aivancity
SCHOOL FOR
TECHNOLOGY, BUSINESS & SOCIETY
PARIS-CACHAN

PGE 5

Generative AI

2023-2024
Y. Almehio

Generative AI

- Gen AI definition
- Primary Gen AI models
- GenAI Model Types
- How does Gen AI works (fast shot)
  - Language models
  - Evolution from simple to complex models
- Transformers
  - From RNN to Transformers
  - Architecture
  - Attention mechanisms
  - Project
  - Using trained transformers
  - Vision transformers
  - GIT transformers
  - Project

aivancity
PARIS-CACHAN

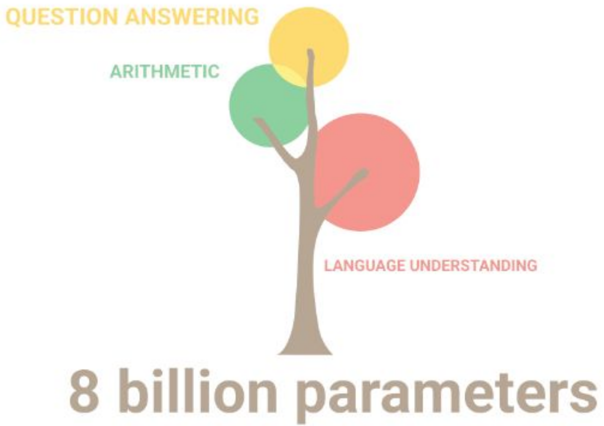
Generative AI

- Transformers
  - Representation
  - Modeling
- Transformer trained for GPT
- LLM models
  - Capacity toward human level performance
  - AI evolution
  - What is LLM model
  - How is LLM trained
- Ways of using LLM
  - Fine-tuning: LoRa
  - Prompting (prompt engineering)
  - 2 projects
  - Platforms for LLM usage
  - Retrieval Augmented generation: RAG
  - Tools to work with LLM
  - Project
- Final project description

aivancity
PARIS-CACHAN

Part 1

Generative AI from simple to today's LLM



aivancity
PARIS-CACHAN

The machine-generated image based on text input

What is Generative AI

- CV: generative models can generate realistic images, modify existing ones, or complete partial images.
- NLP: they can be used for tasks like language translation and text synthesis.
- Generative models help in creating conversational agents that can produce human-like responses.

- Capable of art generation, data augmentation, and creating synthetic medical images for research and diagnosis, showcasing its versatility and creativity.
- it raises ethical concerns due to the potential misuse in creating convincing fake content, leading to efforts in developing detection and mitigation techniques.

www.bing.com/images/create/

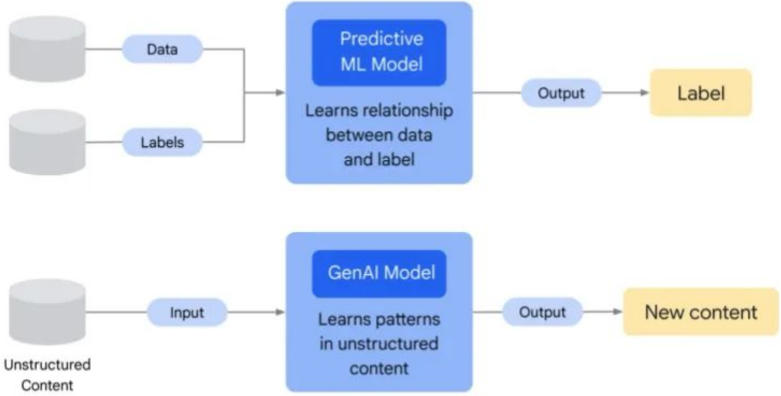
Part 1

What is Generative AI

Generative AI: based on deep learning (ANN), generates text, images, audio, and synthetic data. It operates via supervised, unsupervised, or semi-supervised learning, encompassing:

1. **? Models:** Predict labels from data features using labeled datasets.

2. **? Models:** These create new data instances by learning the **probability** distribution of existing data, thus generating new content.



what is the generative architecture ? in two
NLP, CNN

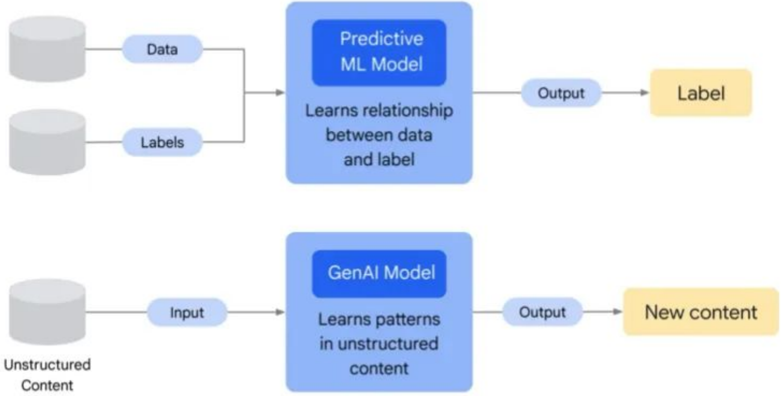
aivancity
PARIS-CACHAN

What is Generative AI

Generative AI: based on deep learning (ANN), generates text, images, audio, and synthetic data. It operates via supervised, unsupervised, or semi-supervised learning, encompassing:

1. **Discriminative Models:** Predict labels from data features using labeled datasets.

2. **Generative Models:** These create new data instances by learning the **probability** distribution of existing data, thus generating new content.



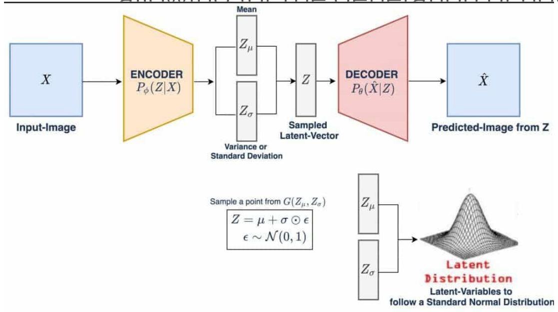
aivancity
PARIS-CACHAN

Part 1

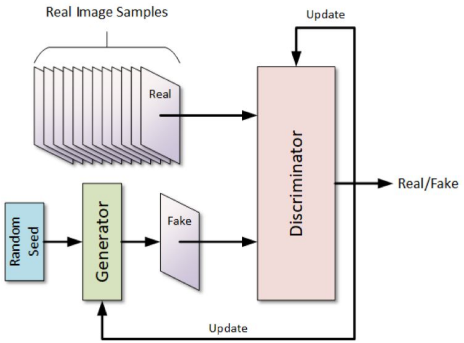
Primary Generative AI Models

Variational Autoencoders (VAEs): Generative models using an encoder and decoder network pair.

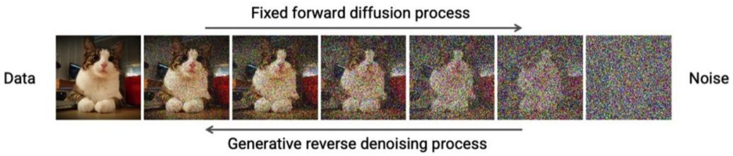
1. Encoder-Decoder Structure
2. Latent Space Representation
3. Reconstruction and Regularization: VAEs learn by reconstructing the input data from the latent representation while ensuring that the latent space has good properties, allowing for the generation of novel,



Generative adversarial networks (GANs), discovered in 2014, consist of two competing neural networks: a generator that creates new data examples, and a discriminator that evaluates them as real or fake. Once the most popular generative model, GANs facilitate a dynamic approach to training models to produce high-quality, realistic outputs.



Diffusion models, or denoising diffusion probabilistic models (DDPMs), are generative models using a two-step process of forward and reverse diffusion. Forward diffusion gradually adds noise to data, and reverse diffusion reconstructs the original by progressively removing noise. This method allows for the generation of novel data from a random noise starting point.

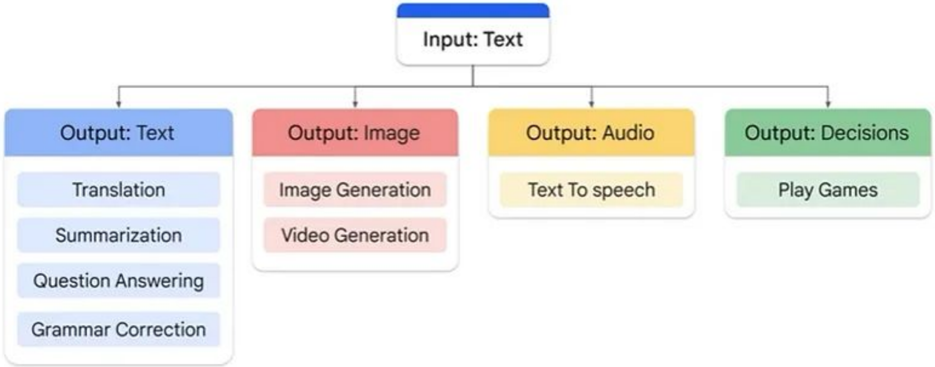


Transformers

Part 1

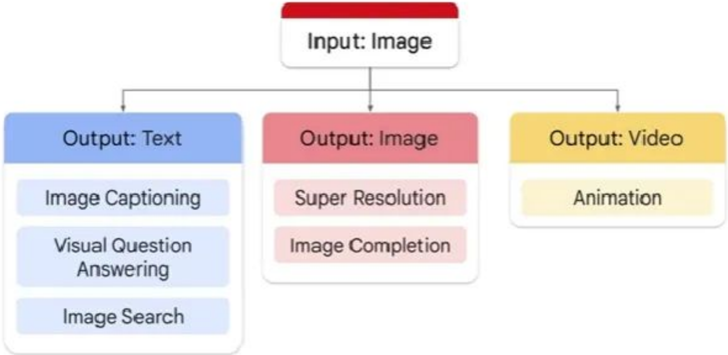
Generative AI Models Types

Generative language models



- **Text**: Translations, summarization, question answering, and grammar correction
- **Image**: Image and video generation
- **Audio**: Text to speech
- **Decisions**: Play games
- Examples: PaLM API for chat, PaLM API for Text, and BERT

