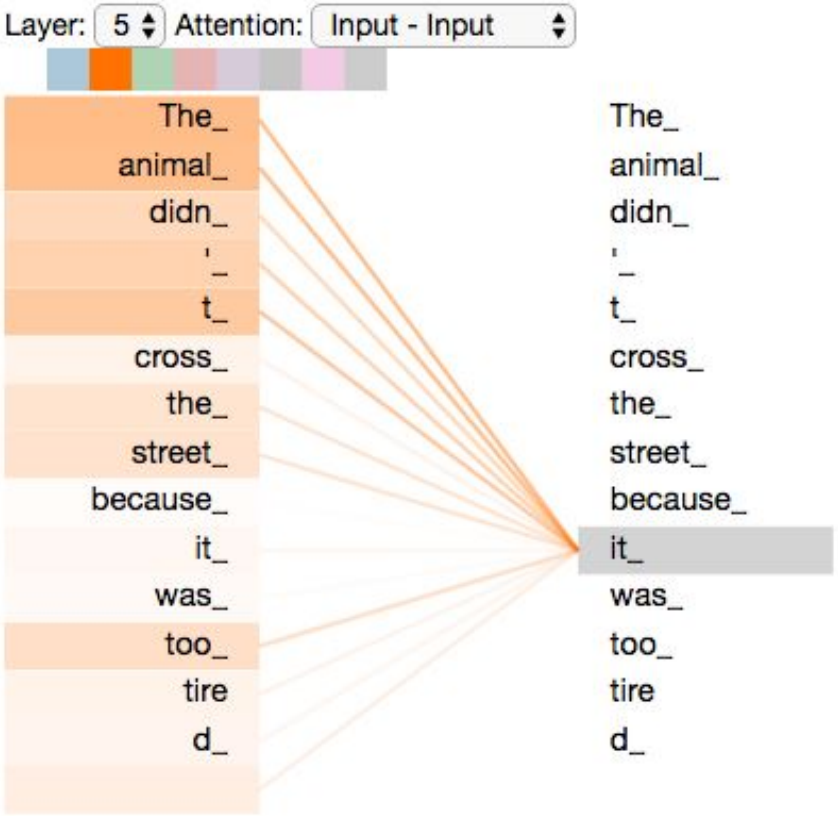
Hospitals are places where people get treated

Representation (in an RNN) **with attention**

Hospitals are places where people get treated

The idea is that learnable weights can identify the important input components and give those a higher weighting when the average of tokens is taken, resulting in a more meaningful embedding of the input.

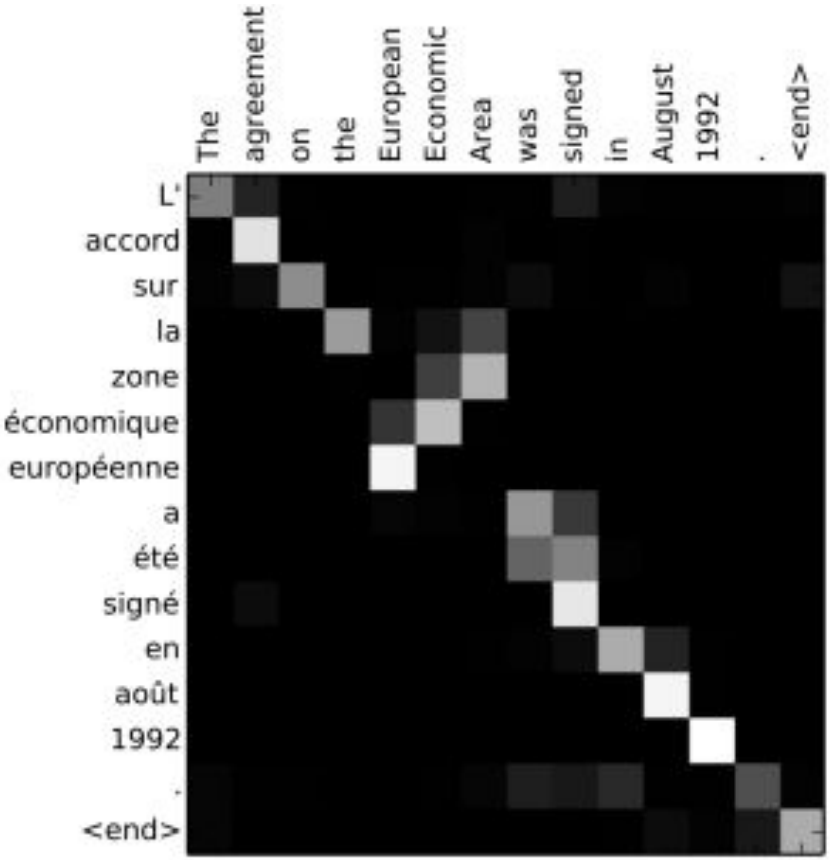
“Attention”


Layer: 5
Attention: Input - Input
The_
animal_
didn_
'_
t_
cross_
the_
street_
because_
it_
was_
too_
tire
d_
The_
animal_
didn_
'_
t_
cross_
the_
street_
because_
it_
was_
too_
tire
d_

“Attention” Visualisation

- Visualising learned attention weights
- Machine Translation model (ENG-FRE)
- Word order mostly similar (strong diagonal)
- Adjective + Noun ordering different

Visualising attention weights shows that when translating “**the European Economic Area**”, the model is aware that it needs to incorporate stronger weights in the reverse order, so the input (“Area” from English), which occurs outside of a linear ordering, is added into an earlier representation of (“zone” in French).



