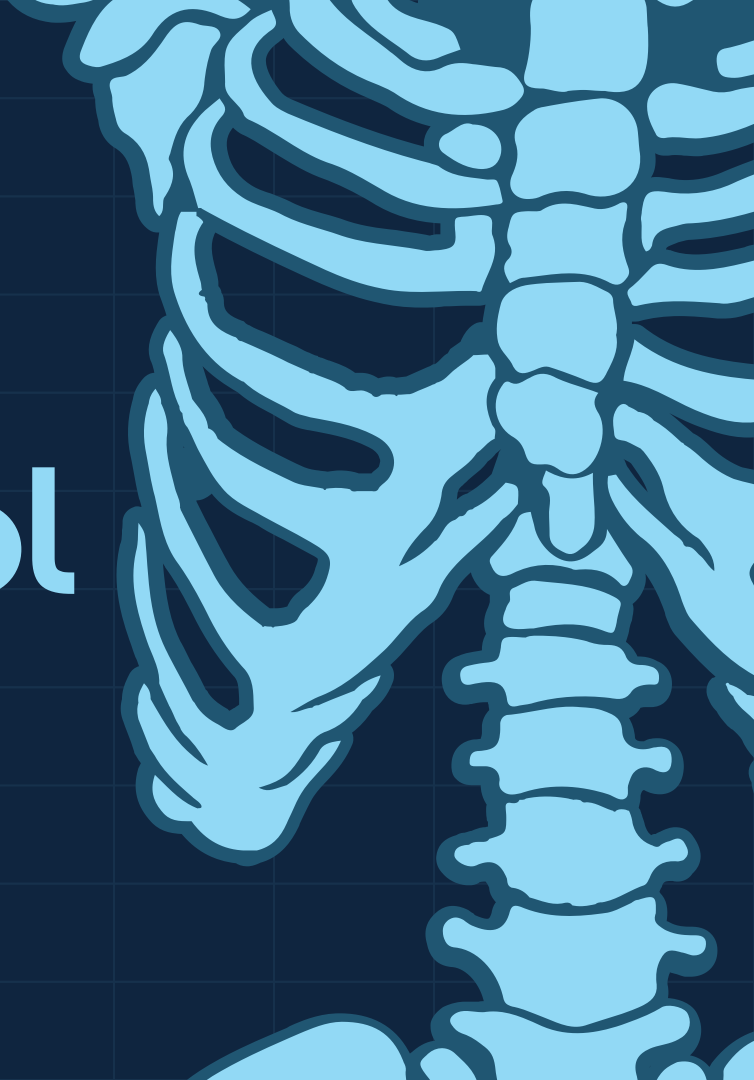
Chest Diagnosis Tool



10cm

Our team

Introduced By:

Mariam Khaled

Abdelrahman Tarek

Hassan Mohamed

Abdullah Shawwaf

Mohamed Magdy

Sarah Abdelmoaty

Mentored by :