Generative image models



- **Text**: Image captioning, visual question answering, and image search
- **Image**: Super-resolution or image completion
- **Video**: Animation
- Examples: Stable Diffusion v1-, BLIP image Captioning, BLIP VQA, CLIP, and Vit GPT2

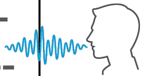
aivancity PARIS-CACHAN

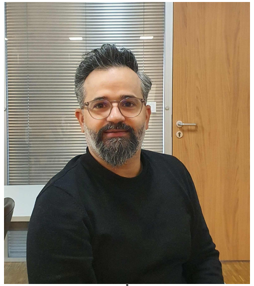
Part 1

Generative AI Models Types

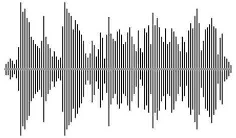
Input

Text--------





Output



Generated audio

Generated Video

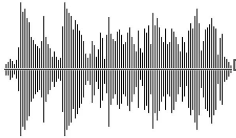
Idea

prompting

Input

Generated text

Output



Generated audio

Part 1

How does Generative AI works

LLM models

Capabilities in a nutshell

- Statistical text prediction.
- Impressive text generation capabilities.
- Interesting applications scenarios if carefully controlled.

Caveats in a nutshell

Part 1

How does Generative AI works

LLM models

Capabilities in a nutshell

- Statistical text prediction.
- Impressive text generation capabilities.
- Interesting applications scenarios if carefully controlled.

Caveats in a nutshell

- Foundation models are expensive to build and run. ⇒ GPT?
- Built from largely uncurated training data.
- No control over output quality (hallucinations, bias).
- Outputs must be validated.

How does Generative AI works

Language model: Likelihood of a sequence of words

"Eyes awe of an.."
"I saw a van"

"Sea, see the sight"
"See, sea the sight"

- Both sentences are plausible based on the sounds produced.
- The task is to determine which sentence is more likely to be said.
- A language model technology selects the more probable of the two sentences.

Prob("Sea, see the sight") > *Prob*("See, sea the sight")

aivancity
PARIS-CACHAN

Part 1

How does Generative AI works

Language models: probability of next word

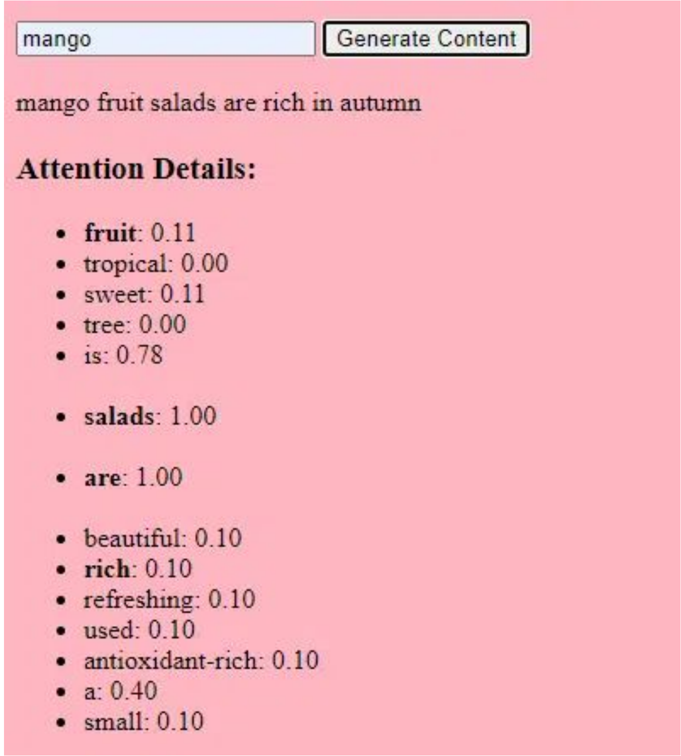
Early language models:
-> Start with a couple of words.

I want ..... *Pick a high-probability next word*

I want -> to .... *next word?*

I want to -> eat ... *next word? (pick one)*

I want to eat -> *{ lunch, my, Chinese, potatoes ... }*



aivancity
PARIS-CACHAN

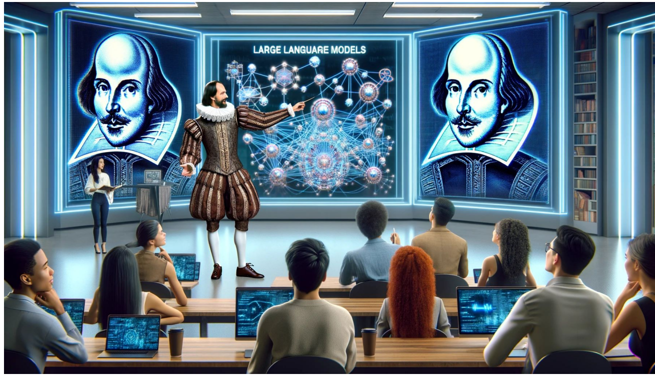
Part 1

How does Generative AI works
LLM models

Early Language Models:

- Example: n-gram models.
- Description: Simple statistical models predicting the next word based on the previous 'n-1' words.

We can train a simple next-word model using the works of Shakespeare. Then create more Shakespeare.



aivancity PARIS-CACHAN

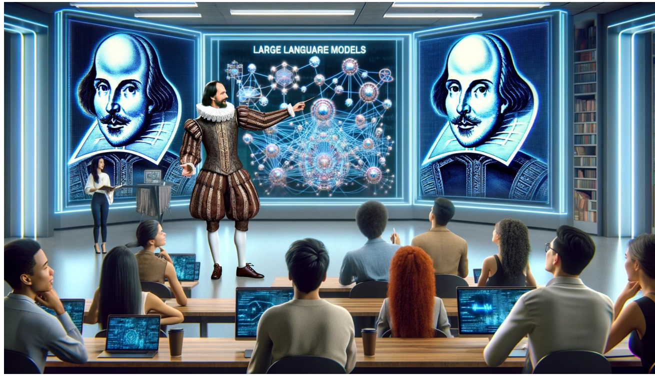
How does Generative AI works
LLM models

Early Language Models:

- Example: n-gram models.
- Description: Simple statistical models predicting the next word based on the previous 'n-1' words.

We can train a simple next-word model using the works of Shakespeare. Then create more Shakespeare.

I.e using 3-words . Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, 'tis done.



- Using a 3-word window to pick a likely next word.
- Using a 4 words grams model ends up plagiarizing Shakespeare pretty quickly.)

We have huge quantities of digital text for training which we didn't have before

Part 1

Evolution from simple to complex models

LLM models

Ability to understand context (not only previous sequence)

Considering longer context

COnsidering huge amount of data

Memory

Computing performance

Models architecture

Datasets availability

RNN/CNN to Transformers

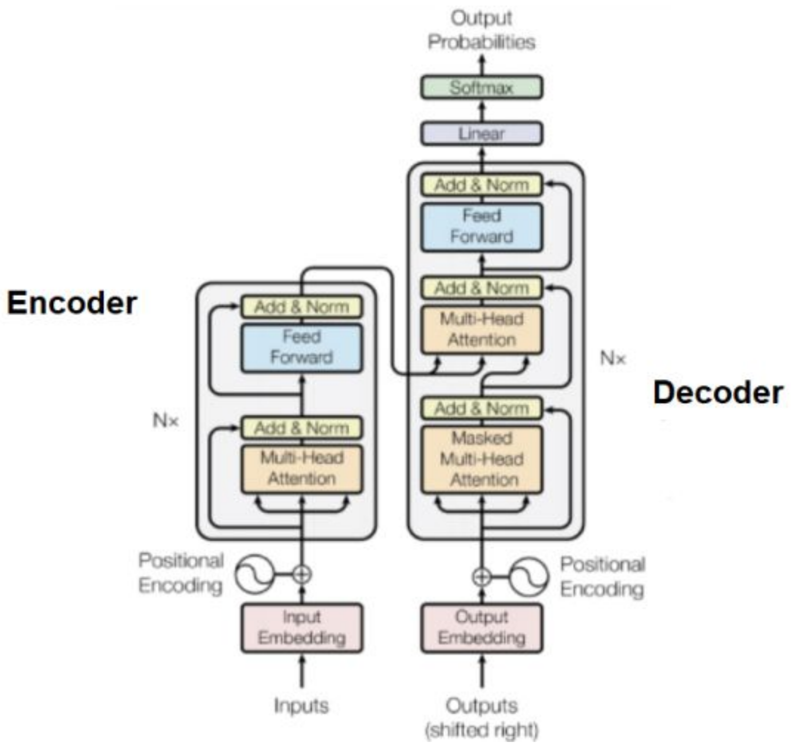
aivancity PARIS-CACHAN

Transformers

The foundation for LLMs and generative AI.

Part 1

Transformers: from RNN-based to transformers



Game changer: google paper

Attention is all you need: Transformer

- **key to new architectures** replacing older models.
- Architecture revolutionized NLP and beyond (meet CV)
-

Part 1

RNN: How do work?

