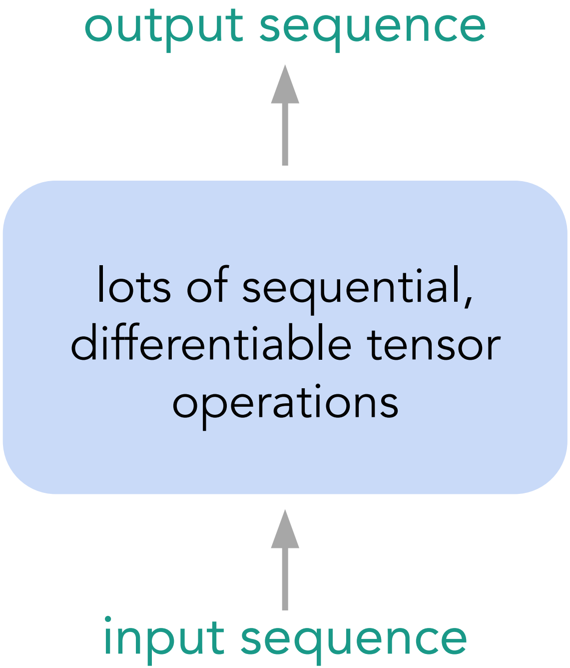
Transformers and Generative AI Essentials

Dipanjan (DJ) Sarkar

Analytics Vidhya

Generative AI for NLP in a nutshell

A versatile set of models that can be used to process and generate sequential data



Generative AI for NLP - Sequence-to-Sequence Models

Examples of sequence-to-sequence models.

output sequence

lots of sequential, differentiable tensor operations

input sequence

	inputs	outputs
Sentiment Analysis	Incredibly creative!	★★★★★
Machine Translation	Incredibly creative!	Incrivelmente criativo!
Dialog systems	How is the weather?	It is sunny today!
Speech recognition		How are you?

Analytics Vidhya

Two big questions?

How can we make make machines understand text?

Analytics Vidhya

1. How to represent words and documents? Embeddings

Embeddings: model learnt latent representations of words

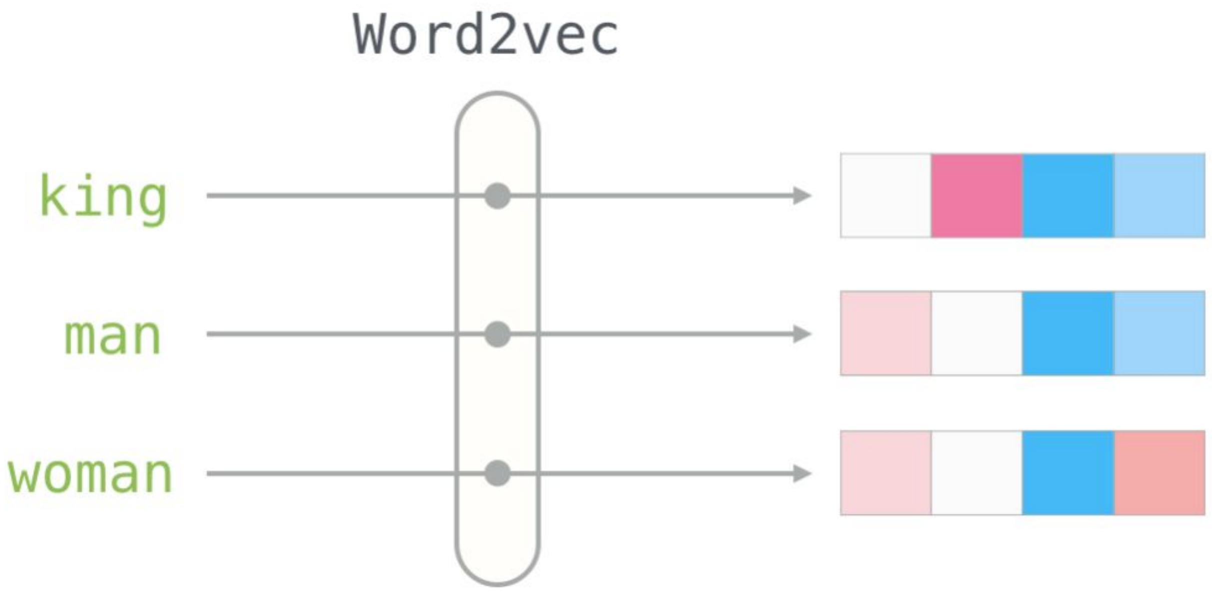
embedding size (<< vocab size)

a	→ .3452 .7162 .1827 .9382 .9182 . . .
ability	→ .1234 .8172 .6473 .5630 .0263 . . .
able	→ .1263 .8054 .5632 .5589 .0374 . . .
about	→ .7364 .2039 .2831 .2837 .1923 . . .
above	→ .9283 .0023 .0065 .2938 .5472 . . .
acarus	→ .1938 .2938 .0293 .5647 .2348 . . .
.	.
.	.
.	.

Analytics Vidhya

Embeddings are learnt from several documents

A Neural Network Language Model like Word2Vec or even Transformers learn vector representations for each word called as embeddings



Source: https://jalammar.github.io/illustrated-word2vec/

Analytics Vidhya

Embeddings encode the meaning of words

Let's say you take the Big Five personality traits test and get some scores as shown here

Openness to experience	79 out of 100
Agreeableness	75 out of 100
Conscientiousness	42 out of 100
Negative emotionality	50 out of 100
Extraversion	58 out of 100

Source: https://jalammar.github.io/illustrated-word2vec/

Embeddings encode the meaning of words

For any person you can use these five attributes or dimensions to represent their personality

Openness to experience	79 out of 100
Agreeableness	75 out of 100
Conscientiousness	42 out of 100
Negative emotionality	50 out of 100
Extraversion	58 out of 100

	Trait #1	Trait #2	Trait #3	Trait #4	Trait #5
Jay	-0.4	0.8	0.5	-0.2	0.3
Person #1	-0.3	0.2	0.3	-0.4	0.9
Person #2	-0.5	-0.4	-0.2	0.7	-0.1

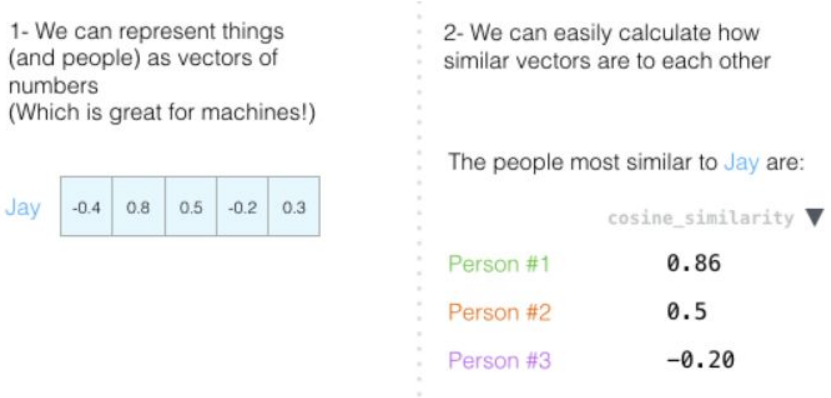
Source: https://jalammar.github.io/illustrated-word2vec/