Dr. Waleed M. Ead

10cm

Agenda

01

Business Problem

02

Data

03

Models

04

Inference and Evaluation

05

Further Experimentations

06

Future Work and Deployment

10cm

01

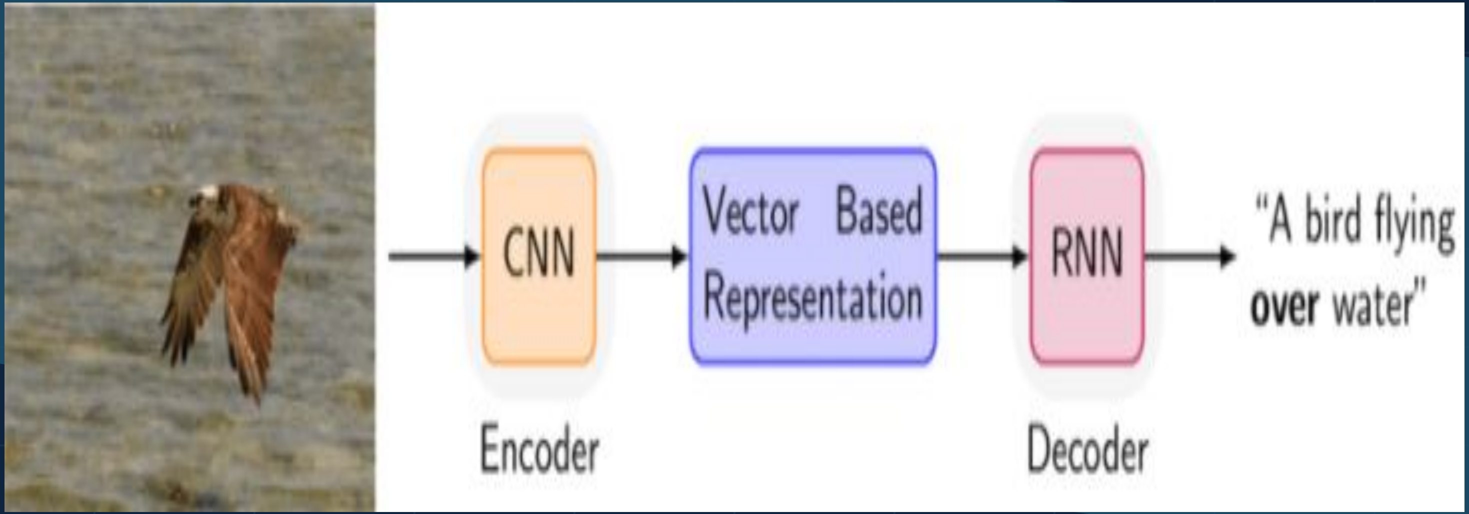
Business Problem

10cm

- The objective of this project is to build a deep learning model that automatically writes the impression part of the medical report of chest X-rays and alleviates some of the burdens of the medical profession.

10cm

Image Captioning



02

Data

10cm

Data Description

Chest X-ray Images

Publicly available dataset from *Indiana University*

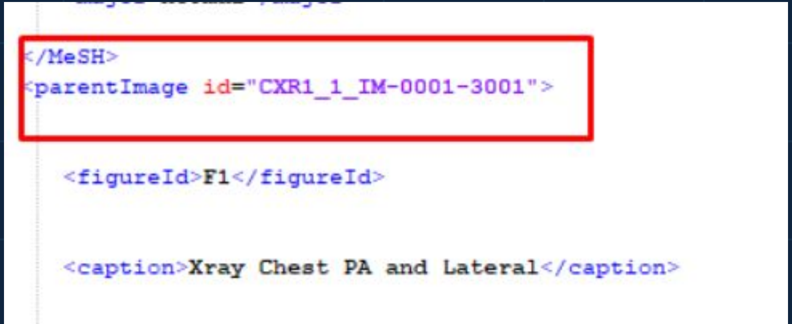
XML Reports

- Comparison
- Indication
- Findings
- Impression

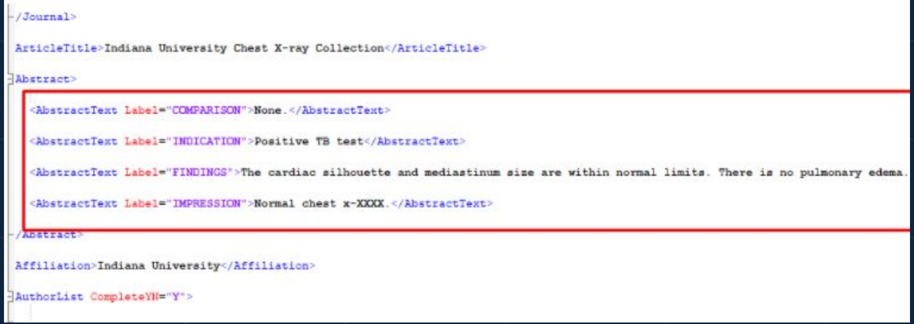
10cm

Data Preprocessing

- We are mainly interested with the *parentImage* tags to get the images associated with each report, and the *impression* tag as it is the target feature that we want the model to generate. We extracted them using regex.



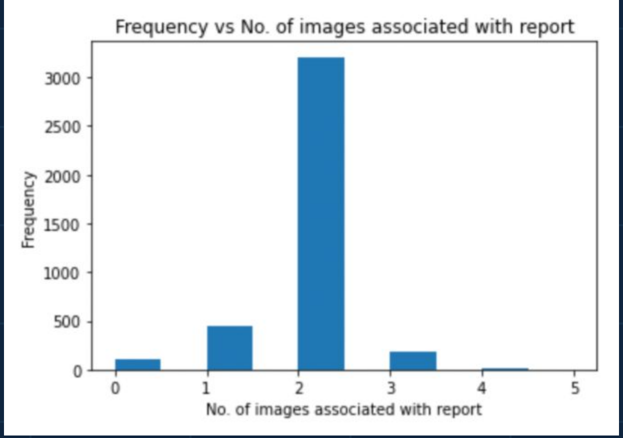
```
</MeSH>
<parentImage id="CXR1_1_IM-0001-3001">

  <figureId>F1</figureId>

  <caption>Xray Chest PA and Lateral</caption>
```



```
/Journal>
ArticleTitle>Indiana University Chest X-ray Collection</ArticleTitle>
Abstract>
  <AbstractText Label="COMPARISON">None.</AbstractText>
  <AbstractText Label="INDICATION">Positive TB test</AbstractText>
  <AbstractText Label="FINDINGS">The cardiac silhouette and mediastinum size are within normal limits. There is no pulmonary edema.
  <AbstractText Label="IMPRESSION">Normal chest x-XXXX.</AbstractText>
/Abstract>
Affiliation>Indiana University</Affiliation>
AuthorList CompleteYN="Y">
```

Data Preprocessing

- We plotted a histogram to find the minimum and maximum number of images associated with each report.
- We took two images as input, since it was found that two images was the most frequent case.