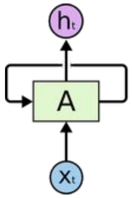
Early model : next word prediction

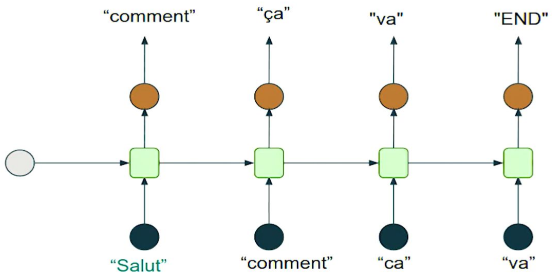
Task modeling using Seq2Seq:

- models can capture local dependencies and process sequentially
- Some challenges: requires more memory, sensitive to noise and lacking of interpretability
- Word position is important, impacted by the neighbors

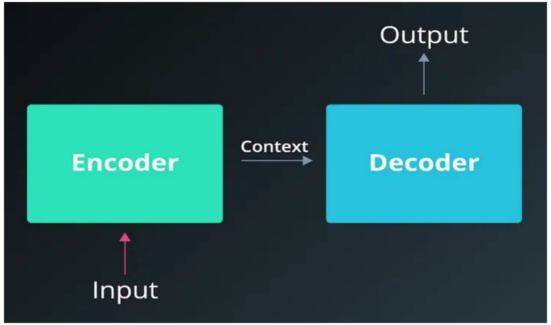
using Seq2Vec?

(output a value)





RNN
Predict next word

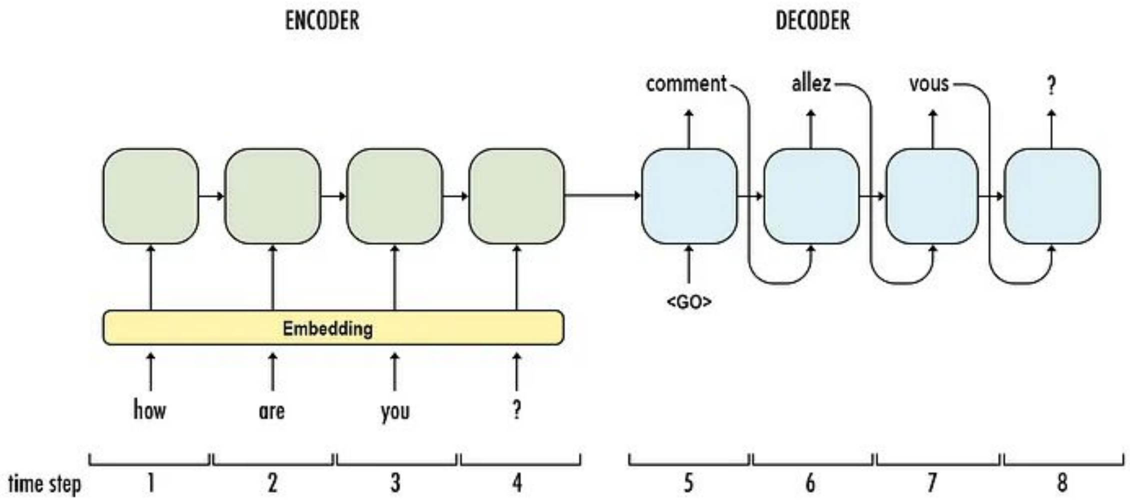


RNN Autoencoder

Part 1

RNN: How do work?

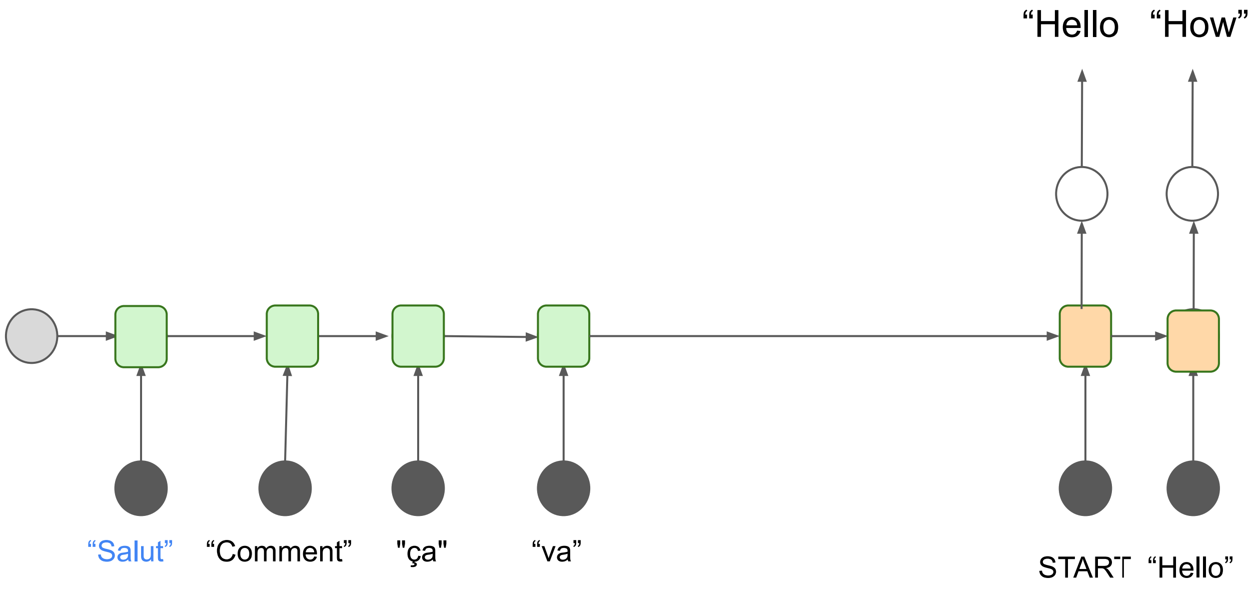
Early model : machine translation (seq2seq)



Part 1

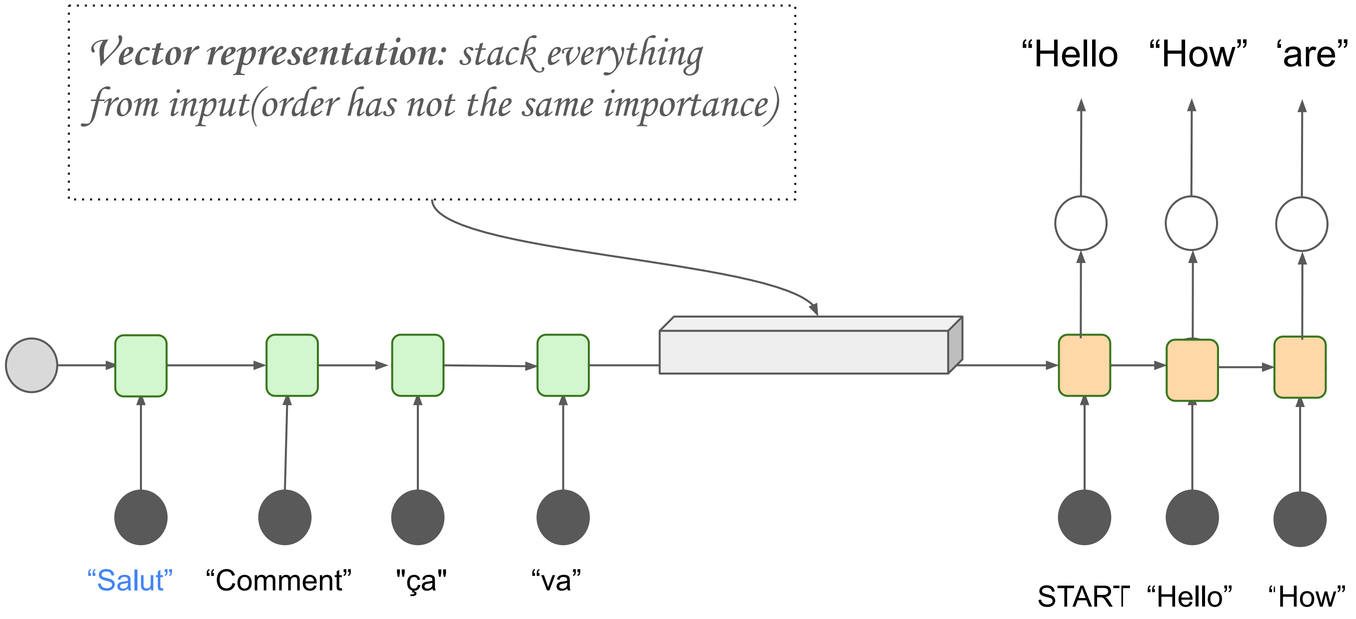
RNN: How do work?

Early model : machine translation (seq2seq)



RNN: How do work?

Early model : machine translation (seq2seq)

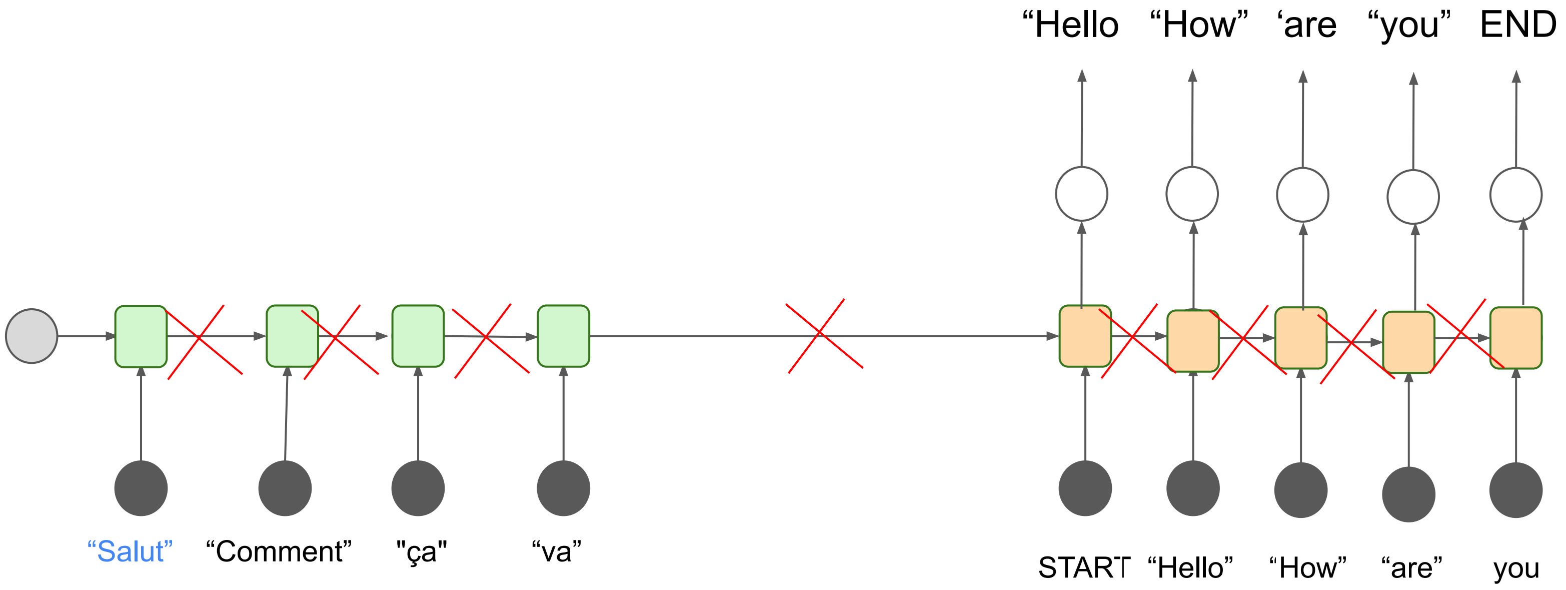


Part 1

RNN: How do work?