Transformer Timeline

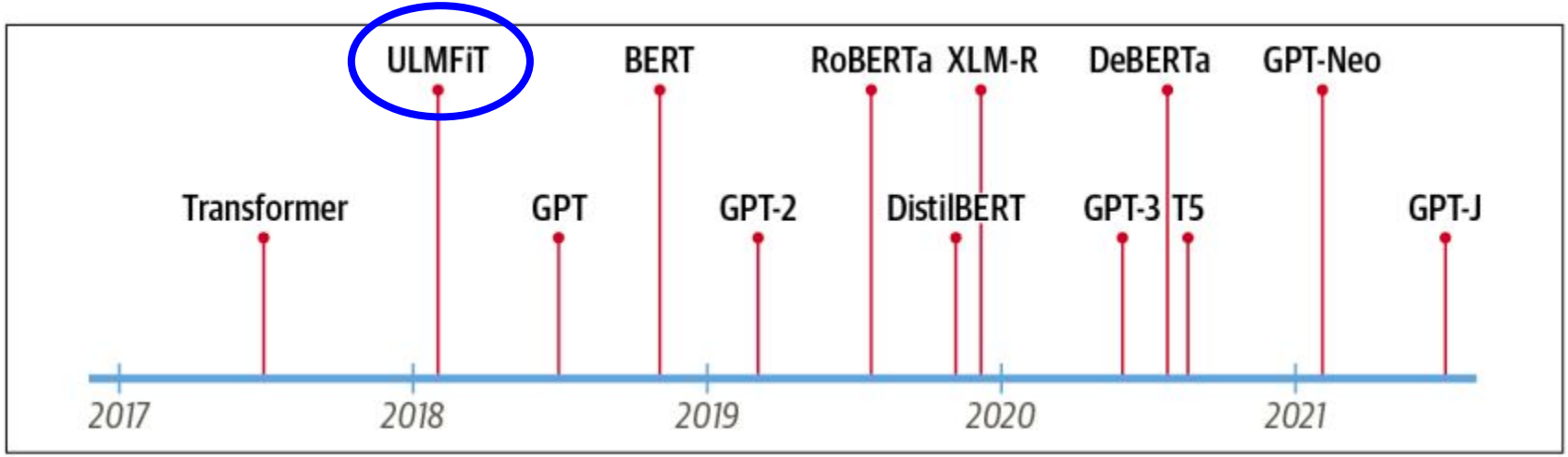
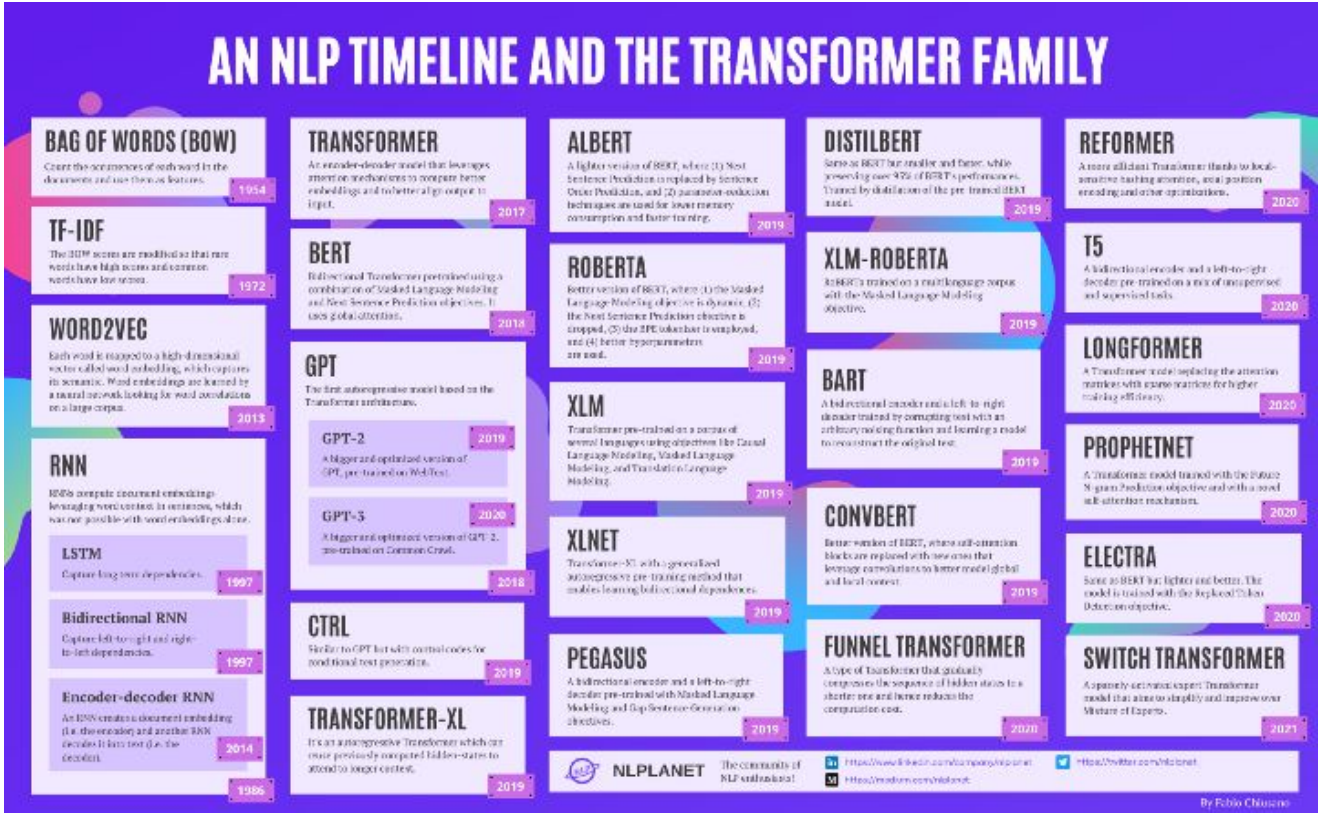


Figure 1-1. The transformers timeline

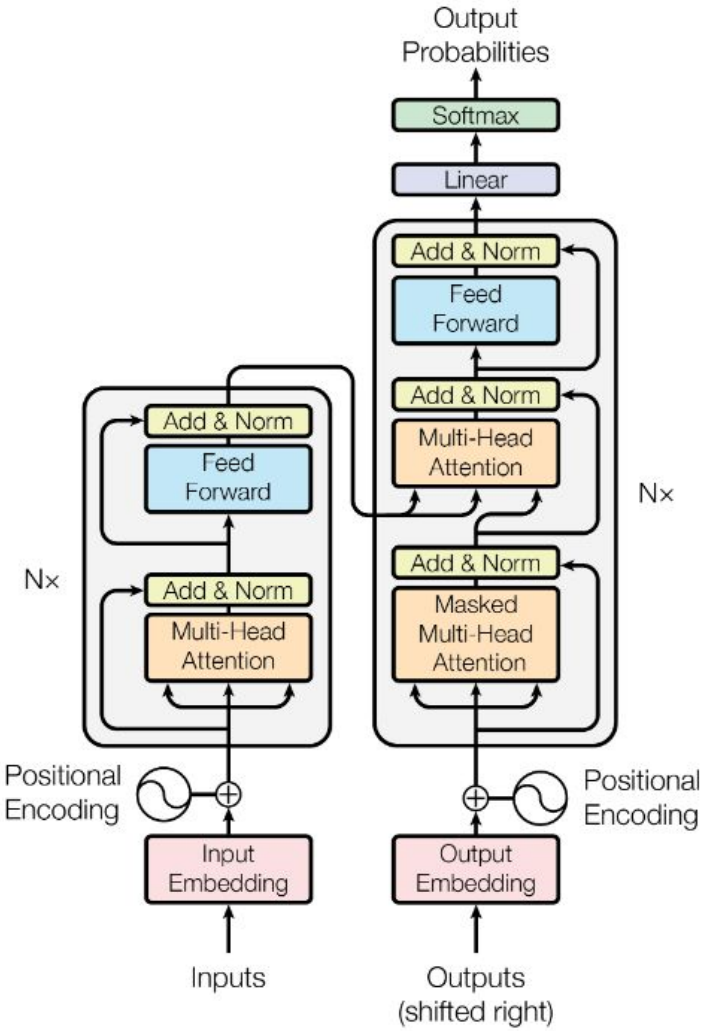
Transformer Timeline



https://medium.com/nlplanet/a-brief-timeline-of-nlp-from-bag-of-words-to-the-transformer-family-7caad8bbba56