Analytics Vidhya

Embeddings encode the meaning of words

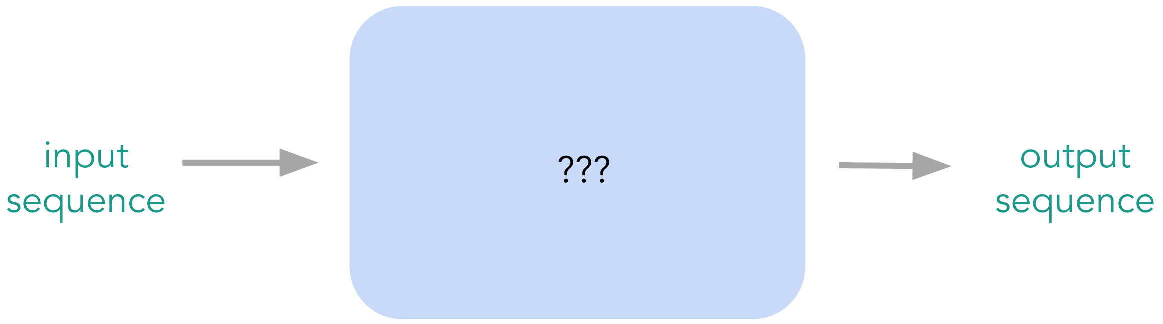
The beauty of embeddings is that you can use them to make machines understand the meaning of words as numeric vectors

1. We can represent people (and things) as vectors of numbers (which is great for machines!).
2. We can easily calculate how similar vectors are to each other.



Source: https://jalammar.github.io/illustrated-word2vec/

What sorts of models are better suited for processing sequential data?



Analytics Vidhya

2. Models? Language Models

- Language Modeling is the task of predicting what word comes next

Analytics Vidhya

Language Models

If one wanted to give an example of an NLP application, one of the best examples would be the next-word prediction feature of a smartphone keyboard. It's a feature that billions of people use hundreds of times every day.

Next-word prediction is a task that can be addressed by a *language model*. A language model can take a list of words (let's say two words), and attempt to predict the word that follows them.

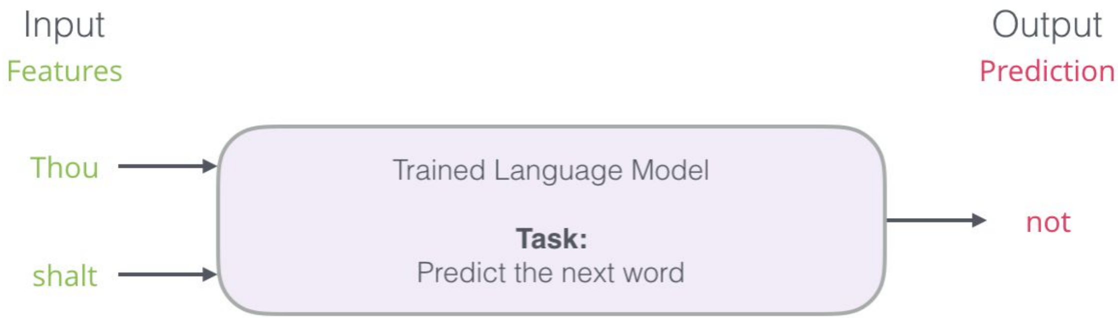
In the screenshot above, we can think of the model as one that took in these two green words (`thou shalt`) and returned a list of suggestions ("not" being the one with the highest probability):

Source: https://jalammar.github.io/illustrated-word2vec/

Analytics Vidhya

Language Models

We can think of the model as looking like this black box:

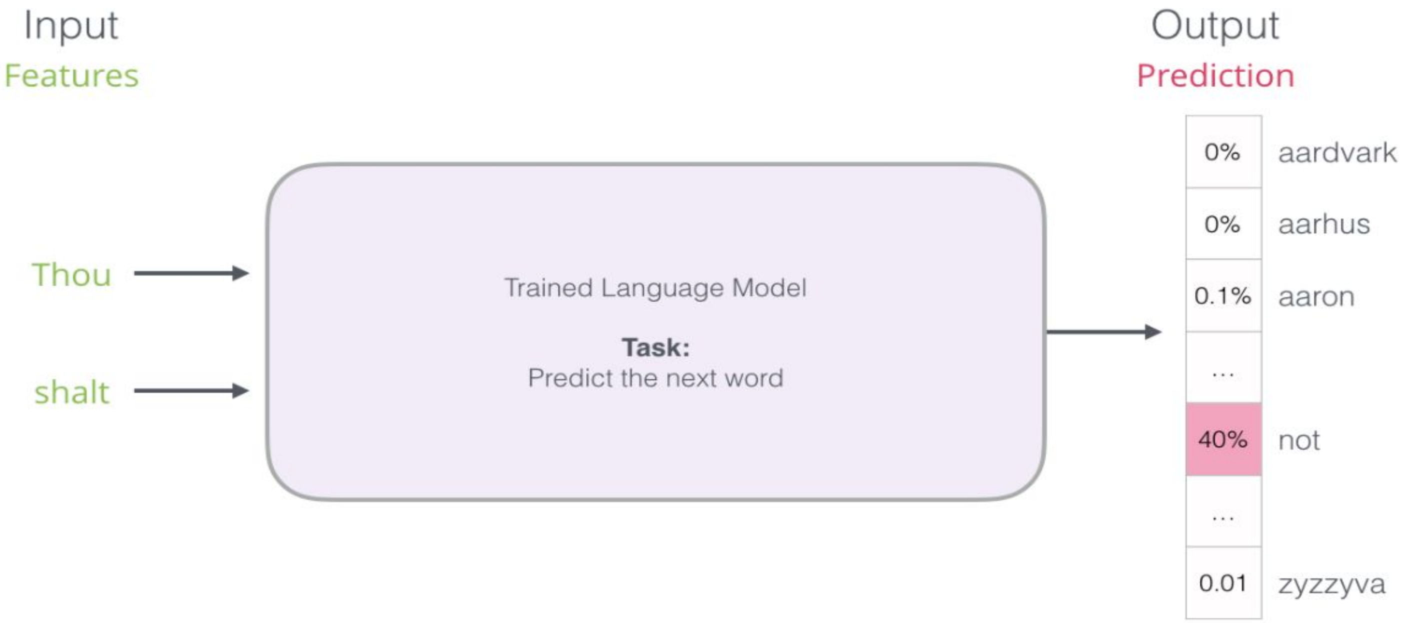


Source: https://jalammar.github.io/illustrated-word2vec/

Analytics Vidhya

Language Models

But in practice, the model doesn't output only one word. It actually outputs a probability score for all the words it knows (the model's "vocabulary", which can range from a few thousand to over a million words). The keyboard application then has to find the words with the highest scores, and present those to the user.



The output of the neural language model is a probability score for all the words the model knows. We're referring to the probability as a percentage here, but 40% would actually be represented as 0.4 in the output vector.