Sequential to parallelization

RNN Drawbacks

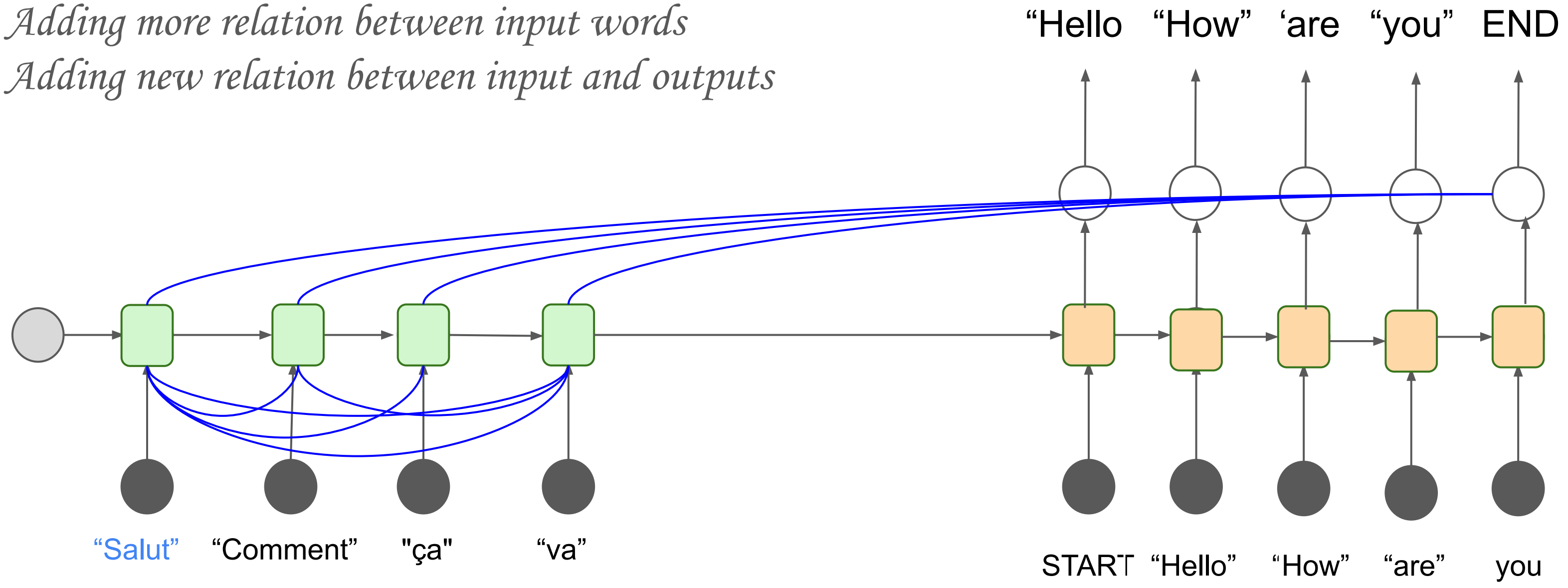


Part 1

RNN: How do work?

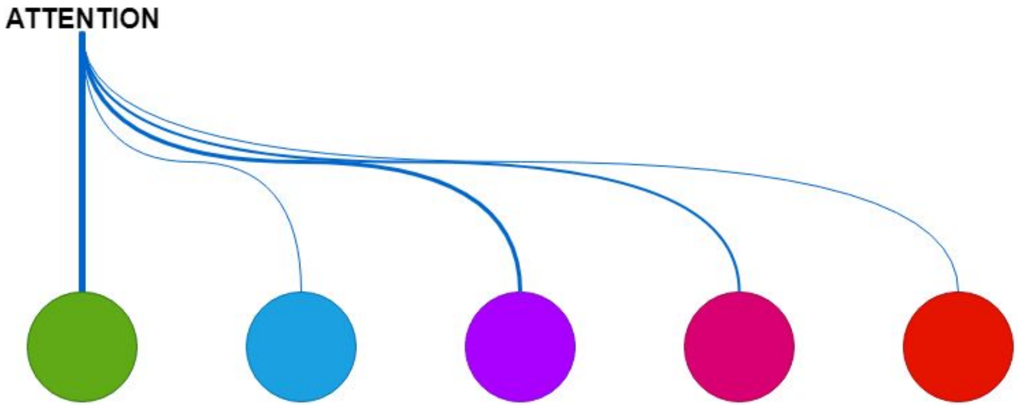
How to make attention

Adding more relation between input words
Adding new relation between input and outputs



Part 1

Attention is all you need: Transformers

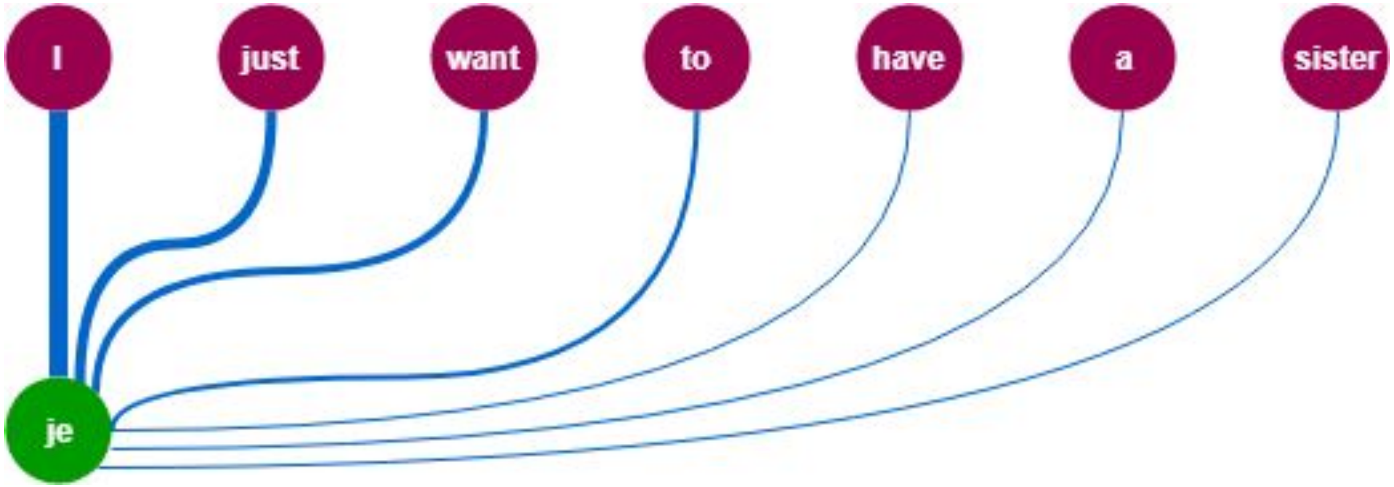


[Game changer: google paper]

The training process will strengthen the model's ability to capture relationships between input words

Part 1

Attention is all you need: Transformers

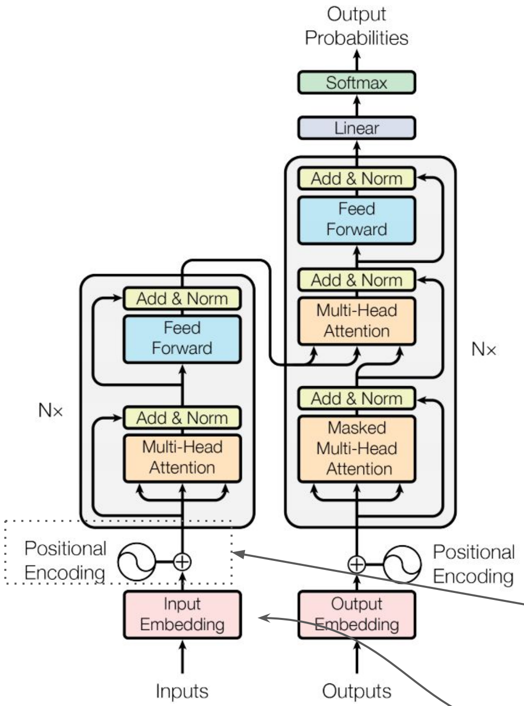


[Game changer: google paper]

Attention is all you need: Transformer

Part 1

Transformers: Architecture



Autoencoder
Self-attention
Multihead attention

In training, no sequential feeding: all input and and output are fed once

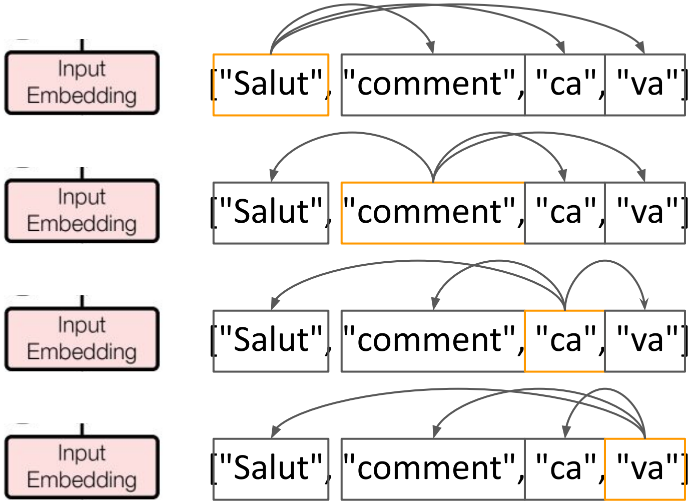
?