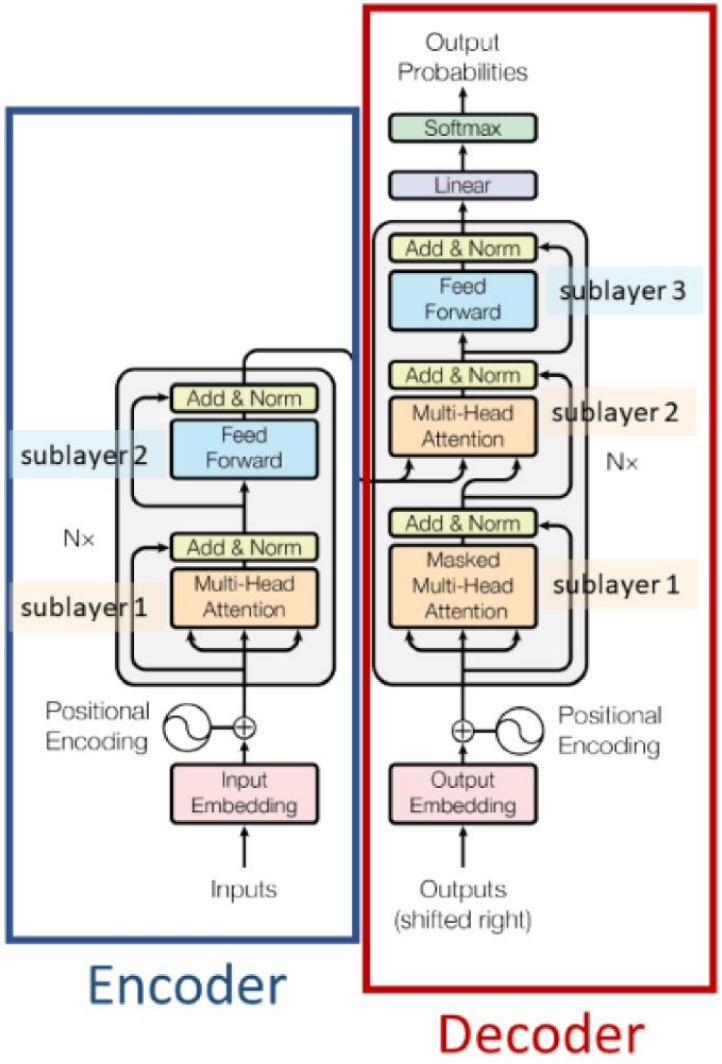
Attention is all you Need (2017)

- `<< Rewind one year >>`
- Removed the need for “**Recurrence**” in RNNs
- This dramatically improved training efficiency
- Positional embeddings represent input order
- Introduced “**Transformer**” architecture



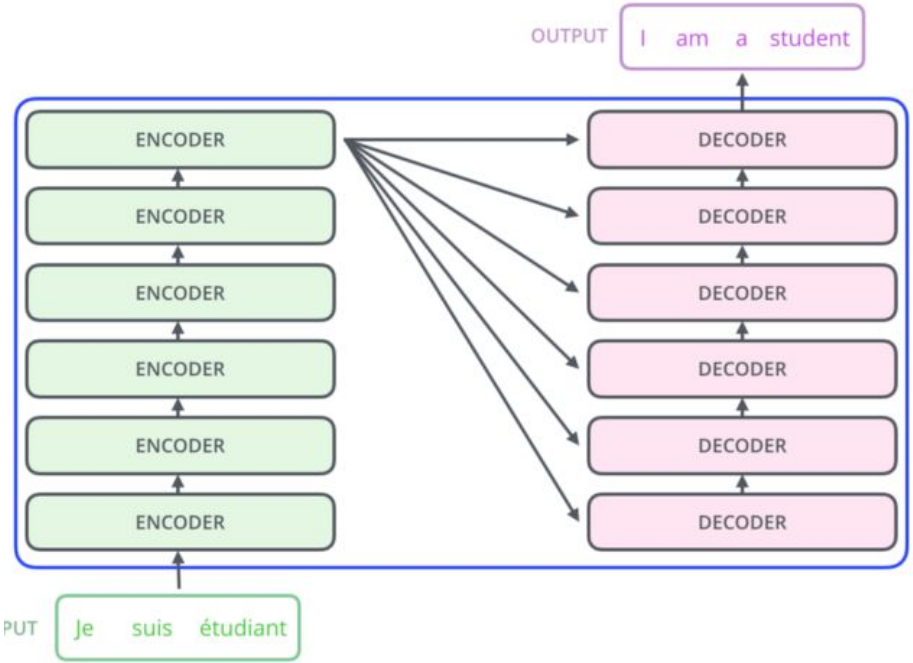
Attention is all you Need (2017)

- << Rewind one year >>
- Removed the need for “**Recurrence**” in RNNs
- This dramatically improved training efficiency
- Positional embeddings represent input order
- Introduced “**Transformer**” architecture



Attention is all you Need (2017)

- << Rewind one year >>
- Removed the need for “**Recurrence**” in RNNs
- This dramatically improved training efficiency
- Positional embeddings represent input order
- Introduced “**Transformer**” architecture



The Transformer Family of Models

- Seq2seq (encoder-decoder)
- Encoder
  - Masked Language Modelling
  - Great feature extractors
  - Useful for non-sequential models
- Decoder
  - Causal Models
  - Great at generating sequences

- Different LMs are based on different components of the Transformer architecture