Source: https://jalammar.github.io/illustrated-word2vec/

Analytics Vidhya

Current State of NLP

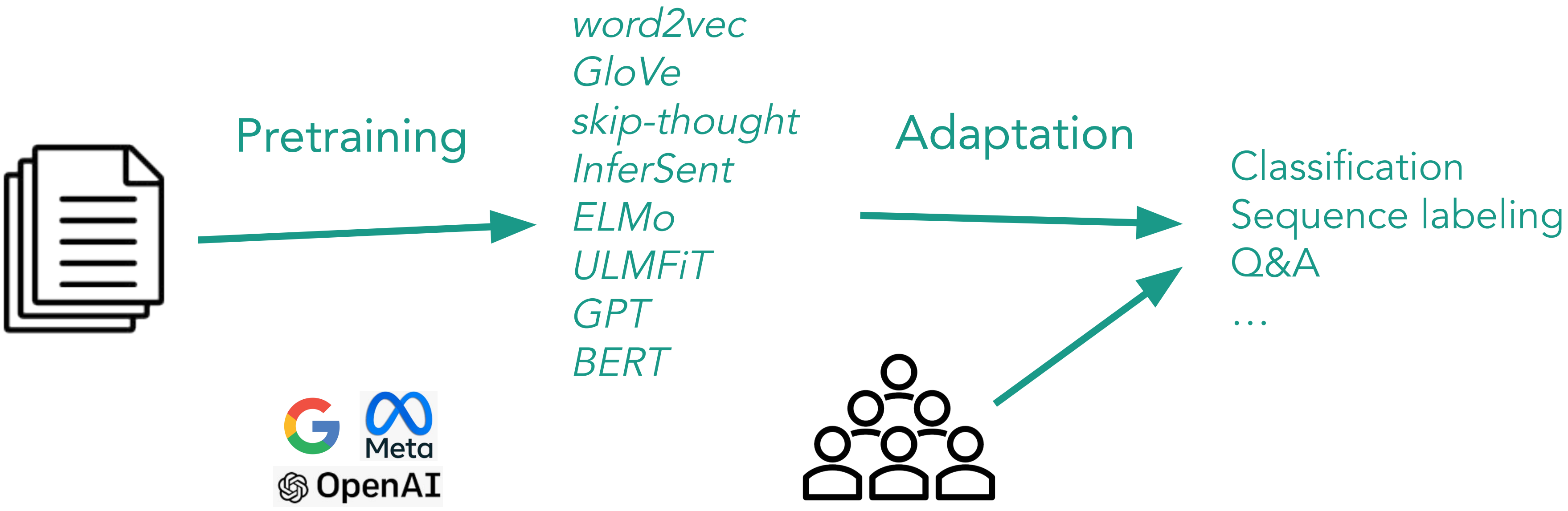
- In recent years, NLP has seen a surge in advancements due to the development of deep learning techniques and the availability of large language datasets.

- These advancements have led to significant progress in several NLP applications thanks to the advent of sequential transfer learning, transformers and LLMs

- Transfer learning involves the use of already pre-trained HUGE models, most notably transformers, and adapting it to solve diverse problems, including machine translation, sentiment analysis, and question-answering systems

Sequential Transfer Learning for NLP

Sequential transfer learning has led to the biggest improvements so far in the field of NLP



What led to LLMs? Transformers

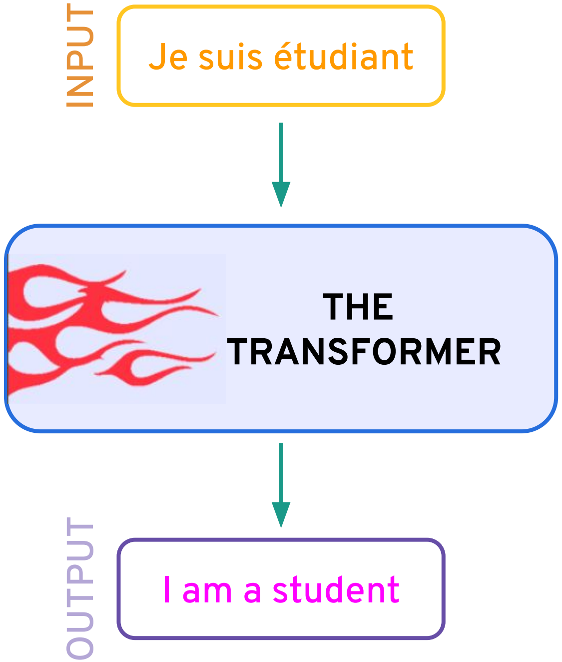
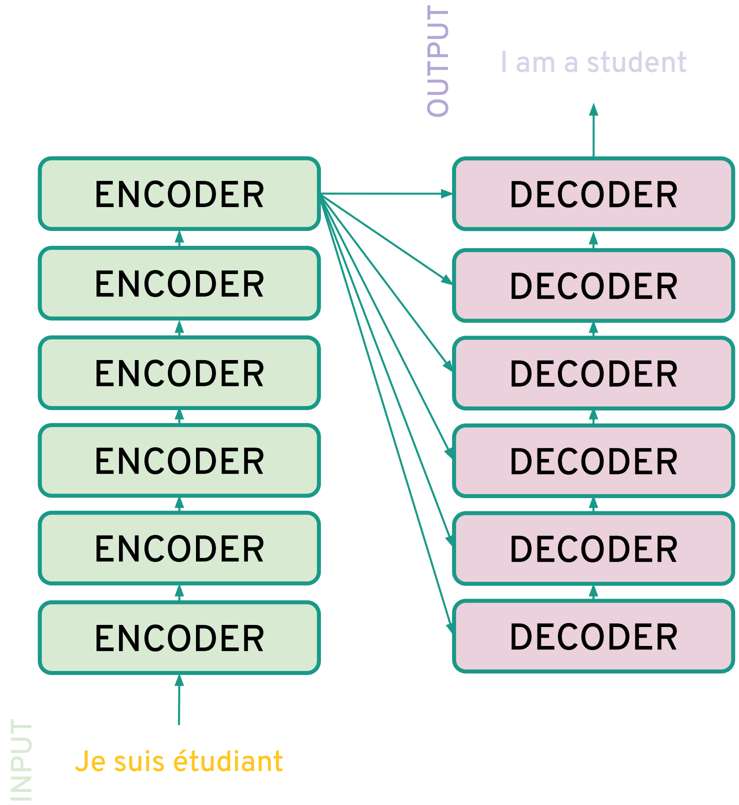


Image Credits: Udacity

- Layered & Stacked Encoder-Decoder Model
- Relies on Multi-headed Self-Attention & Encoder-Decoder Attention
- No sequential RNN-based training (completely parallelized)
- Achieved state-of-the-art performance on several NLP tasks