Data Preprocessing

- The resulting data frame was as follows.

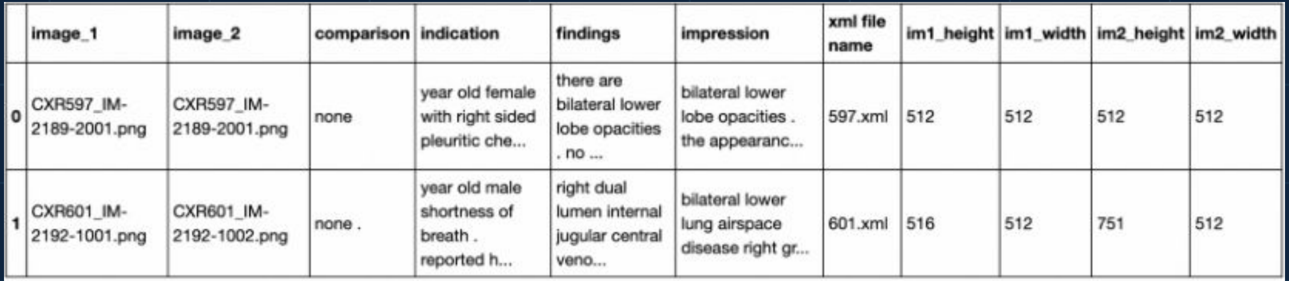
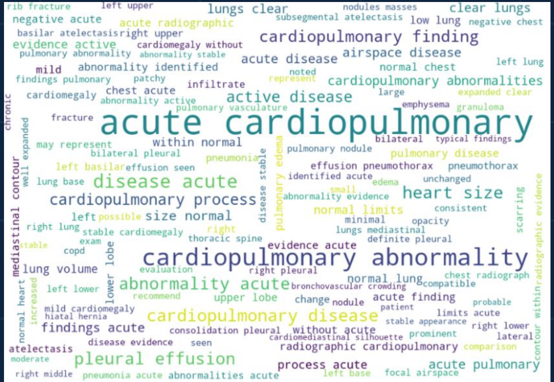


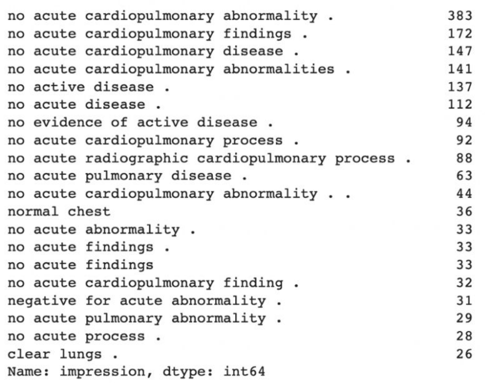
	image_1	image_2	comparison	indication	findings	impression	xml file name	im1_height	im1_width	im2_height	im2_width
0	CXR597_IM-2189-2001.png	CXR597_IM-2189-2001.png	none	year old female with right sided pleuritic che...	there are bilateral lower lobe opacities . no ...	bilateral lower lobe opacities . the appearanc...	597.xml	512	512	512	512
1	CXR601_IM-2192-1001.png	CXR601_IM-2192-1002.png	none .	year old male shortness of breath . reported h...	right dual lumen internal jugular central veno...	bilateral lower lung airspace disease right gr...	601.xml	516	512	751	512

- To deal with the missing values in the data frame, all the datapoints which had image_1 and impression value null were removed from the data frame. All missing values found in image_2 were filled with the same data path of that of image_1.
- Since pretrained models are modelled for square-sized images we chose 224*224*3 as the specified size of the images.

Data Preprocessing

- To examine the distribution of the impression values, we plotted the following word cloud.
- From the value counts, we can see that top 20 most frequently occurring words had the same meaning, suggesting one type of data is dominating for this data.
- We applied upsampling and downsampling to the data so that the model doesn’t over-fit.





```
no acute cardiopulmonary abnormality .              383
no acute cardiopulmonary findings .                 172
no acute cardiopulmonary disease .                  147
no acute cardiopulmonary abnormalities .            141
no active disease .                                 137
no acute disease .                                  112
no evidence of active disease .                      94
no acute cardiopulmonary process .                   92
no acute radiographic cardiopulmonary process .      88
no acute pulmonary disease .                         63
no acute cardiopulmonary abnormality . .             44
normal chest                                         36
no acute abnormality .                               33
no acute findings .                                  33
no acute findings                                    33
no acute cardiopulmonary finding .                   32
negative for acute abnormality .                     31
no acute pulmonary abnormality .                     29
no acute process .                                   28
clear lungs .                                        26
Name: impression, dtype: int64
```