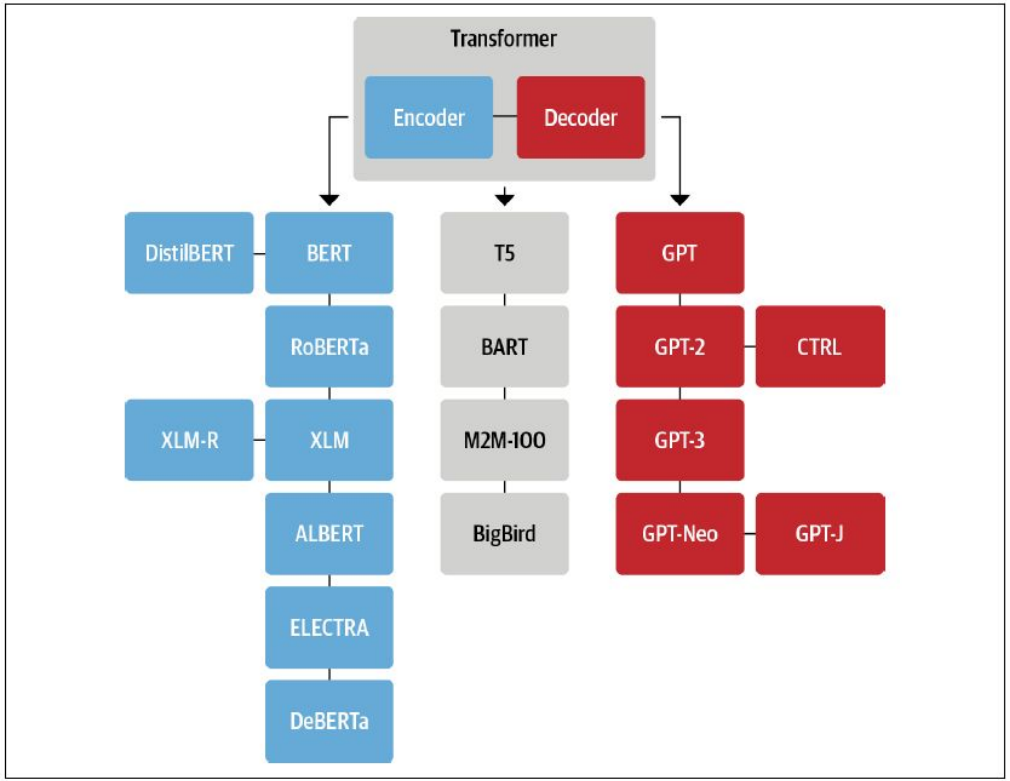
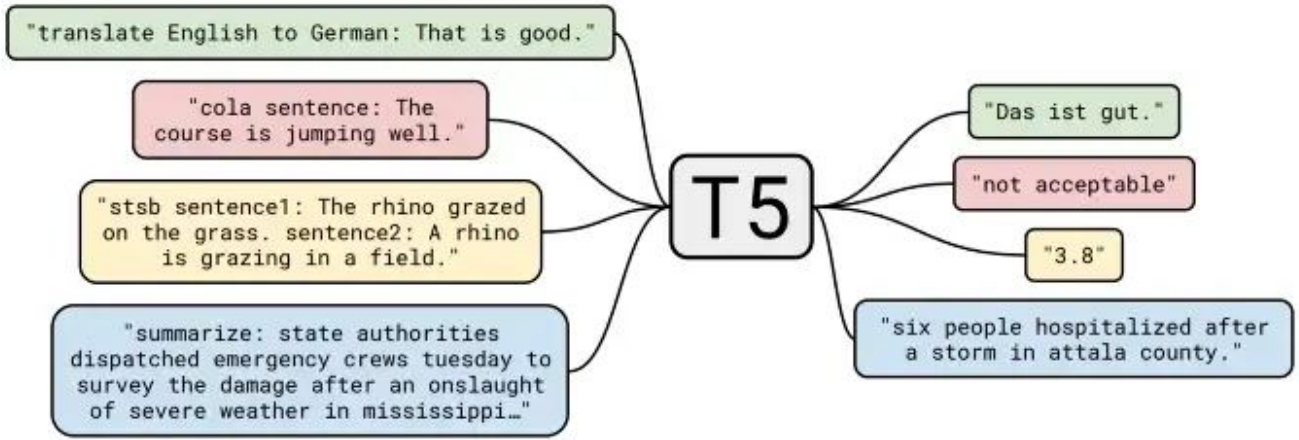


Figure 3-8. An overview of some of the most prominent transformer architectures

Encoder-Decoder Branch

- The smallest branch of the Transformer family tree
- Keeps both encoder and decoder
- Input and output are both sequences



Encoder Branch

- Transformers are originally sequence models
- They had great feature extraction and encoding representations of inputs
- What if you wanted outputs based on the entire input?
- i.e. a single answer and not a sequence?

Many problems require such a setup.

Transformer Encoders were then used for their feature extraction properties and to embed complex inputs into numerical vector representations, meaning Transformers spread around to virtually all NLP tasks quite quickly.

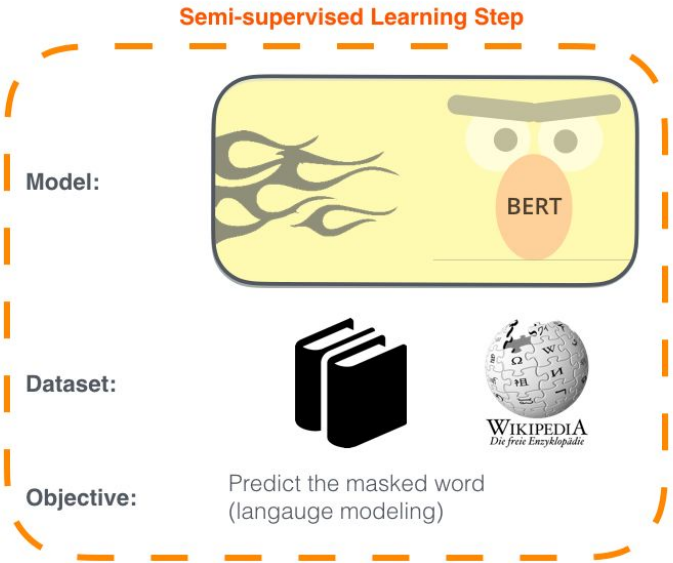
Encoder Branch - BERT (2018)

- Bidirectional Encoder Representations for Transformers
- Remember what ELMo did to ordinary word vectors?
  - reframed the language model objective over the more expressive seq2seq architecture
- BERT did a similar thing
  - reframed the language modelling over the more expressive Transformer architecture
  -
- Specifically, it used the **encoder part** of the Transformer to learn contextual embeddings
- Then all the records and scoreboards for NLP tasks were shattered in an instant
- Big media *attention* (intended)
- **Masked Language Modelling** + Next Sentence Prediction
- BERT powers almost all of Google's search functionality nowadays

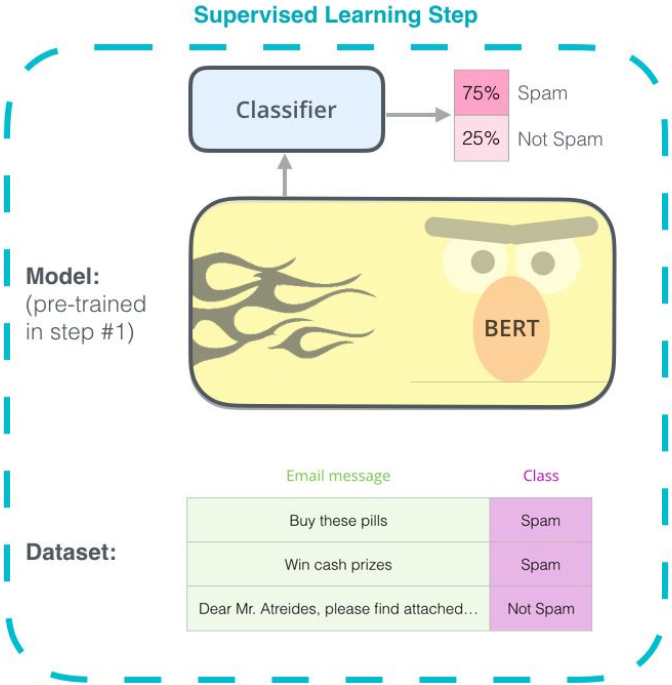
Encoder Branch - BERT (2018)

1 - Semi-supervised training on large amounts of text (books, wikipedia..etc).