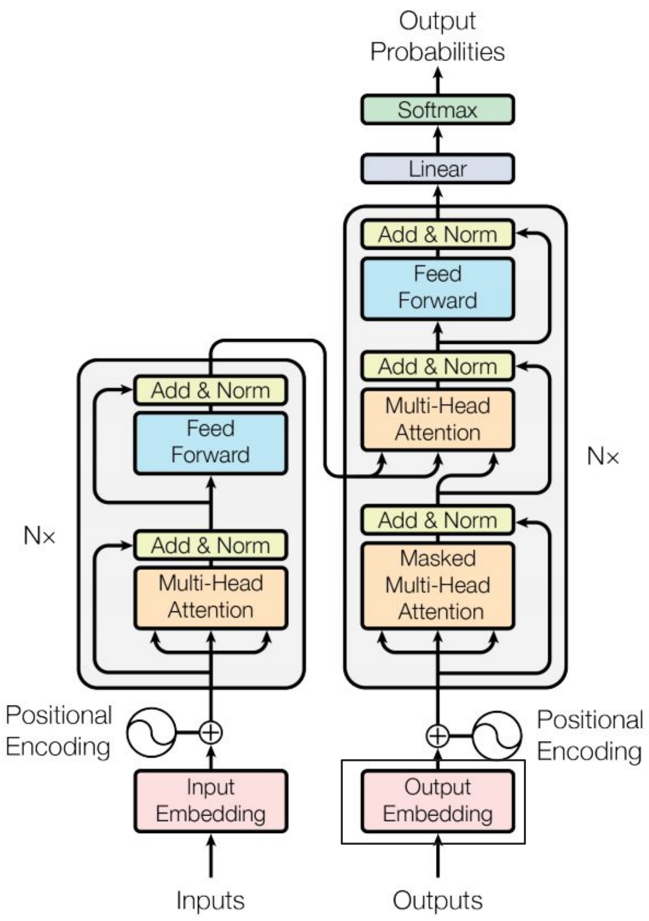
Embedding 512

Part 1

Transformers: Attention

Self -attention

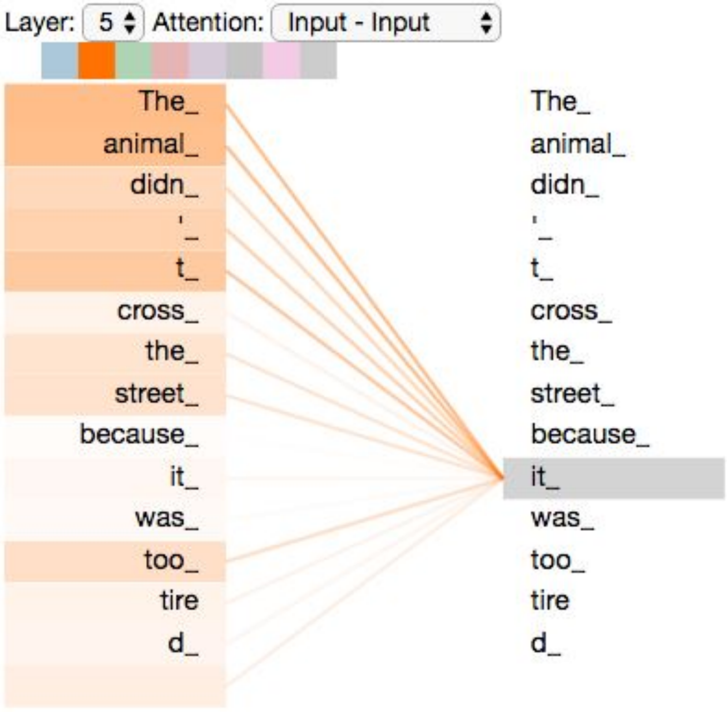


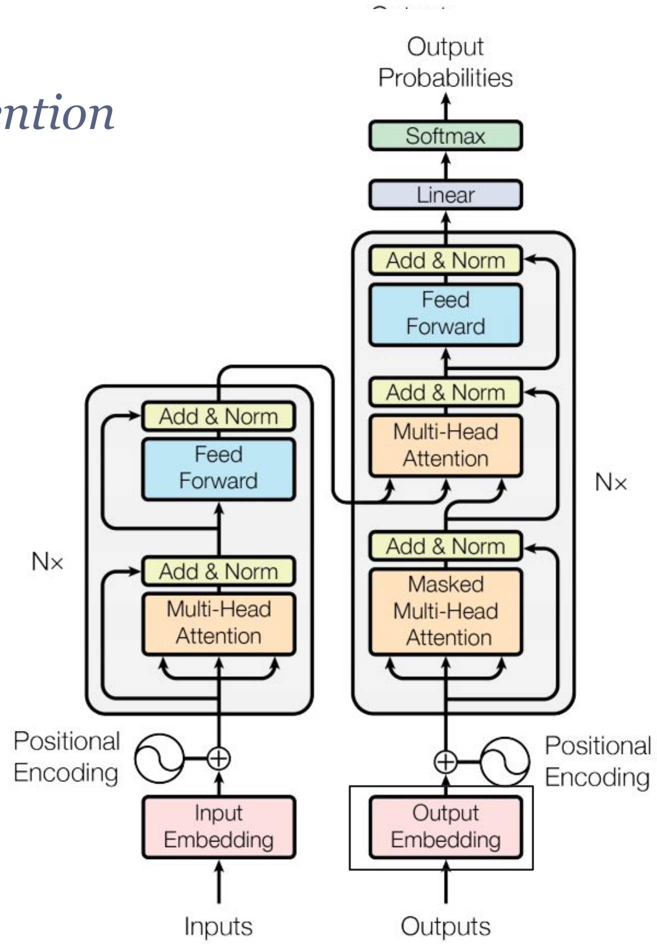


Part 1

Transformers: Attention

Self -attention: training finds relational attention



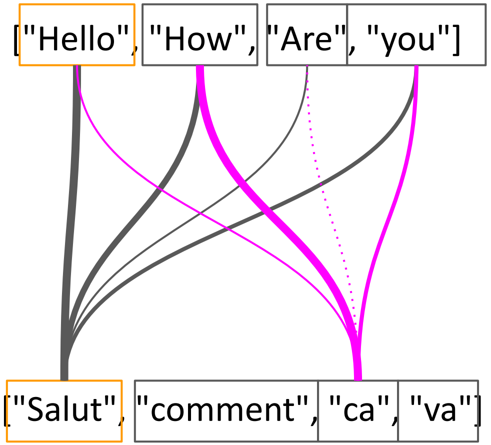


Tensor2Tensor notebook where you can load a Transformer model, and examine it using this interactive visualization.