Analytics Vidhya

Transformer Model Architecture

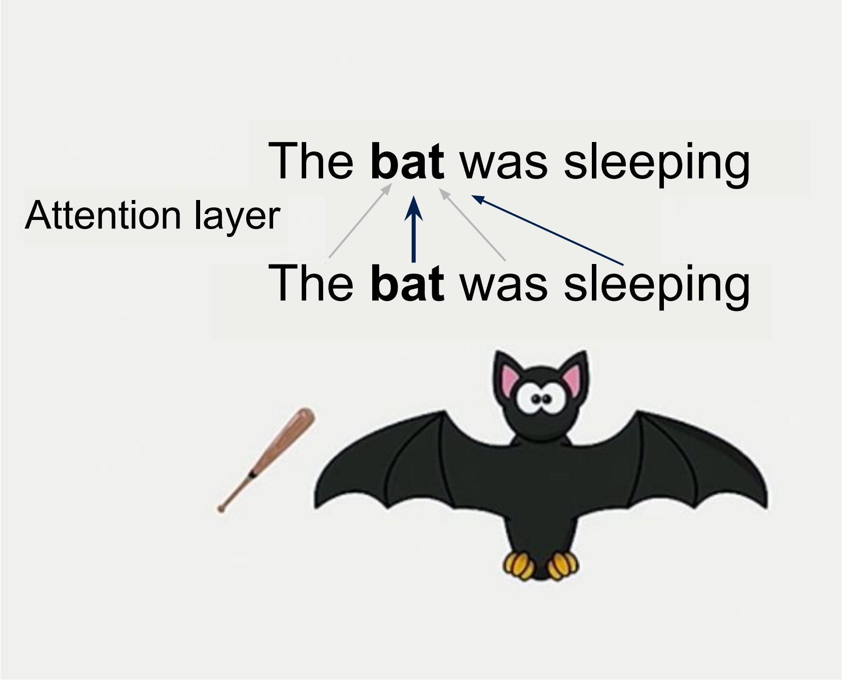


- At heart, a transformer model is a stacked encoder-decoder model
- Has multiple stacked encoder & decoder blocks usually based on standard architectures
- Based on the type of transformer model, only encoder/decoder (or both) are used

Adapted from figure at: https://jalammar.github.io/illustrated-transformer/

Analytics Vidhya

Self Attention: A simple overview



- What does “bat” in this sentence refer to? (A baseball/cricket bat, or the animal?)
- When the model is processing the word “bat”, self-attention allows it to associate “bat” with “sleeping” strongly
  - This gives it the indication that it could be an animal
- Self attention allows the model to look at other positions in the input sequence for clues that can help lead to a better encoding for each word