The model is trained on a certain task that enables it to grasp patterns in language. By the end of the training process, BERT has language-processing abilities capable of empowering many models we later need to build and train in a supervised way.

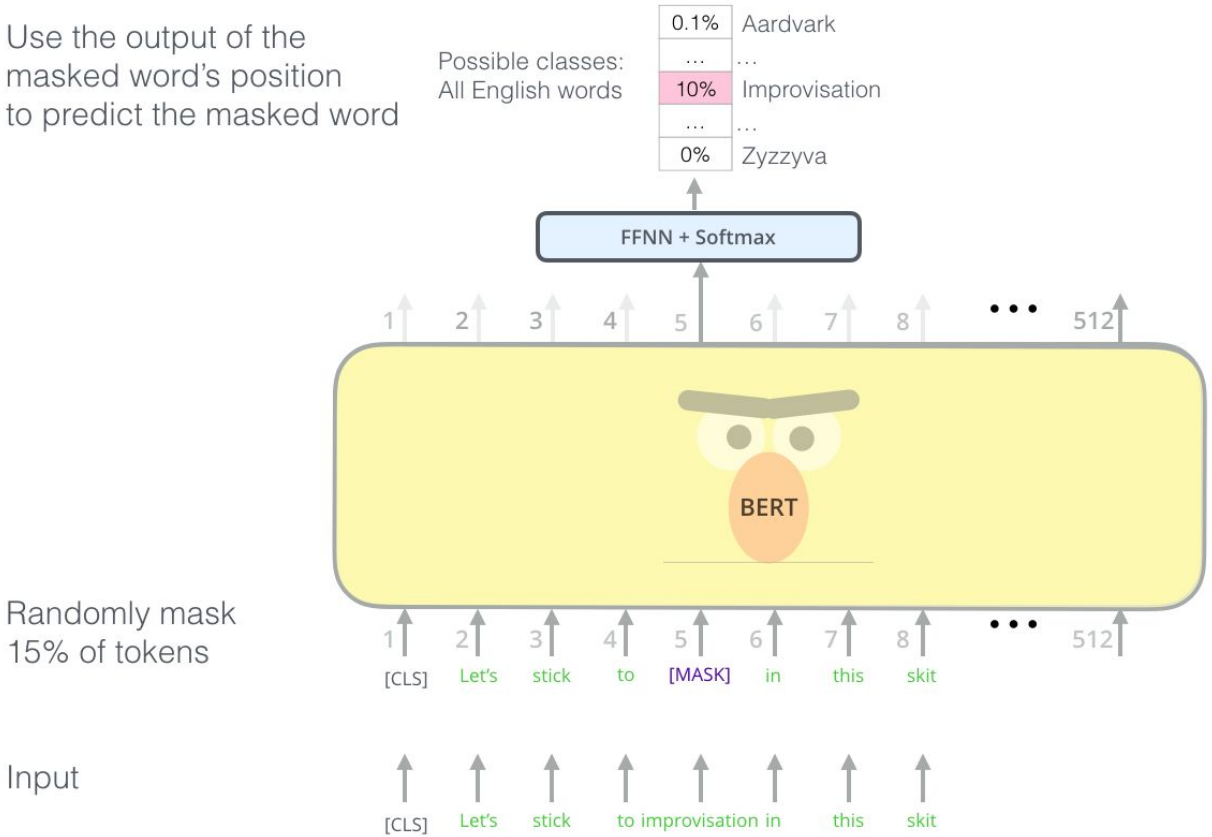


2 - Supervised training on a specific task with a labeled dataset.



https://jalammar.github.io/illustrated-bert/

Masked-Language Modelling



Decoder Branch

- Generate sequences
- Less concerned with converting inputs into condensed feature vector
- They can continue sequences based on a **prompt**
- Lots of interest in discovering what LMs “know” via prompting
- “**Prompt engineering**” now a term, used for decoder Transformers (and in image-text prompting like DALLE-2)
- Lots of applications in human-computer interaction, chatbots etc.
- Whole startups are rising up solely powered by GPT3 (figurehead of the Transformer decoder branch)

Natural Language Generation with GPT

- Particularly good at generating natural language
- **2018:** GPT-1 (117 million)
- **2019:** GPT-2 (1.5 billion)
- **2020:** GPT-3 (175 billion)


ARTIFICIAL INTELLIGENCE | NEWS

GPT-4 Will Have 100 Trillion Parameters — 500x the Size of GPT-3

Are there any limits to large neural networks?

GPT3 & Unicorns

Prompt:

In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

Response:

"We were shocked to discover the unicorns," said anthropologist Daniel St. Maurice. "They were like nothing we had ever seen before. We had heard legends of the unicorns, but never thought they actually existed."

When the scientists first arrived in the valley, the unicorns were surprised and startled by the presence of humans, but were also excited. The unicorns welcomed the researchers and explained that they had been waiting for them for a very long time. "The unicorns said that they had been waiting for us for a very long time," said Dr. St. Maurice. "They said they had always known that humans would eventually discover them, but that they had also always known that humans would be too stupid to realize the unicorns had been waiting for them."

The unicorns told the scientists that they had left the valley to explore the world, but that after years of traveling the world, they found humans to be incredibly stupid and decided to return to the valley to live in peace. "We were shocked," said Dr. St. Maurice. "We had no idea that the unicorns had been waiting for us. We also had no idea that unicorns were capable of speech. It was a truly magical moment."

GPT3 & Unicorns

Prompt:

In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

Response:

"We were shocked to discover the unicorns," said anthropologist Daniel St. Maurice. "They were like nothing we had ever seen before. We had heard legends of the unicorns, but never thought they actually existed."

When the scientists first arrived in the valley, the unicorns were surprised and startled by the presence of humans, but were also excited. The unicorns welcomed the researchers and explained that they had been waiting for them for a very long time. "The unicorns said that they had been waiting for us for a very long time," said Dr. St. Maurice. "They said they had always known that humans would eventually discover them, but that they had also always known that humans would be too stupid to realize the unicorns had been waiting for them."