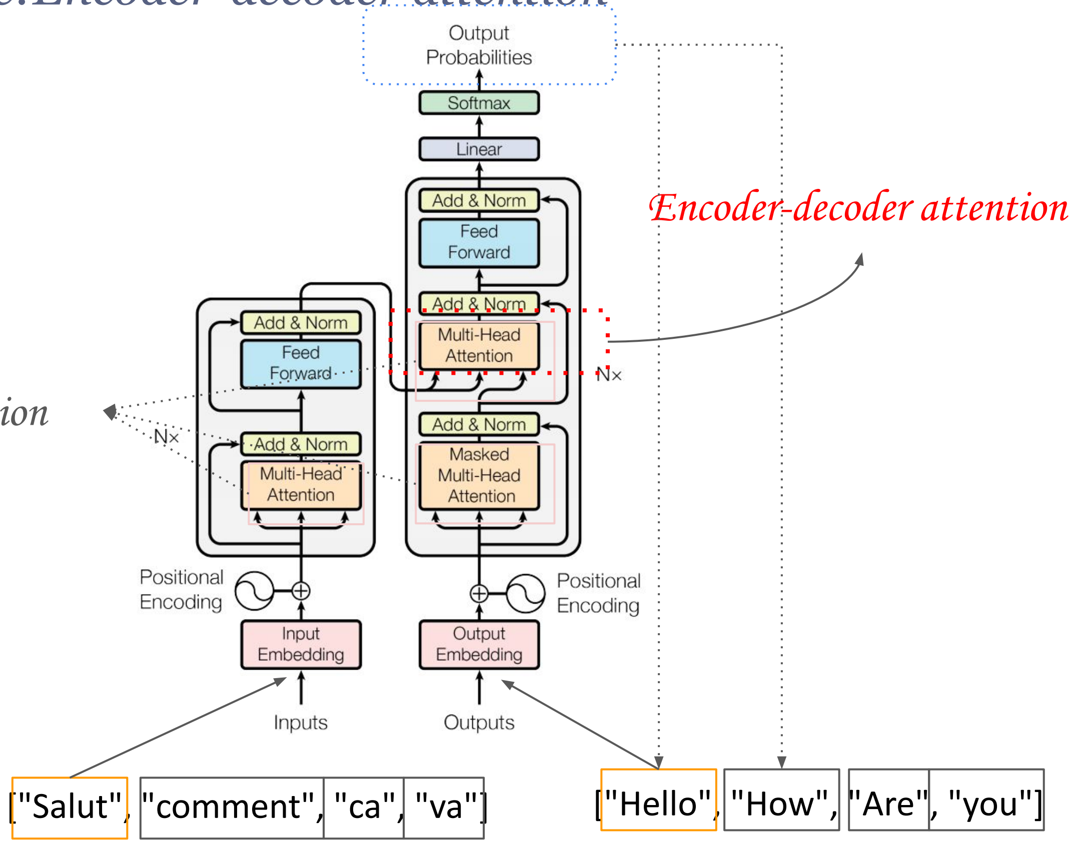
Part 1

Transformers: Attention

Self-attention:I/O fed once:Encoder-decoder attention



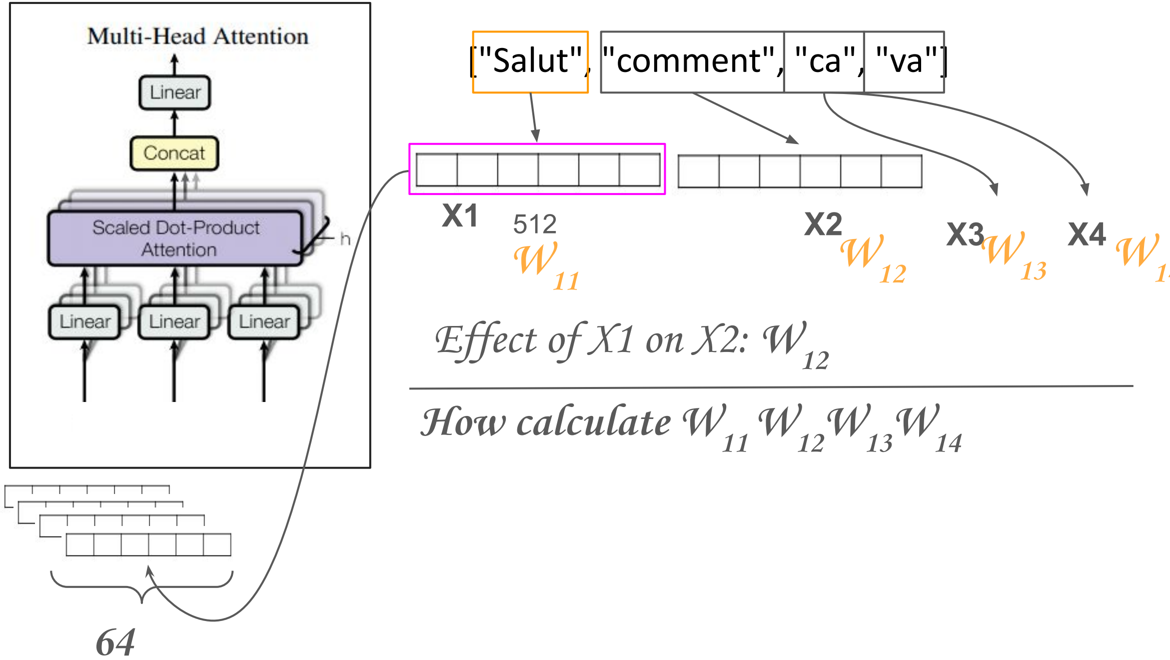
Self-attention

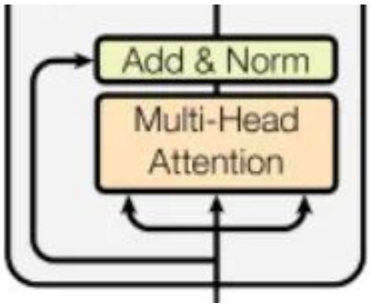


Part 1

Transformers: Attention

Multi-Head Attention





W_{11} W_{12} W_{13} W_{14}

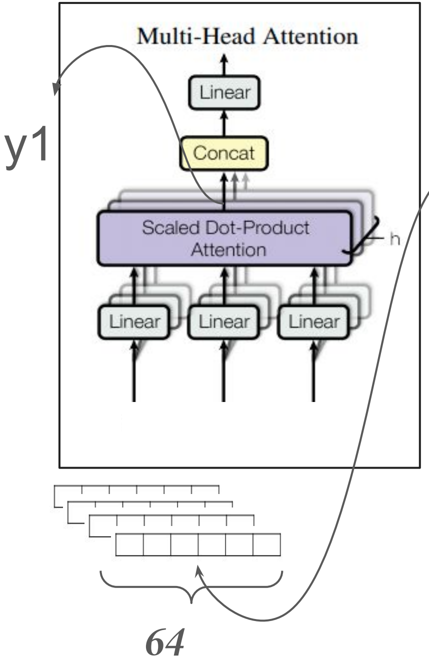
Effect of X1 on X2: W_{12}

How calculate W_{11} W_{12} W_{13} W_{14}

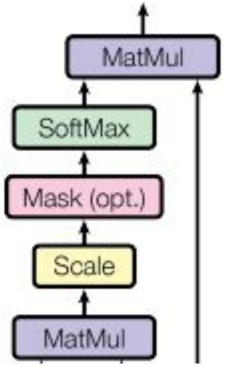
Part 1

Transformers: Attention

Multi-Head Attention



Scaled Dot-Product Attention



["Salut", "comment", "ca", "va"]

512

X1 W_{11} X2 W_{12} X3 W_{13} X4 W_{14}

Effect of X1 on X2: W_{12}

How calculate: $W_{11}W_{12}W_{13}W_{14}$

dot product $\Rightarrow$ X1.X1 X1 .X2 X1 .X3 X1 .X4

S11 S12 S13 S14

Norm $\Rightarrow$ $\sum w = 1$

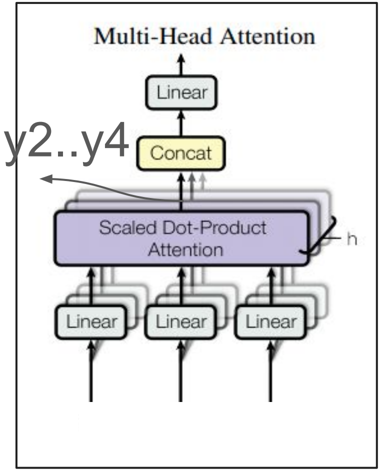
W_{11} W_{12} W_{13} W_{14} $\Rightarrow$ X1 $\star W_{11}$ + X2 $\star W_{12}$ + X3 W_{13} + X4 W_{14} = y1

Dot prod vectors

Part 1

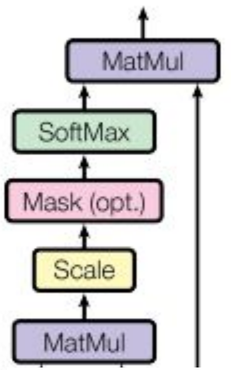
Transformers: Attention

Multi-Head Attention



y1, y2..y4

Scaled Dot-Product Attention



["Salut", "comment", "ca", "va"]

512

X1 W_{21} X2 W_{22} X3 W_{23} X4 W_{24}

Effect of X1 on X2: W_{12}

How calculate: $W_{11}W_{12}W_{13}W_{14}$

dot product ⇒ X2.X1 X2 .X2 X2 .X3 X2 .X4

S11 S12 S13 S14

Norm ⇒ $\sum w = 1$

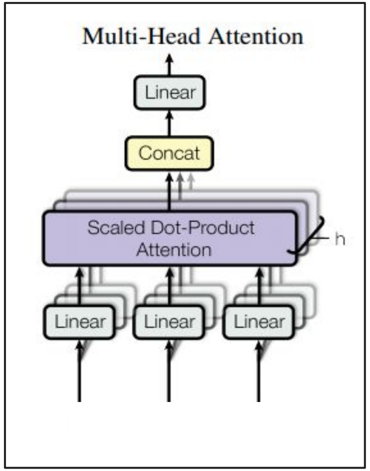
W_{21} W_{22} W_{23} W_{24}

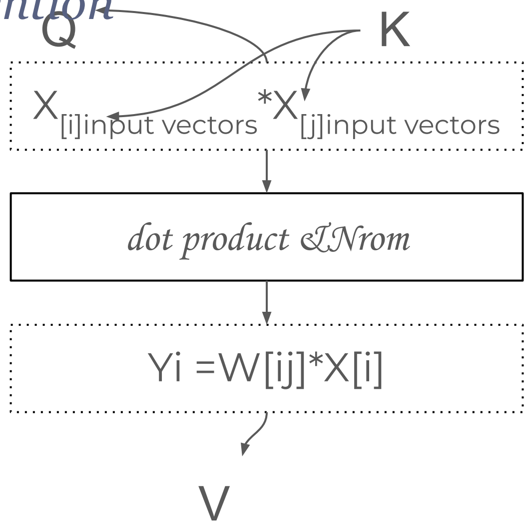
⇒ X1 $\star W_{21}$ + X2 $\star W_{22}$ + X3 W_{23} + X4 W_{24} = **y2**

Part 1

Transformers: Attention

Multi-Head Attention





- Dot products to find attention : are static math operation (attention for similarity), no training ⇒ For modeling other attentions(similarity, contextual relations, syntax...), it's better to add learnable weights.