03

Models

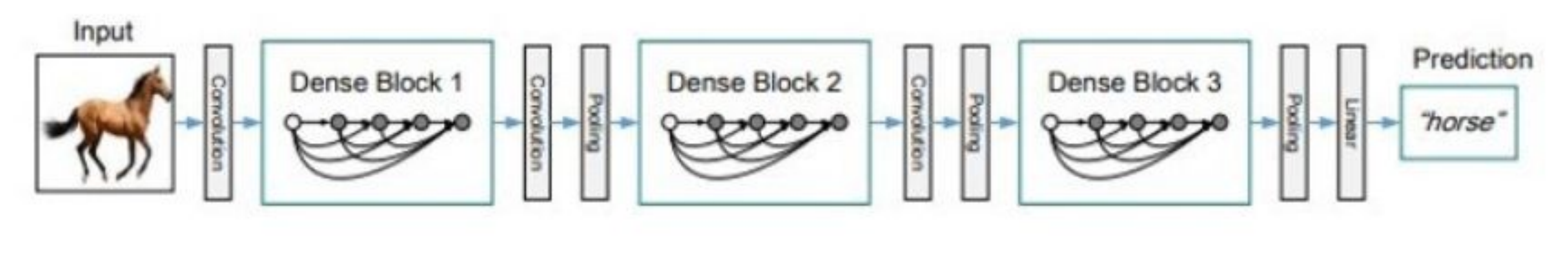
10cm

Image Captioning

- Our problem requires generating a textual description for given chest x-ray images, which is an image captioning problem.
- We need a computer vision model - CNN - to deal with the images, and an NLP model - RNN - to deal with text generation □ Encoder decoder Model.
- For the encoder □ CHEXNET model.
- For the decoder □ embedding layer + LSTM.

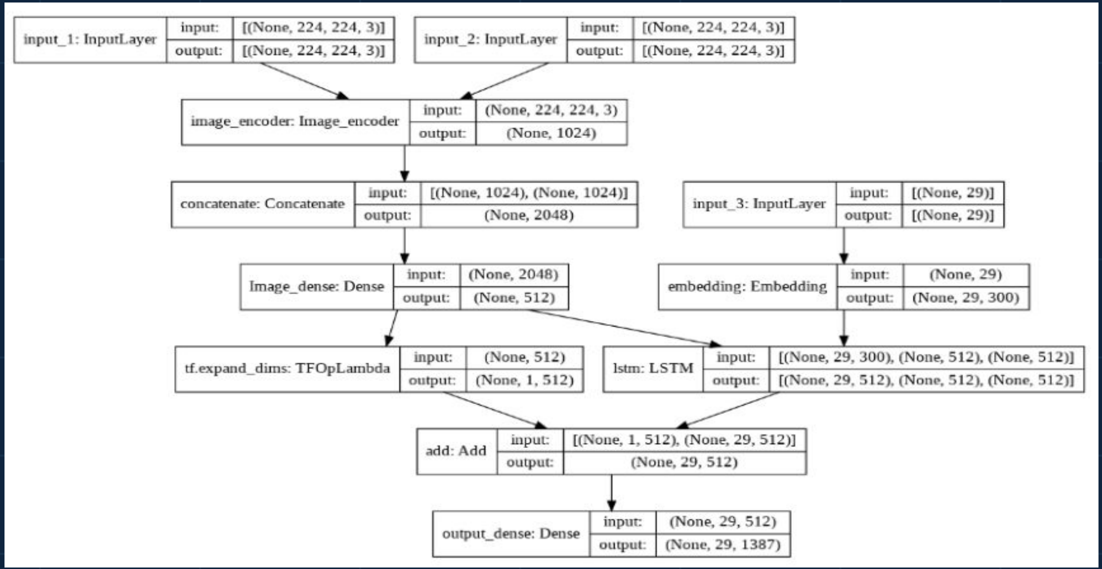
CHEXNET

- Dense-121 architecture pretrained on thousands of chest x-ray images to detect 14 diseases.