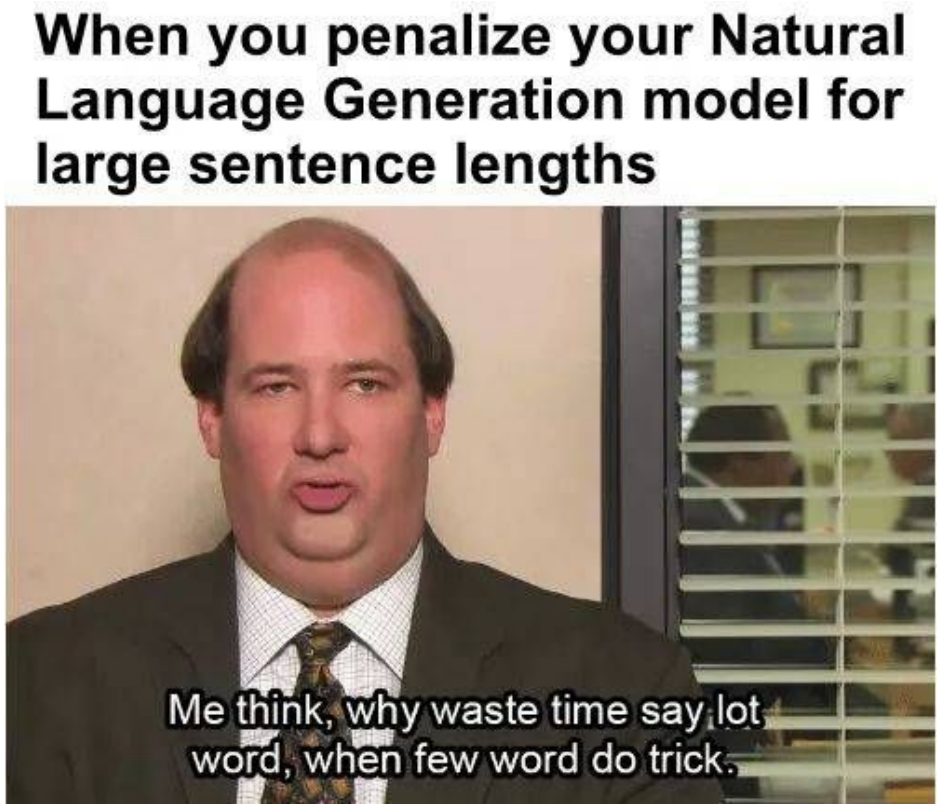
The unicorns told the scientists that they had left the valley to explore the world, but that after years of traveling the world, they found humans to be incredibly stupid and decided to return to the valley to live in peace. "We were shocked," said Dr. St. Maurice. "We had no idea that the unicorns had been waiting for us. We also had no idea that unicorns were capable of speech. It was a truly magical moment."

How do you generate text from Transformer-decoders?

How do you generate text from Transformer-decoders?

- Decoders are **causal**
- Iteratively select next word
- Each step involves a choice
- Each step involves a **probability distribution** over possible next words

The **sampling strategy** you use has a huge impact on the quality of generated text.



r/MLmemes

Sampling Strategies

- Greedy
- Beam Search
- Random
- Top-K
- Nucleus (top-p)

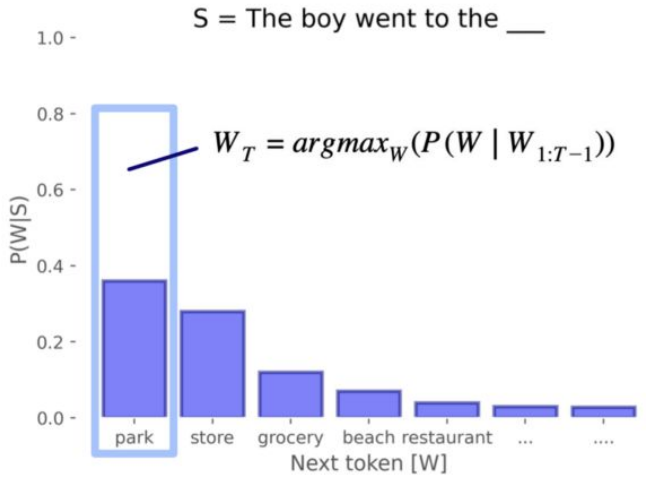
Sampling Strategies

- **Greedy**
- Beam Search
- Random
- Top-K
- Nucleus (top-p)

Greedy sampling is very simple.

You have a probability over next word tokens, so pick the token with the highest probability.

It's not very clever and does not produce flowing varied language.



Sampling Strategies

- **Greedy**
- Beam Search
- Random
- Top-K
- Nucleus (top-p)

Greedy sampling is very simple.

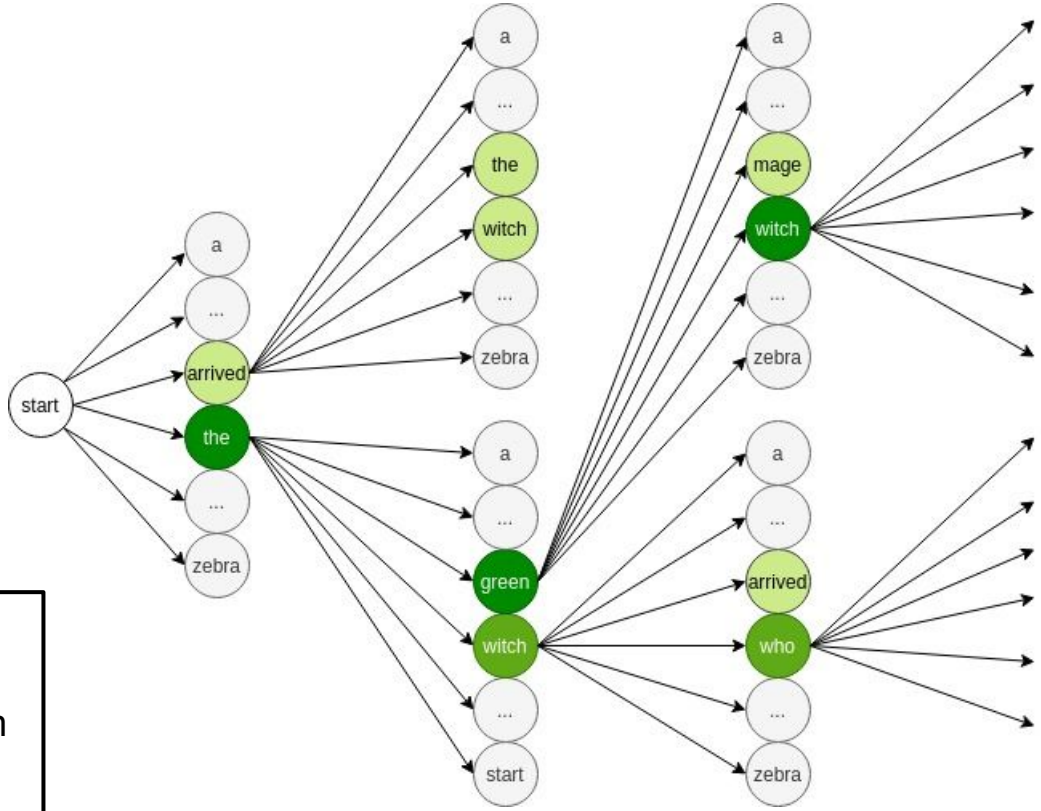
You have a probability over next word tokens, so pick the token with the highest probability.

It's not very clever and does not produce flowing varied language.