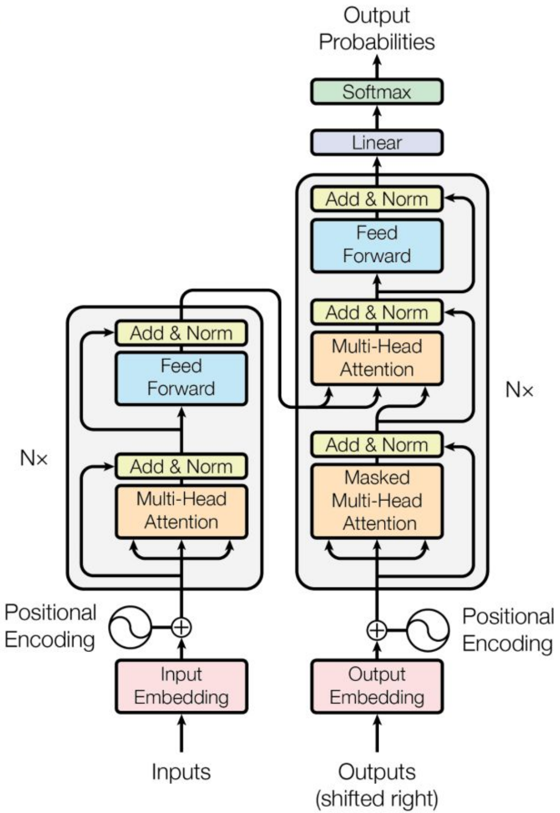
Analytics Vidhya

Transformer Internals - Essentials

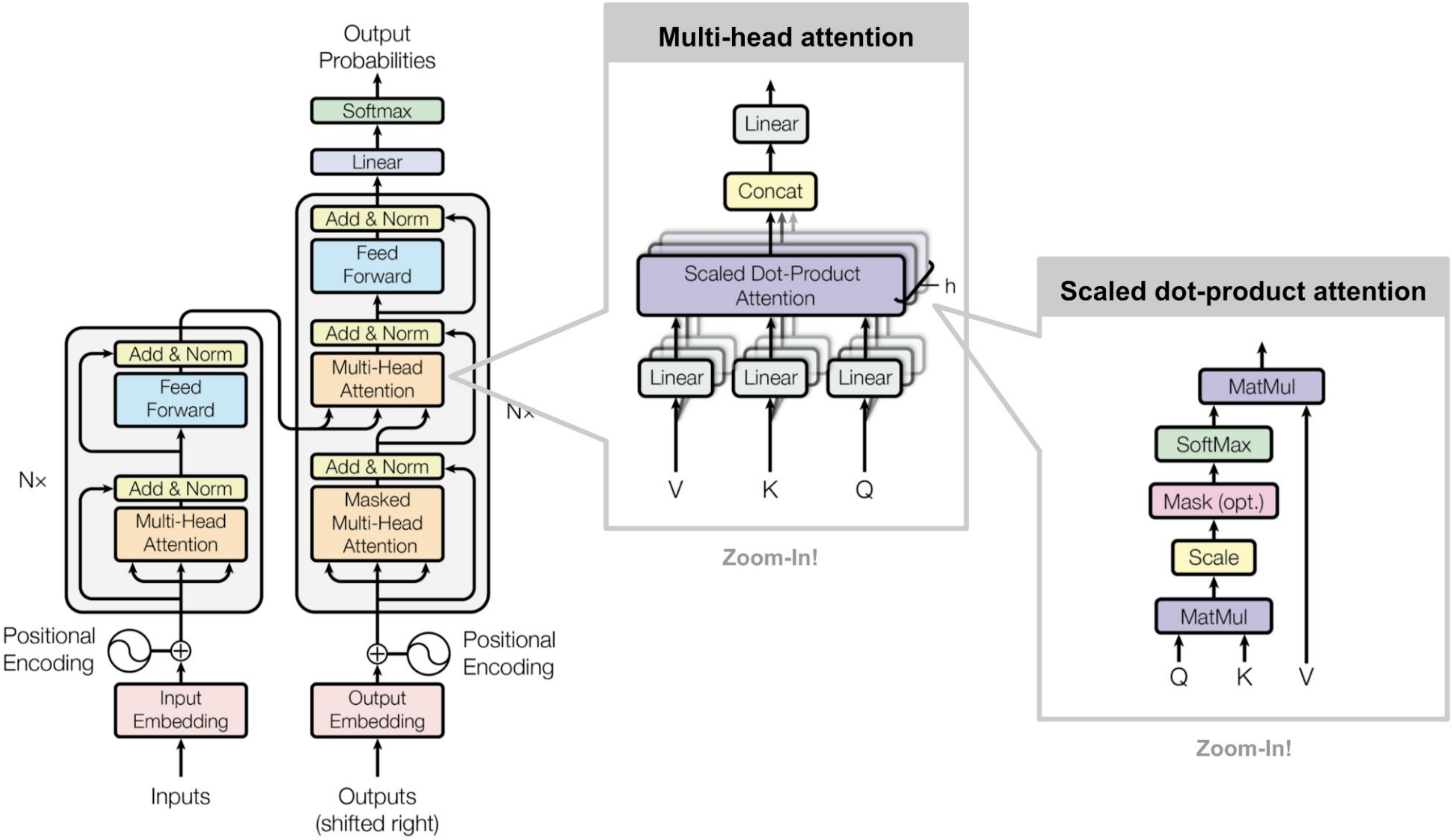
Inputs

Processing Inputs

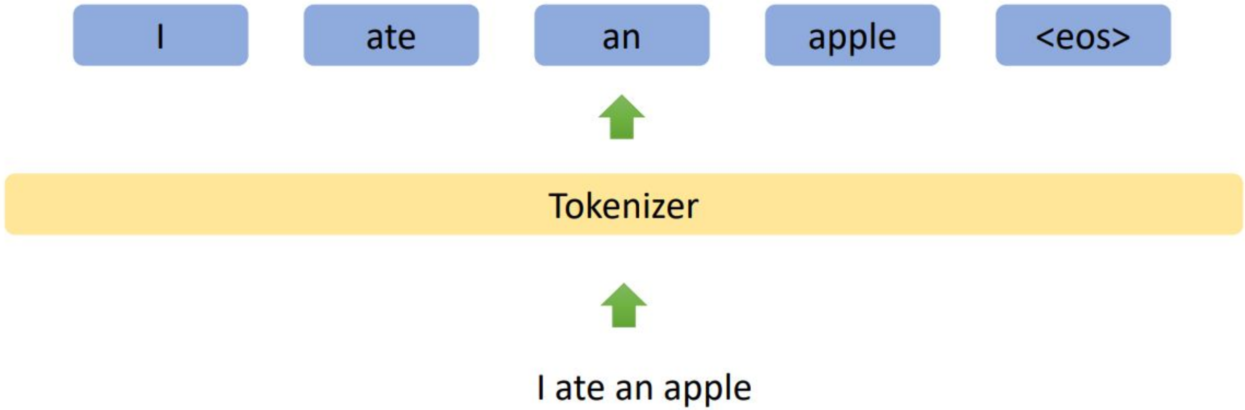
Inputs
I ate an apple

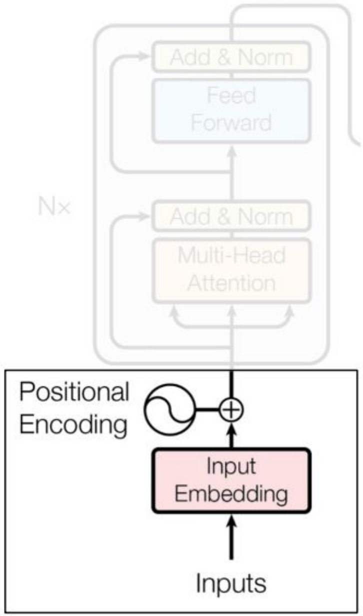


Transformer Internals - Essentials



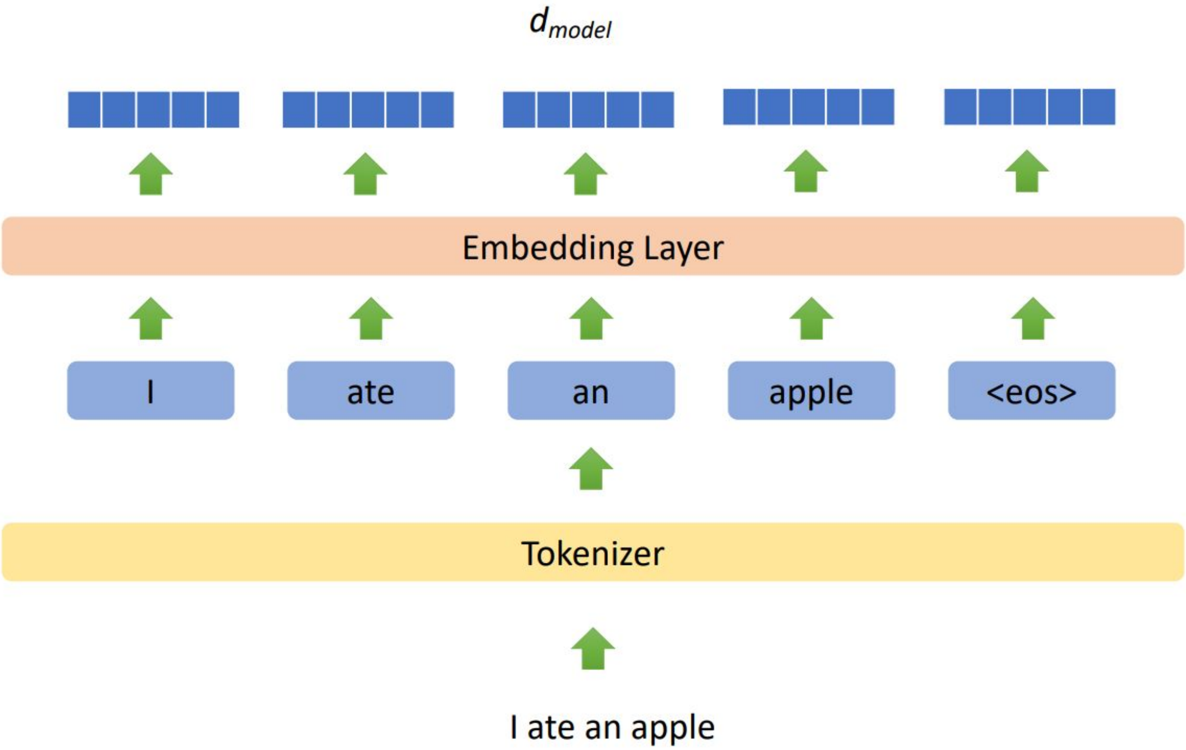