Query Q: the word vector used in investigation, i.e X1

Keys K : the values vectors of other words, i.e X2….X4

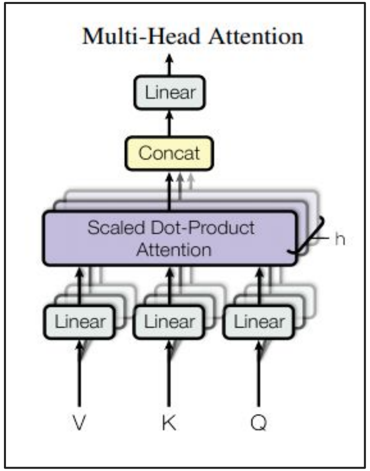
example

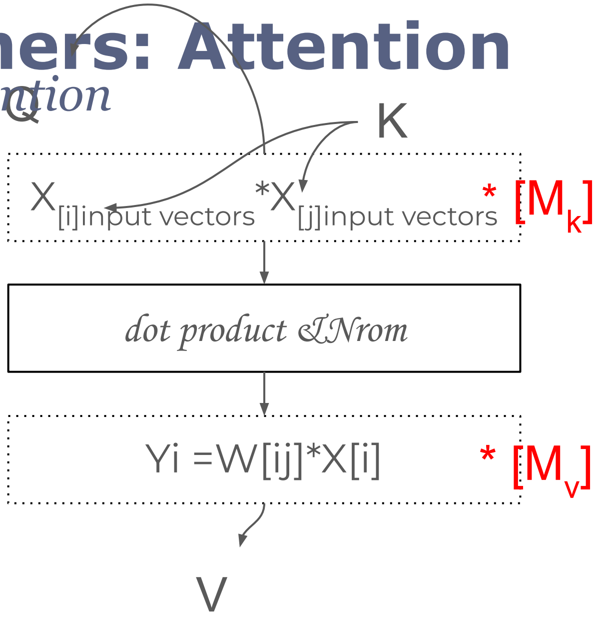
Values V: The results of Q*K to get Y1, Y2..

Part 1

Transformers: Attention

Multi-Head Attention





- Dot products are not enough ⇒ add matrices of weights to learn more attentions

Query Q: the word vector used in investigation, i.e X1

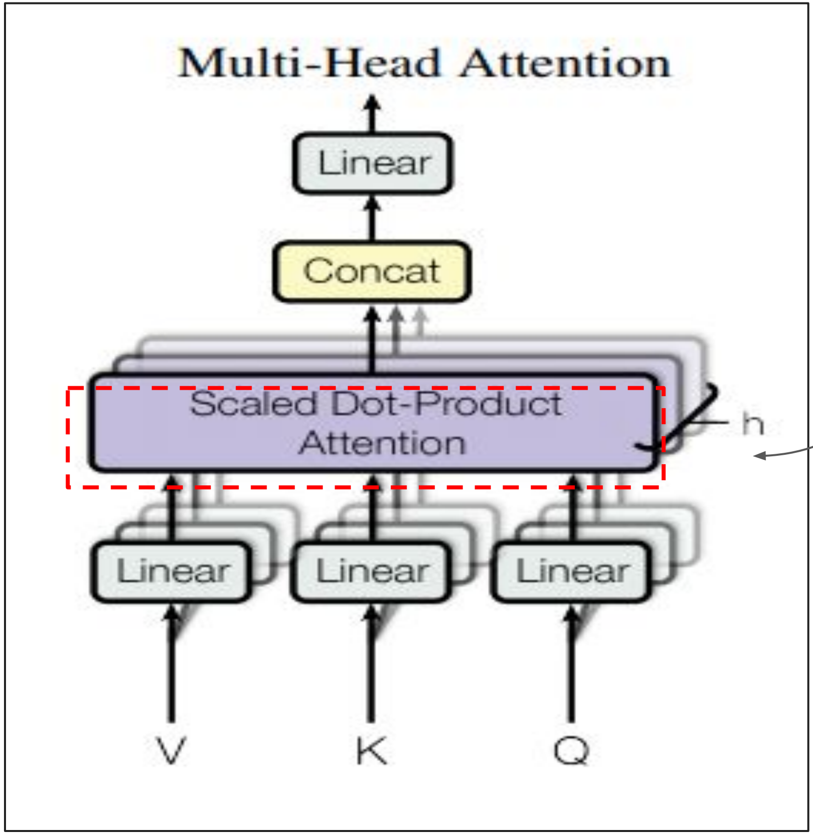
Keys K : the values vectors of other words, i.e X2….X4

Values V: The results of Q*K to get Y1, Y2..

Part 1

Transformers: Attention

Multi-Head Attention



- This is only for one head attention

TO capture more and more attention, we repeat it few times, how much ?

H = 8

Part 1

Transformers: Attention

Take a time to capitalize your understanding :
Helpful illustration:
https://www.tensorflow.org/text/tutorials/transformer

Part 1

Transformers

Project

- In this project, we will be using the **transformer** model, similar to introduced in this paper **Attention Is All You Need**. Specifically, we will be using the BERT (Bidirectional Encoder Representations from Transformers) model from this paper.

- **Transformer** models are considerably larger than anything else covered in Deep Learning. As such we are going to use the transformers library to get pre-trained transformers and use them as our embedding layers. We will freeze (not train) the transformer and only train the remainder of the model which learns from the representations produced by the transformer.

Part 2

ViT Vision Transformers

- ViT: strong competitors to CNNs in image recognition tasks.
- outperform CNNs significantly in computational efficiency and accuracy, nearly 4x better.
- When applied to mid-sized datasets, standard Transformers show modest accuracy in image tasks compared to ResNets.
- Transformers (ViT) excel with larger datasets, achieving or surpassing state-of-the-art results in various image recognition benchmarks.

Published as a conference paper at ICLR 2021

arXiv:2010.11929v2 [cs.CV] 3 Jun 2021

AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE

Alexey Dosovitskiy[*,†], Lucas Beyer[*], Alexander Kolesnikov[*], Dirk Weissenborn[*], Xiaohua Zhai[*], Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, Neil Houlsby[*,†]
[*]equal technical contribution, [†]equal advising
Google Research, Brain Team
{adosovitskiy, neilhoulsby}@google.com

ABSTRACT

While the Transformer architecture has become the de-facto standard for natural language processing tasks, its applications to computer vision remain limited. In vision, attention is either applied in conjunction with convolutional networks, or used to replace certain components of convolutional networks while keeping their overall structure in place. We show that this reliance on CNNs is not necessary and a pure transformer applied directly to sequences of image patches can perform very well on image classification tasks. When pre-trained on large amounts of data and transferred to multiple mid-sized or small image recognition benchmarks (ImageNet, CIFAR-100, VTAB, etc.), Vision Transformer (ViT) attains excellent results compared to state-of-the-art convolutional networks while requiring substantially fewer computational resources to train.[1]

1 INTRODUCTION

Self-attention-based architectures, in particular Transformers (Vaswani et al., 2017), have become the model of choice in natural language processing (NLP). The dominant approach is to pre-train on a large text corpus and then fine-tune on a smaller task-specific dataset (Devlin et al., 2019). Thanks to Transformers' computational efficiency and scalability, it has become possible to train models of unprecedented size, with over 100B parameters (Brown et al., 2020; Lepikhin et al., 2020). With the models and datasets growing, there is still no sign of saturating performance.

In computer vision, however, convolutional architectures remain dominant (LeCun et al., 1989; Krizhevsky et al., 2012; He et al., 2016). Inspired by NLP successes, multiple works try combining CNN-like architectures with self-attention (Wang et al., 2018; Carion et al., 2020), some replacing the convolutions entirely (Ramachandran et al., 2019; Wang et al., 2020a). The latter models, while

Part 1

ViT Vision Transformers

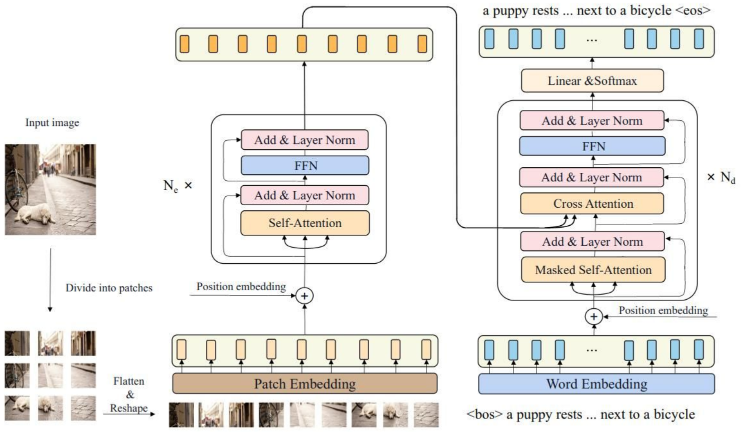
An image is worth a thousand words

Transformers say an image worth 9x9 words

Part 1

ViT Vision Transformers

- **Image Patching**: image is divided into fixed-size patches. These patches are treated similarly to tokens (words) in NLP tasks. Each patch is flattened into a one-dimensional vector.
- **Linear Embedding**: Each flattened patch is then mapped to a D-dimensional embedding space using a trainable linear projection. This step is analogous to word embeddings in NLP.
- **Positional Encoding:** To retain positional information, positional encodings are added to the patch embeddings. This is crucial since the transformer architecture itself doesn't have any notion of order or position.



transformer-image-captioning

Test: caption transformer (catr): [project]

Part 1

Transformers: fr

Nice project to run and learn

https://colab.research.google.com/github/keras-team/keras-io/blob/master/examples/vision/ipynb/image_captioning.ipynb

Part 1

Generative Image2Text (GIT)Transformers

vision-language tasks

Purpose and Task Focus:

- GIT: Designed specifically for generative vision-language tasks, it aims to create textual descriptions or answers from visual data, handling complex tasks like image captioning and question answering.

Architecture and Design:

- GIT: Simplifies the architecture into one image encoder and one text decoder under a single language modeling task, ViT: Adapts the transformer architecture originally designed for text to handle images, dividing the image into patches and processing these as if they were tokens in a sequence, primarily for understanding and classifying images.