Simple Encoder Decoder

- It’s a simple implementation of an image captioning model and was used as our baseline model.



input_1: InputLayer | input: [(None, 224, 224, 3)] | output: [(None, 224, 224, 3)]

input_2: InputLayer | input: [(None, 224, 224, 3)] | output: [(None, 224, 224, 3)]

image_encoder: Image_encoder | input: (None, 224, 224, 3) | output: (None, 1024)

concatenate: Concatenate | input: [(None, 1024), (None, 1024)] | output: (None, 2048)

input_3: InputLayer | input: [(None, 29)] | output: [(None, 29)]

Image_dense: Dense | input: (None, 2048) | output: (None, 512)

embedding: Embedding | input: (None, 29) | output: (None, 29, 300)

tf.expand_dims: TFOpLambda | input: (None, 512) | output: (None, 1, 512)

lstm: LSTM | input: [(None, 29, 300), (None, 512), (None, 512)] | output: [(None, 29, 512), (None, 512), (None, 512)]

add: Add | input: [(None, 1, 512), (None, 29, 512)] | output: (None, 29, 512)

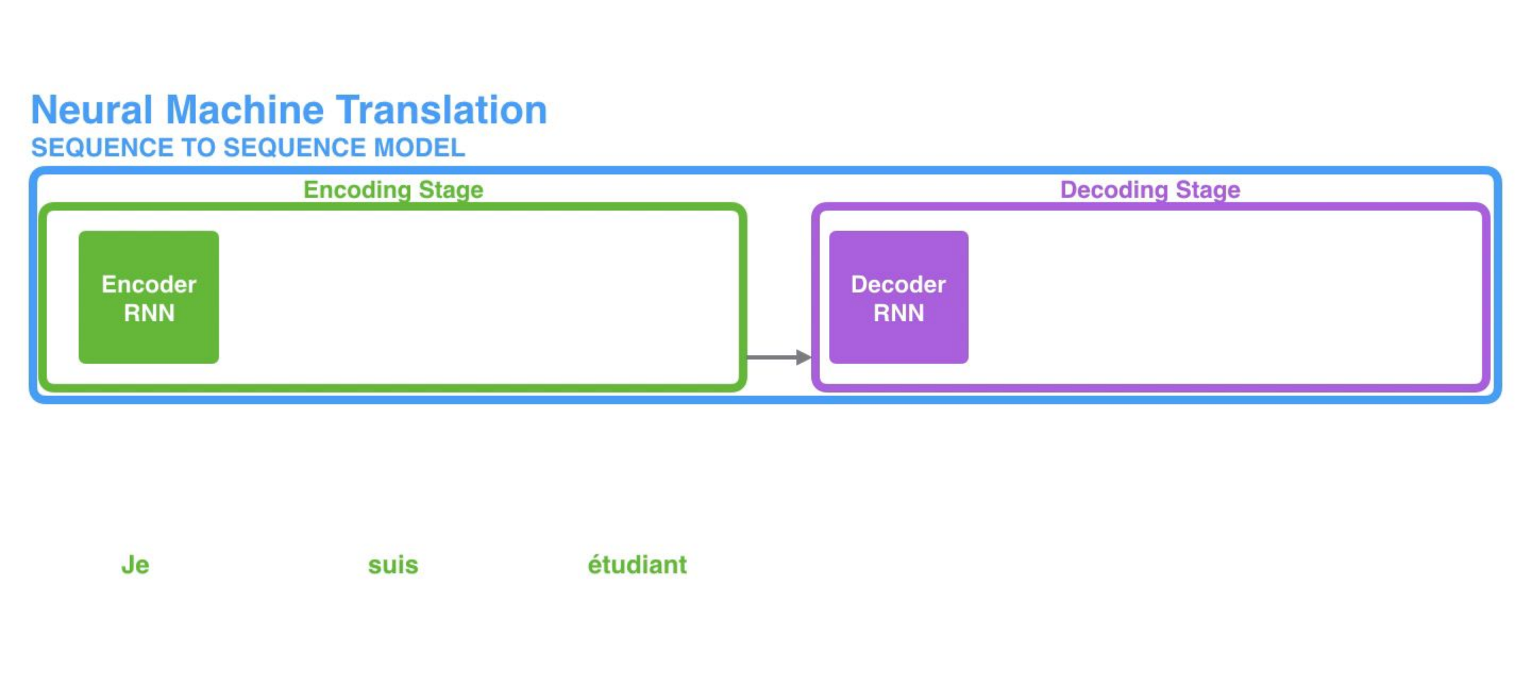
output_dense: Dense | input: (None, 29, 512) | output: (None, 29, 1387)

Attention Decoder Model

- The encoder part was CHEXNET, same as the simple encoder decoder architecture.
- For the decoder part, we adopted the global attention mechanism to utilize the most relevant parts of the input sequence in a flexible manner, by a weighted combination of all of the encoded input vectors, with the most relevant vectors being attributed the highest weights.