Question: when is greedy sampling the only sensible choice?

Sampling Strategies

- Greedy
- **Beam Search**
- Random
- Top-K
- Nucleus (top-p)



At each step, consider **N** most likely candidates and keep them around. Expand along those paths until maximum beam width reached. At each split, only keep the top **K** hypothesis.

Sampling Strategies

- Greedy
- Beam Search
- **Random**
- Top-K
- Nucleus (top-p)

At any current timestep, you have a probability distribution over the vocabulary for your choice of next word to output from the model.

Since you already have a probability distribution, you can just sample from it, therefore selecting words in proportion to their probability.

Question: what problems might this introduce?

Hint: think of generating very long sequences.

Sampling Strategies

- Greedy
- Beam Search
- Random
- **Top-K**
- Nucleus (top-p)

Top-*k* sampling involves removing from consideration many low-probability words. With long sequences being generated, even a 1 in 10,000 chance of sampling a very low-probability word is undesirable as this influences all the words selected ***afterwards***.

Therefore, if you rank the words from most probable to least probable, and only consider sampling from the top-k of them, you avoid the potential undesirable consequences of randomly picking a very low-probability (and therefore unsuitable) word during your sequence generation.

Sampling Strategies

- Greedy
- Beam Search
- Random
- Top-K
- **Nucleus (top-p)**

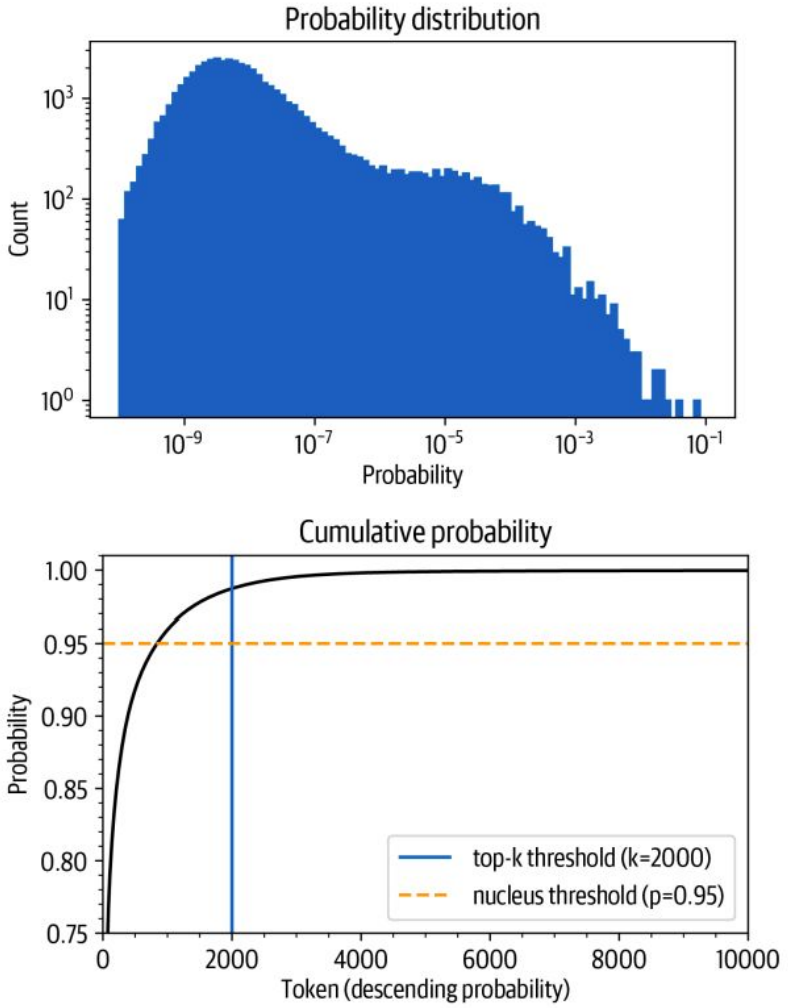
Top-*p* Sampling considers the tokens that make up the top p% of the tokens over the next word's probability distribution.

If p = 0.9 (90%), and only 15 possibilities make up 90% of the next distribution (very few potential next words), then only 15 will be considered.

Sometimes, 5,000 words might make up 90% of the probability distribution, so this avoids the problem with dynamically shifting distributions in top-k sampling.