Transformer Internals - Essentials

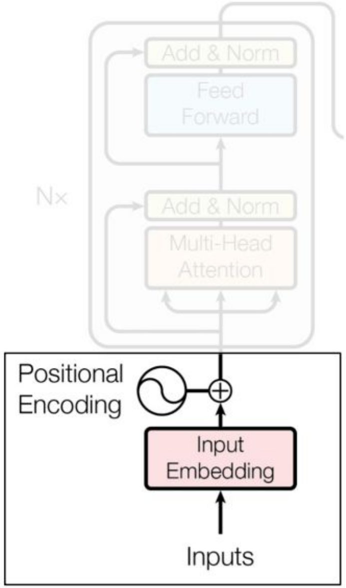




Generate Input Emebeddings

Transformer Internals - Essentials

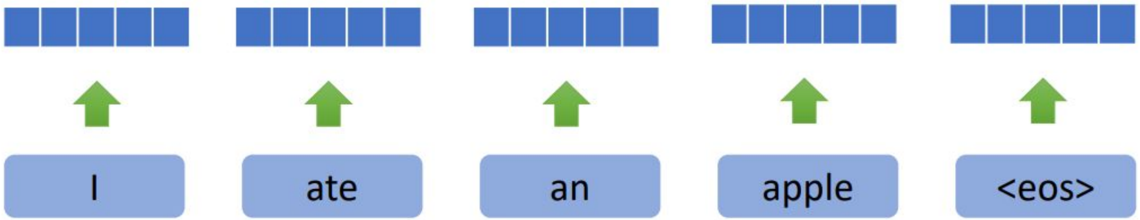


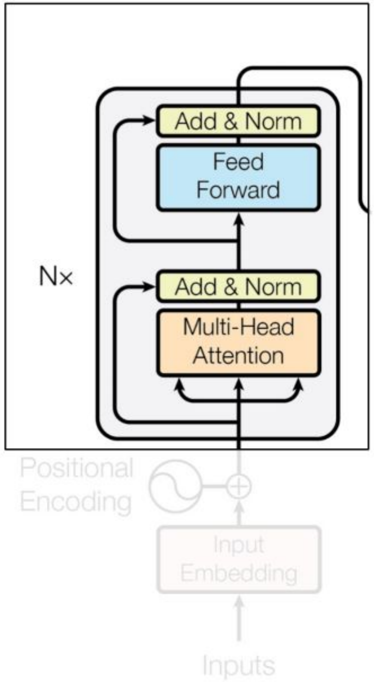


Generate Input Emebeddings

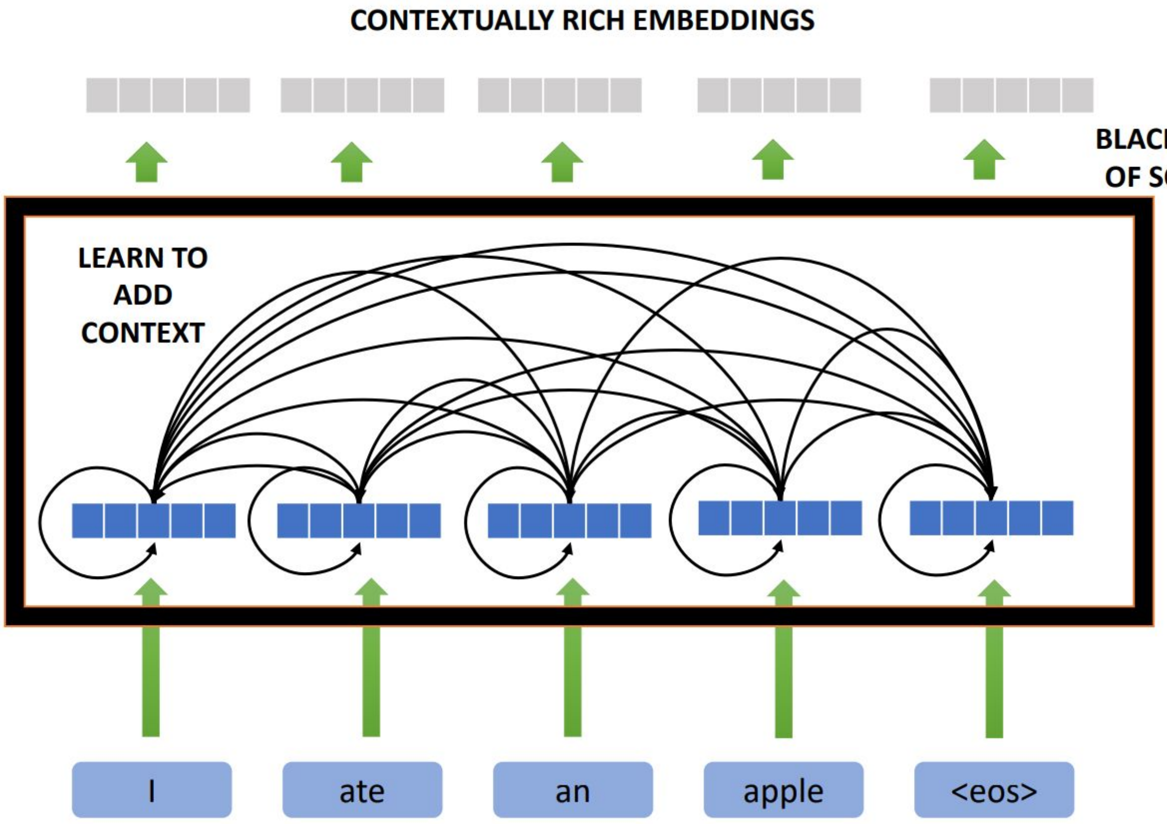
Transformer Internals - Essentials

WHERE IS THE CONTEXT ?