10cm

Bottle Neck Problem



Neural Machine Translation

SEQUENCE TO SEQUENCE MODEL

Encoding Stage

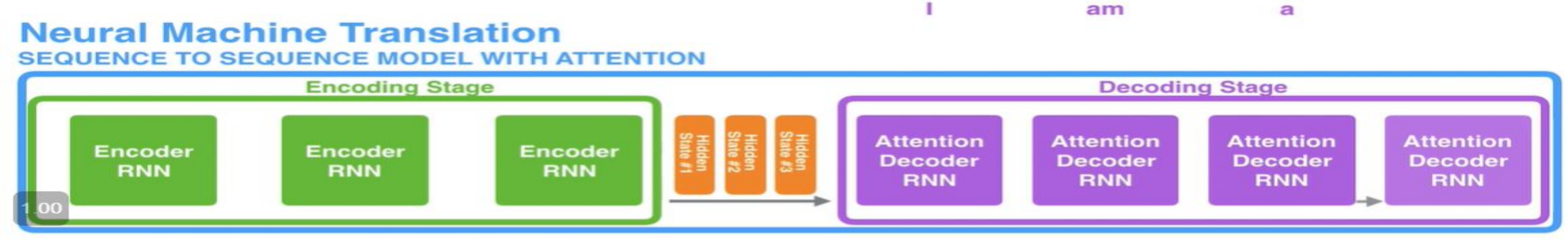
Encoder RNN

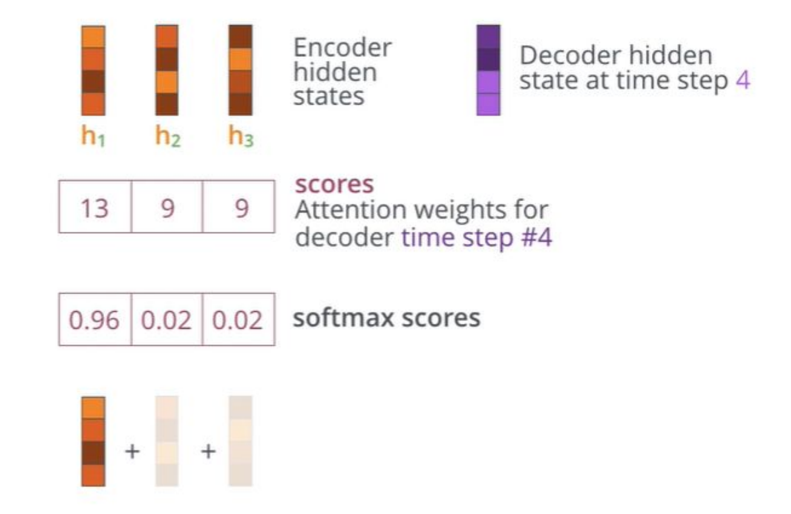
Decoding Stage

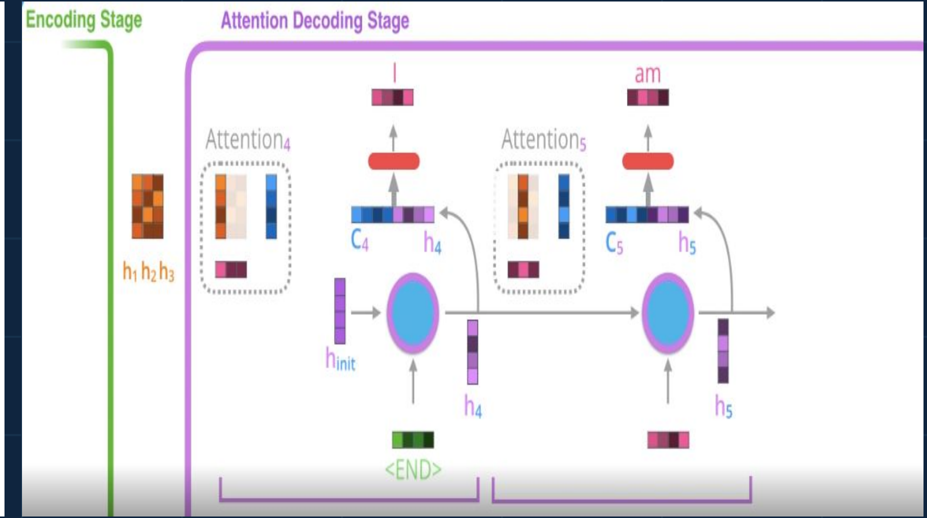
Decoder RNN

Je suis étudiant

Attention

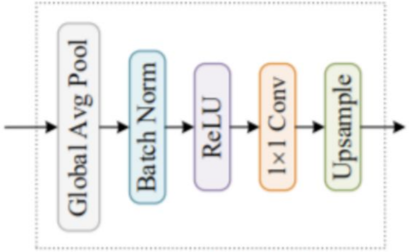

Neural Machine Translation
SEQUENCE TO SEQUENCE MODEL WITH ATTENTION
I
am
a
Encoding Stage
Decoding Stage
Encoder RNN
Encoder RNN
Encoder RNN
Hidden State #1
Hidden State #2
Hidden State #3
Attention Decoder RNN
Attention Decoder RNN
Attention Decoder RNN
Attention Decoder RNN
1.00


h1
h2
h3
Encoder hidden states
Decoder hidden state at time step 4
13
9
9
scores
Attention weights for decoder time step #4
0.96
0.02
0.02
softmax scores
+
+