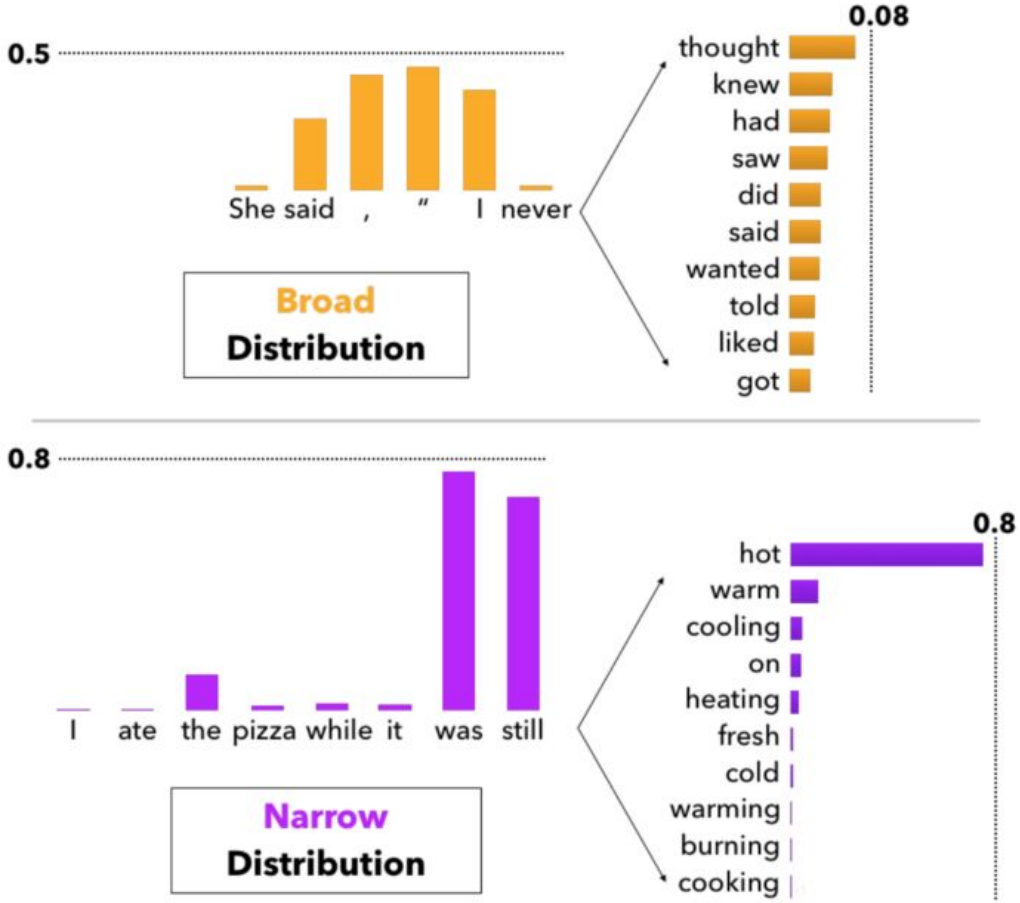
Sampling Strategies

- Greedy
- Beam Search
- Random
- **Top-K**
- **Nucleus (top-p)**



Sampling Strategies

- Greedy
- Beam Search
- Random
- Top-K
- **Nucleus (top-p)**



https://towardsdatascience.com/how-to-sample-from-language-models-682bceb97277

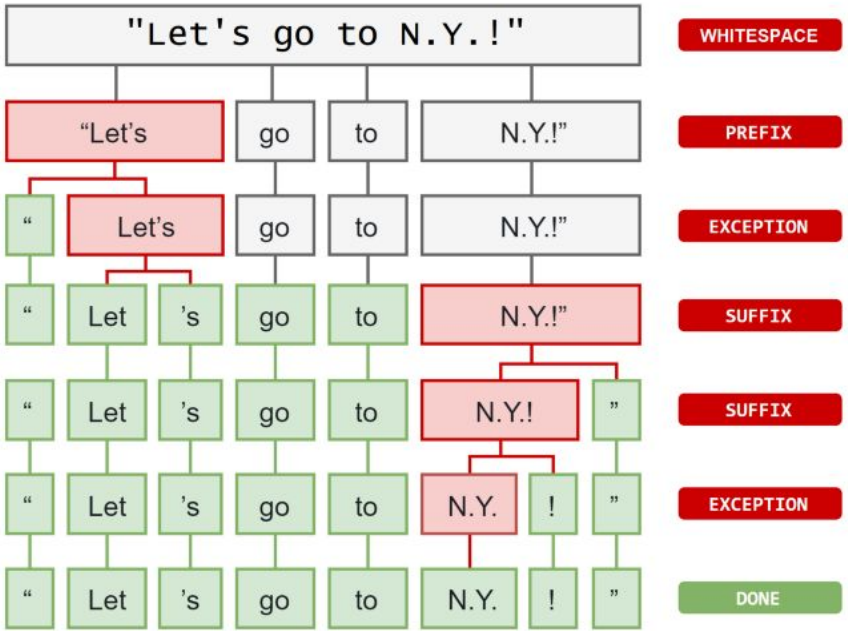
Hallucinations & Other Transformer-Decoder Issues

Examples of Errors in Supervised Model Paraphrase Generation

Error Type	Input Sentence	Model Generated Paraphrase
Stuttering	a man in a chef hat prepping some food	a man is sitting in a a a
	A man is holding up a clear umbrella	A man is holding a in the the
Sentence Fragments	a large brown horse eating a glob of leaves	a horse eating a piece of tall brown
	a skateboarder turning around on a half pipe	a person is riding a a skateboard on a
Hallucination	Two tables next to each other along with laptops	two people sitting on the beach with their laptops
	a city street line with very tall buildings	a city street with several signs on the street

Tokenisation

- Different models expect their inputs to be split in different ways
- Recall word-level, character-level and subword-level distinction
- Sometimes, the subword way to split up text can be quite complicated
- If you don't tokenise inputs to a pretrained model in the way it expects, it likely will not recognise a lot of the input, kind of like it's in a different language



Language Models of the Future

- LMs + external memory systems
- LMs + databases (knowledge bases)
  - Don’t rely the model to compress all info in its weights, but consult info sources
  - External changing facts of the world can be updated + re-embedded in knowledge base
  - LMs learn to access these facts and don’t need to be retrained to “track” world events
- Virtual Assistants
  - Removing the need for translation
  - Opening up communication barriers across the world
  - Instant access to vast amounts of material available in foreign languages

Wrapping Up

- Modern Language Models are tough to understand in detail
- Hopefully you've got a sense of the scale and applicability of what LMs do
- It's fine for a lot to still remain unclear
- You will see more of this if you continue in ML

HuggingFace

- Transformer Language Models used to be very inaccessible
- Complicated code
- Required full understanding of the model and lots of compute power
- HuggingFace devised a library and ecosystem to bring LMs to the masses
- Abstracts away required technical knowledge down to a few lines of code
- Makes it very, very easy to get started and play around with pretrained models

```
>>> pipe = pipeline("text-classification")
>>> pipe("This restaurant is awesome")
[{'label': 'POSITIVE', 'score': 0.9998743534088135}]
```

If you want to use a specific model from the hub you can ignore the task if the model on

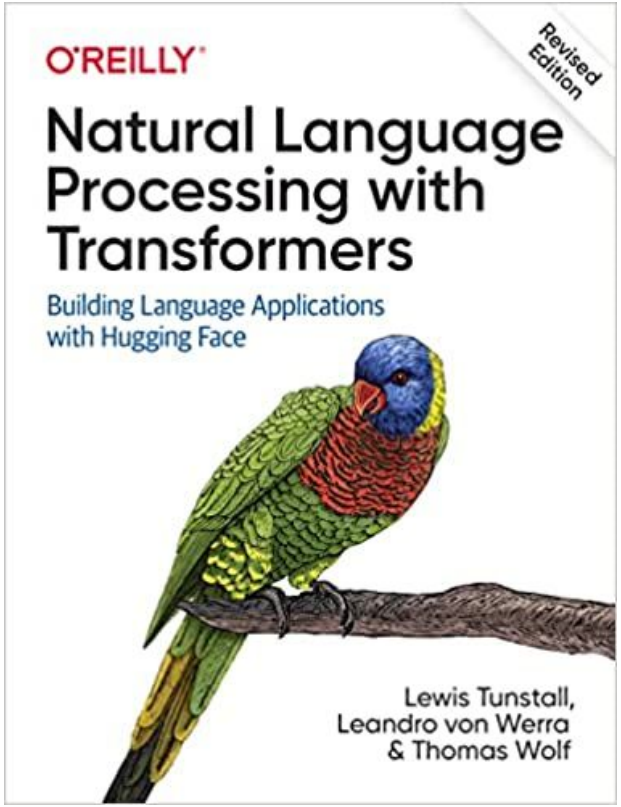
```
>>> pipe = pipeline(model="roberta-large-mnli")
>>> pipe("This restaurant is awesome")
[{'label': 'POSITIVE', 'score': 0.9998743534088135}]
```

To call a pipeline on many items, you can either call with a *list*.

```
>>> pipe = pipeline("text-classification")
>>> pipe(["This restaurant is awesome", "This restaurant is aweful"])
[{'label': 'POSITIVE', 'score': 0.9998743534088135},
 {'label': 'NEGATIVE', 'score': 0.9996669292449951}]
```

Further Reading

- https://huggingface.co/course/chapter1/1
- https://jalammar.github.io/



Thank you :)