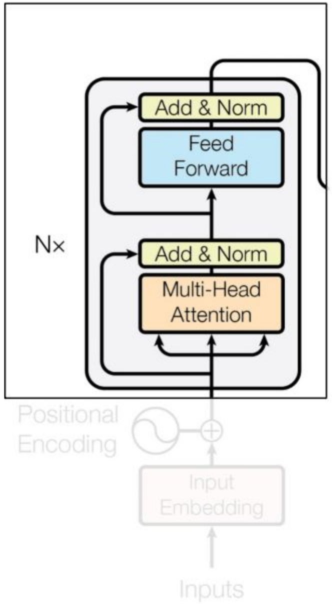




Transformer Internals - Essentials



BLACK BOX OF SORTS

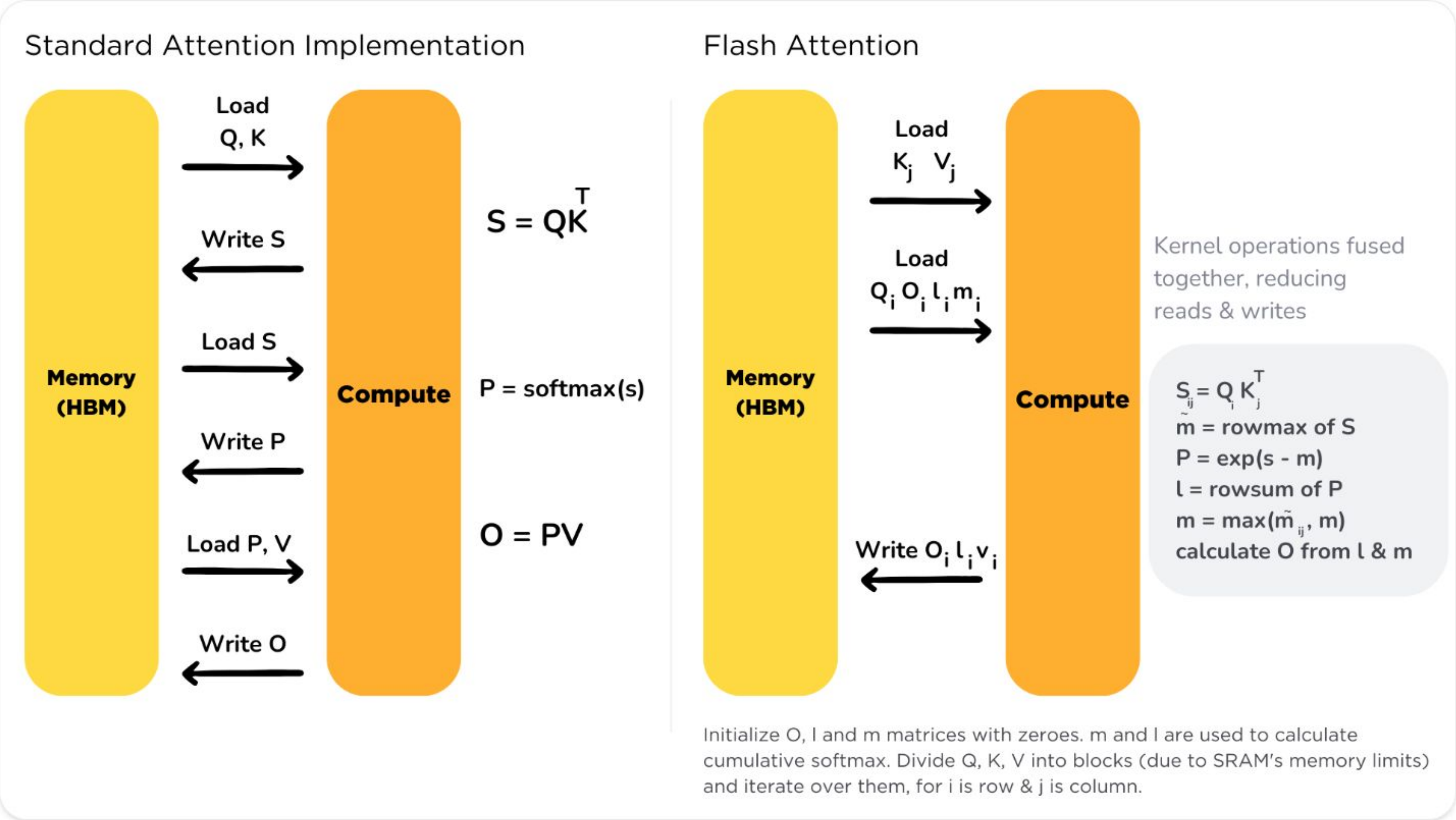


nalytics
dhya

Types of Attention in a Transformer - Brief

- **Self-Attention:**
  - **Understands Relationships:** Helps the model to look at all words in the input to better understand each word.
  - **Improves Context Awareness:** By considering how each word relates to others, it makes the model more context-aware
- **Cross-Attention or Encoder-Decoder Attention:**
  - **Interacts Between Different Sequences:** Used to consider the relationship between elements of two different sequences, like a sentence in English and its translation in French.
  - **Enhances Sequence-to-Sequence Tasks:** Makes the model to "pay attention" to relevant parts of the input sequence when processing the output sequence.
- **Multi-Head Attention:**
  - **Parallel Attention Heads:** Splits the attention mechanism into multiple "heads", allowing the model to simultaneously focus on different parts of the input sequence from different perspectives - like filters of a CNN extracting different features from the same image

Self Attention is Slow. Recent Advancements?



Analytics Vidhya

Transformer Architectures

① Autoregressive Models

- Correspond to the **decoder** of the original transformer model
- Pretrained on the classic language modeling task: guess the next token
- Example - GPT family

② Autoencoding Models

- They correspond to the **encoder** of the original transformer model
- Pretrained by corrupting the input tokens in some way and trying to reconstruct the original sentence (Masked Language Modeling)
- Example - BERT family

③ Sequence to Sequence Models

- Uses **both the encoder and the decoder** of the original transformer
- Most popular applications are translation, summarization, and question answering
- Example - BART, T5

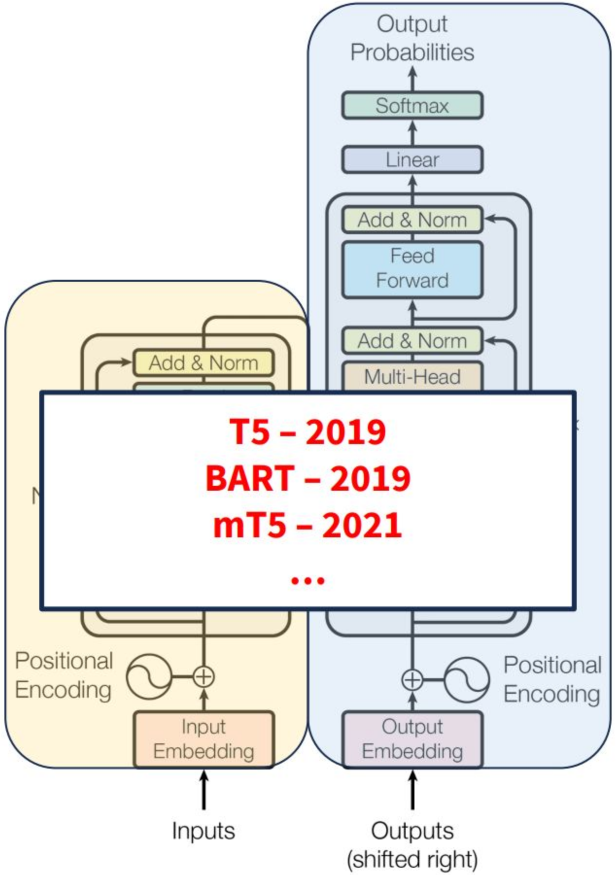
④ Multi-modal Models

- Mixes data of different modalities, e.g. text & images Example - CLIP, DALL-E

Transformer Architectures

BERT – 2018
DistilBERT – 2019
RoBERTa – 2019
ALBERT – 2019
ELECTRA – 2020
DeBERTa – 2020
...