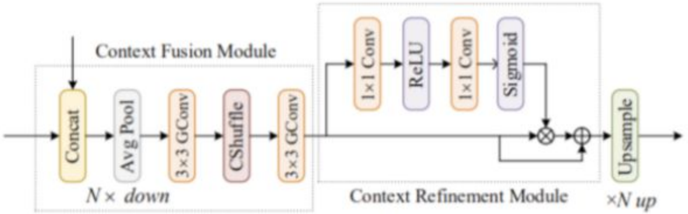

Encoding Stage
Attention Decoding Stage
h1 h2 h3
Attention4
Attention5
I
am
C4
h4
C5
h5
hinit
h4
h5
<END>

CAM Encoder Model

- For the encoder part, the backbone features from the CHEXNET model, specifically 3rd last layer’s output, was passed through global flow and context flow which is actually inspired from another model which was used for image segmentation purposes.
- Global flow extracts the global information of images while context flow extracts the local features of the images.
- The decoder in this model uses attention, same as the previous attention model.



a) Global Flow



c) Context Flow

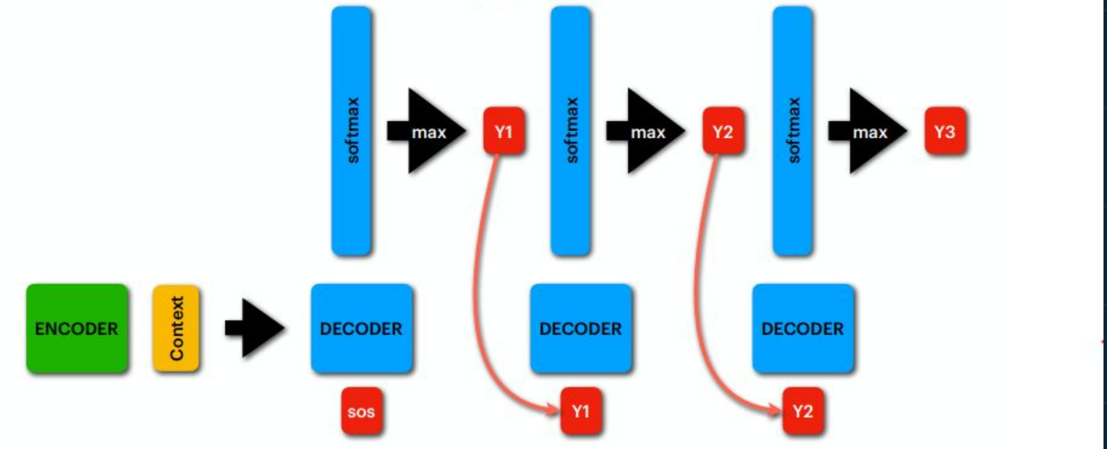
04

Inference and Evaluation

10cm

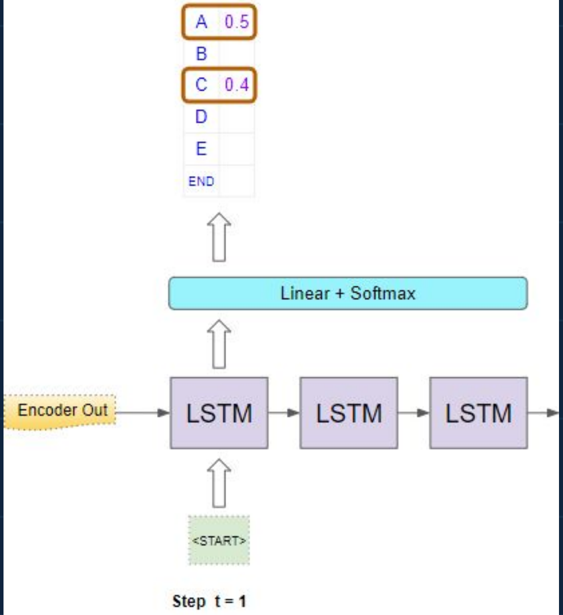
Greedy Search

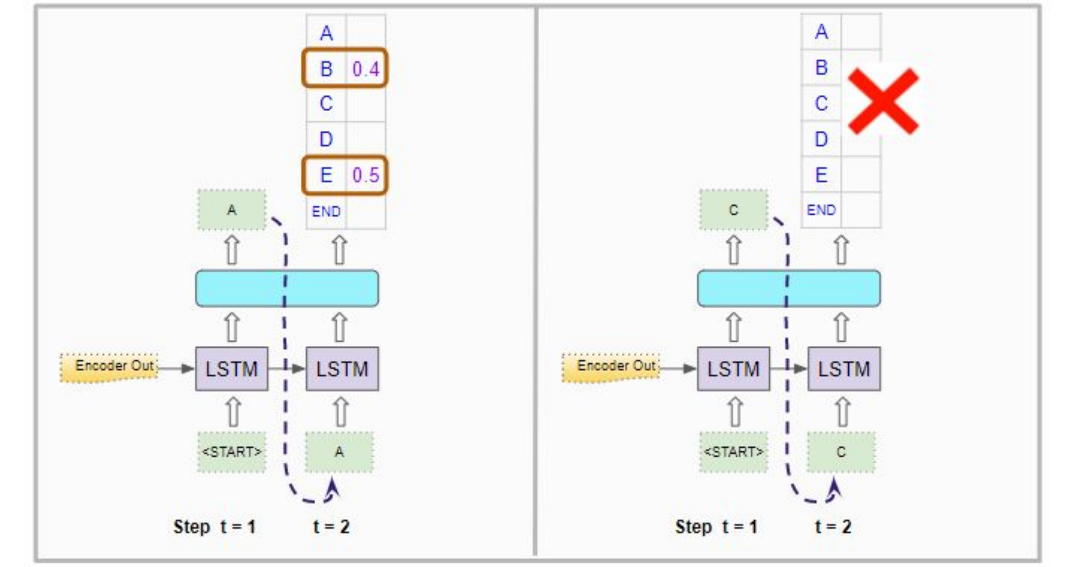