- Project

GIT: A Generative Image-to-text Transformer for Vision and Language

Jianfeng Wang jianfw@microsoft.com
Zhengyuan Yang zhengyang@microsoft.com
Xiaowei Hu xiaowei.hu@microsoft.com
Linjie Li lindsey.li@microsoft.com
Kevin Lin keli@microsoft.com
Zhe Gan zhe.gan@microsoft.com
Zicheng Liu zliu@microsoft.com
Ce Liu ce.liu@microsoft.com
Lijuan Wang lijuanw@microsoft.com
Microsoft Cloud and AI

Abstract

In this paper, we design and train a **G**enerative **I**mage-to-text **T**ransformer, GIT, to unify vision-language tasks such as image/video captioning and question answering. While generative models provide a consistent network architecture between pre-training and fine-tuning, existing work typically contains complex structures (uni/multi-modal encoder/decoder) and depends on external modules such as object detectors/taggers and optical character recognition (OCR). In GIT, we simplify the architecture as one image encoder and one text decoder under a single language modeling task. We also scale up the pre-training data and the model size to boost the model performance. Without bells and whistles, our GIT establishes new state of the arts on numerous challenging benchmarks with a large margin. For instance, our model surpasses the human performance for the first time on TextCaps (138.2 vs. 125.5 in CIDEr). Furthermore, we present a new scheme of generation-based image classification and scene text recognition, achieving decent performance on standard benchmarks.

1 Introduction

Table 1: Comparison with prior SOTA on image/video captioning and question answering (QA) tasks. *: evaluated on the public server. CIDEr scores are reported for Captioning tasks. Prior SOTA: COCO(Zhang et al., 2021a), nocaps (Yu et al., 2022), VizWiz-Caption (Gong et al., 2021), TextCaps (Yang et al., 2021c),ST-VQA (Biten et al., 2022),VizWiz-VQA (Alayrac et al., 2022),OCR-VQA (Biten et al., 2022),MSVD (Lin et al., 2021a),MSRVTT (Seo et al., 2022),VATEX (Tang et al., 2021),TVC (Tang et al., 2021),MSVD-QA (Wang et al., 2022a),TGIF-Frame (Zellers et al., 2021),Text Recog. (Lyu et al., 2022). Details of GIT2 are presented in supplementary materials.

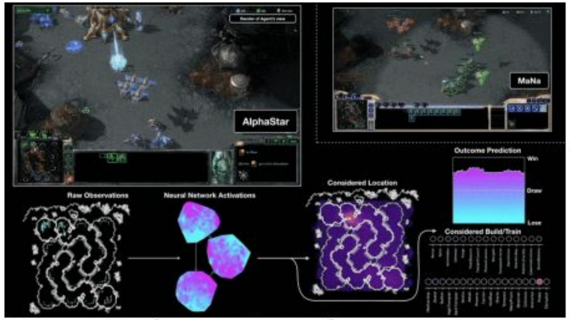
Image captioning	Image QA	Video captioning	Video QA	Text Rec.

Transformer is a general sequence processing tool…

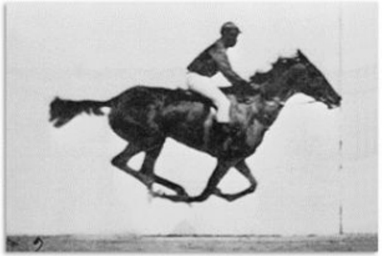
Audio Signal

Music notes

Image (as sequence of patches)



Action sequence in games



Video sequence

Protein sequence

One single architecture to *rule* them all.

"I want to beat 10 of them!" – Donnie Yen

Part 1

Task-based Transformers

Representation
How represent language to machine

Modeling
How model language statistically

Embedding: encoder

BERT((Bidirectional Encoder Representations from Transformers)

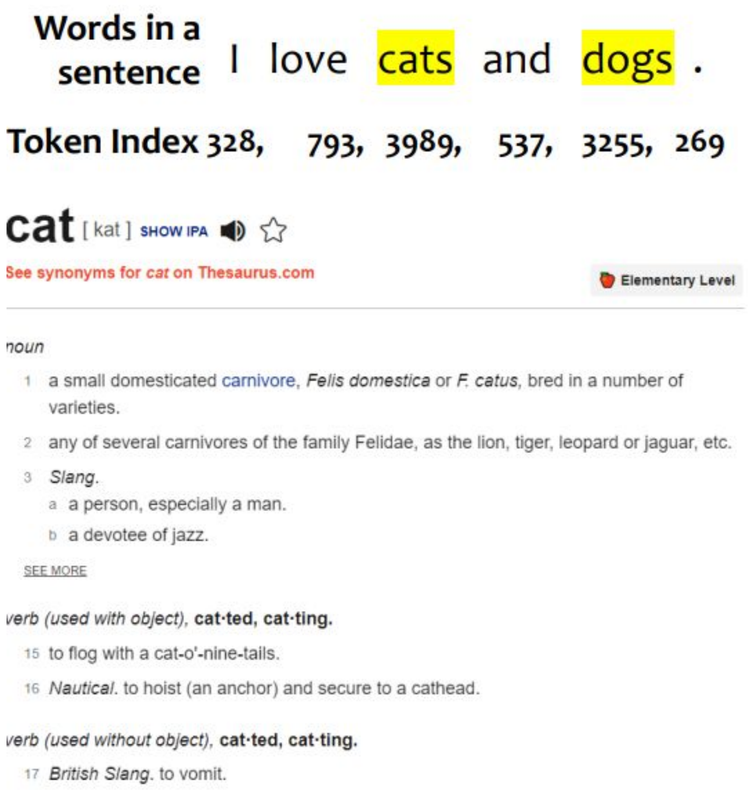
Predicting or generating new language data based on the learned representation

BART((Bidirectional Auto-regressive Transformers)
GPT
ChatGPT

Part 1

Task-based Transformers

- Unlike pixel, meanings of word are not explicitly in the characters.
- Word can be represented as number index
  - But indices are also meaningless.
- List of feature attributes (dictionary entry)
  - {Part of Speech, description, usage,…}
  - how to design those ?

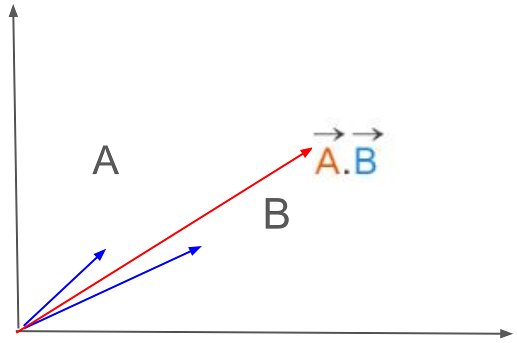


appendix

[Type of attentions, how it works]

Matrix Representation of Dot Product

cuemath THE MATH EXPERT

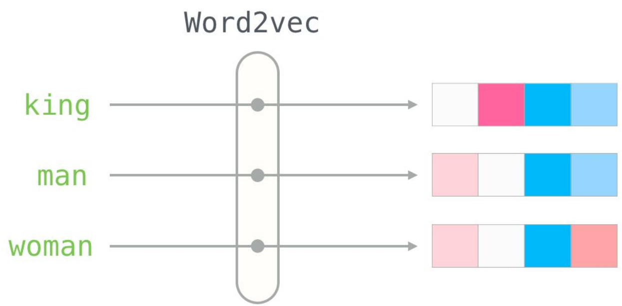


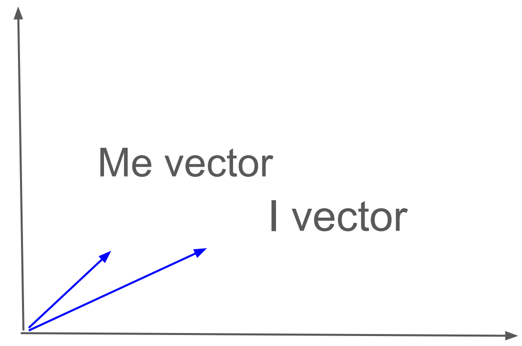
$$\vec{A}^{T} = \begin{bmatrix} A_1 & A_2 & A_3 \end{bmatrix} \qquad \vec{B} = \begin{bmatrix} B_1 \\ B_2 \\ B_3 \end{bmatrix}$$

$$\begin{bmatrix} A_1 & A_2 & A_3 \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ B_3 \end{bmatrix} = A_1B_1 + A_2B_2 + A_3B_3 = \vec{A}.\vec{B}$$

Come back

Embedding



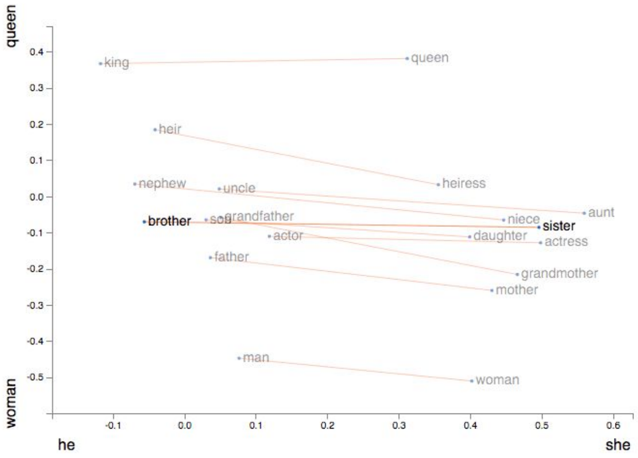


If represent a word be two digits , I, Me

Process Seq2Seq-Word embeddings

- Specific encoding: turns into vectors of numbers that capture a lot mf meaning and semantic information: similar words end up lying close to each other
- Into vector arithmetics to work with analogies
  - Example: relation ? between **king, man, woma, queen**
  -

[king - man + woman = queen]



1. **Word Embeddings:** Word2vec represents each word as a vector, a list of numbers. These vectors capture the meaning and context of the word.
2. **Vector Arithmetic:** Similar to adding or subtracting numbers, basic vector operations can be performed on word vectors.

Come back

Positional encoding

The formula for positional encoding is as follows:

PE(pos, 2i) = sin(pos / 10000^(2i/d_model))

PE(pos, 2i+1) = cos(pos / 10000^(2i/d_model))

where:

- pos: the position of the element in the sequence
- i: the dimension index of the positional encoding
- d_model: the dimension of the model's hidden states (or the size of the embedding)

Come back