Representation

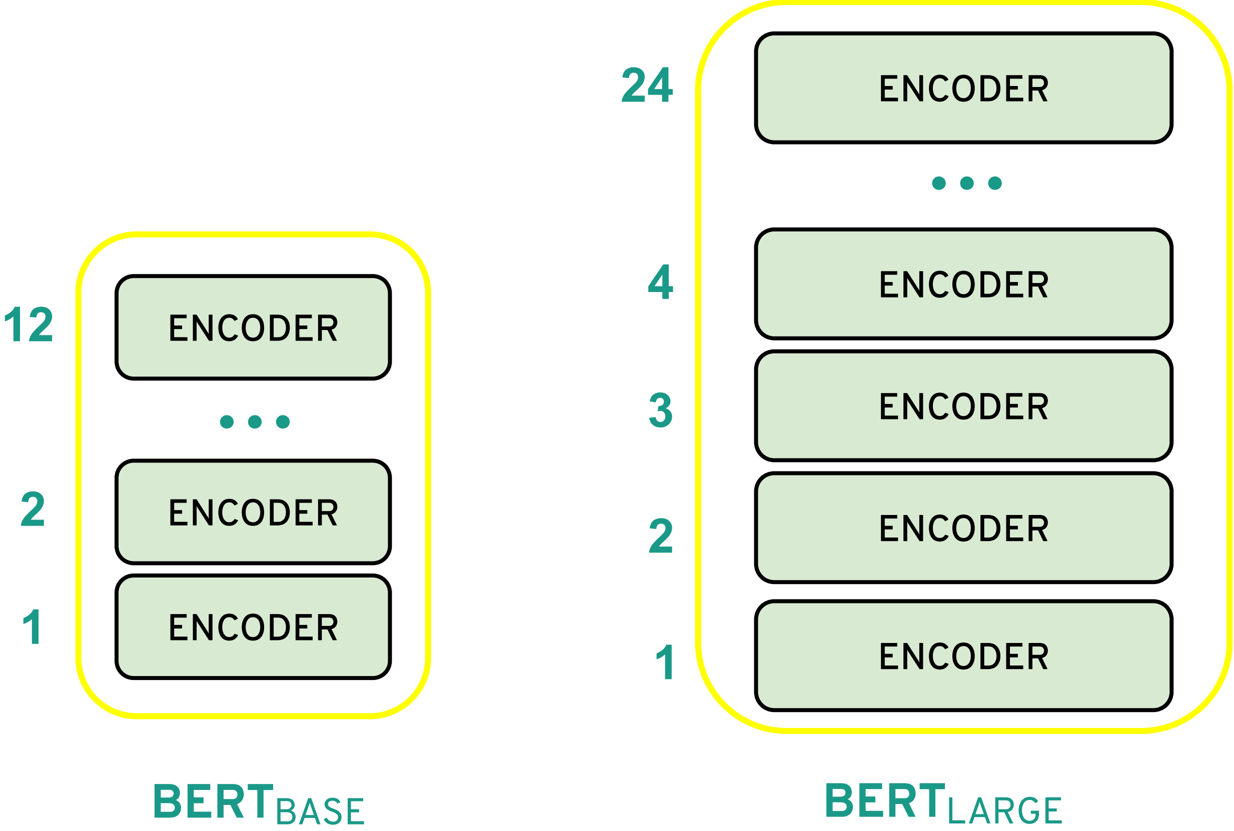


T5 – 2019
BART – 2019
mT5 – 2021
...

GPT – 2018
GPT-2 – 2019
GPT-3 – 2020
GPT-Neo – 2021
GPT-3.5 (ChatGPT) – 2022
LLaMA – 2023
GPT-4 – 2023
...

Generation

Autoencoding (Encoder only) Transformer - BERT



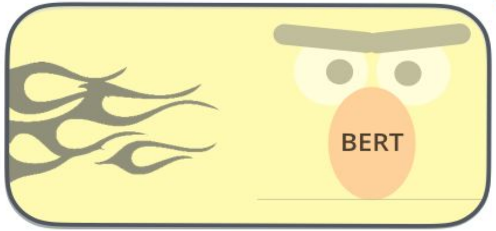
BERT Training Workflow

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.

Semi-supervised Learning Step

Model:



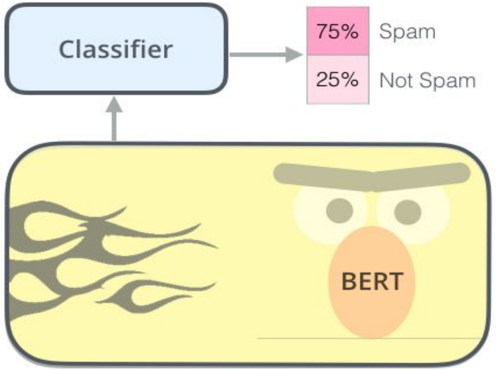
Dataset:

Objective: Predict the masked word (langauge modeling)

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

Supervised Learning Step

Model: (pre-trained in step #1)



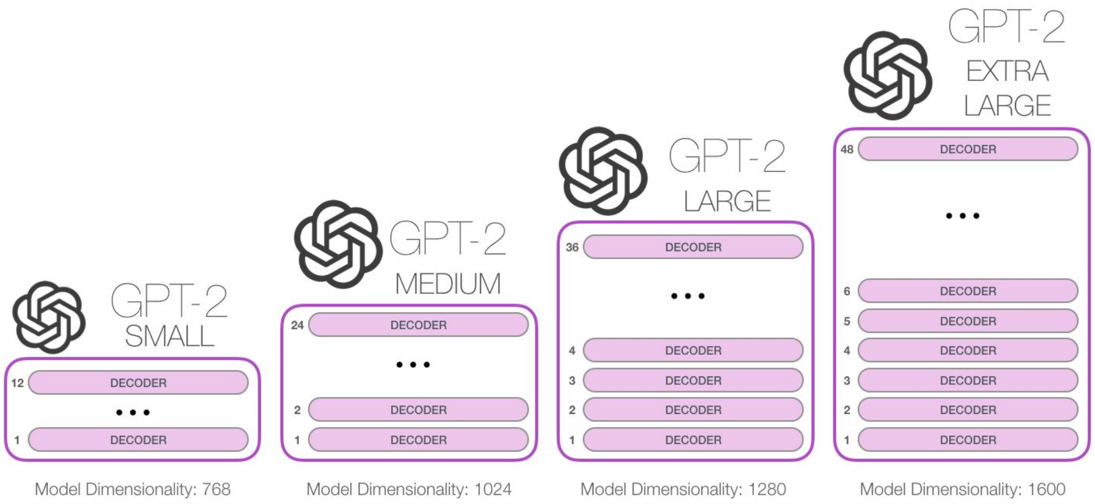
Dataset:

Email message	Class
Buy these pills	Spam
Win cash prizes	Spam
Dear Mr. Atreides, please find attached…	Not Spam

Adapted from: https://jalammar.github.io/illustrated-transformer/

Analytics Vidhya

Autoregressive (Decoder only) Transformer - GPT



- "GPT" stands for **G**enerative **P**re-**T**raining
- Stacked Decoder Model which is a Transformer
- Trained on the classic language modeling task of 'Next Word Prediction'
- Trained on the massive 40GB WebText dataset

Adapted from: https://jalammar.github.io/illustrated-gpt2/

Analytics Vidhya

Standard usage of Transformers

- **Auto-Encoding Transformers**
  - **Task Usage:** Primarily used for tasks like sentence representation, where understanding the context and meaning of input text is crucial.
  - **Examples:** BERT (Bidirectional Encoder Representations from Transformers) is a notable example, used for tasks like text classification, sentiment analysis, and question answering.
- **Auto-Regressive Transformers**
  - **Task Usage:** Suited for generating sequences where the prediction of the next element in the sequence depends on the previously generated elements.
  - **Examples:** GPT (Generative Pre-trained Transformer) series are classic examples, used for text generation, language modeling, and creative writing assistance.
- **Encoder-Decoder Transformers**
  - **Task Usage:** Designed for sequence-to-sequence tasks, where an input sequence is transformed into an output sequence.
  - **Examples:** BART, T5 are often used for tasks like machine translation and summarization
- **Multimodal Transformers**
  - **Task Usage:** Specialized in handling tasks that involve multiple types of data (e.g., text, images, audio) simultaneously.
  - **Examples:** CLIP and DALL-E are examples of multimodal transformers, used in visual question answering, image captioning, and content generation that combines text and visuals.