- There are two approaches used when generating the predicted text.
- The Greedy Search method chooses the word with the highest probability for that time step, and uses that word as input for the next time step.
-



Beam Search

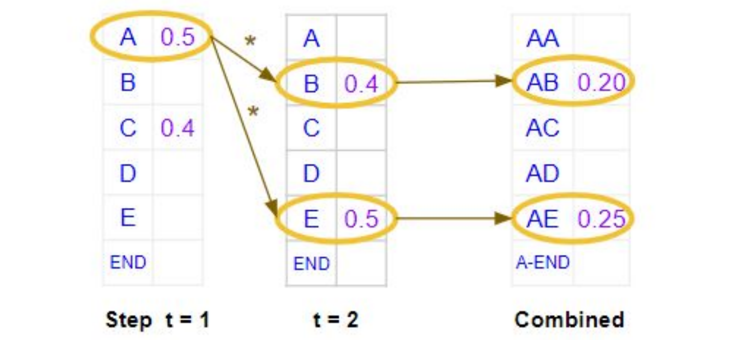
- The Beam Search method tries to search multiple paths by searching for multiple words and choosing the best overall sentence instead of finding the best word in that time step.





Greedy and Beam Search

- The Beam Search calculates the conditional probability of the word being in that position based on the previous input.

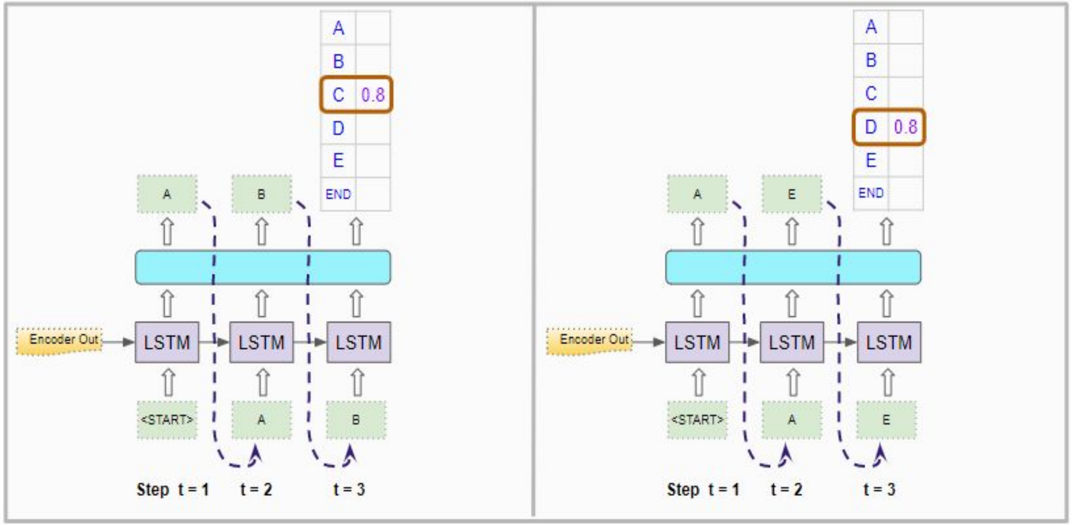


$$\text{Prob (AB | input)} = \text{Prob (A | input)} * \text{Prob (B | A, input)}$$

$$\text{Prob (AB)} = \text{Prob (A)} * \text{Prob (B | A)}$$

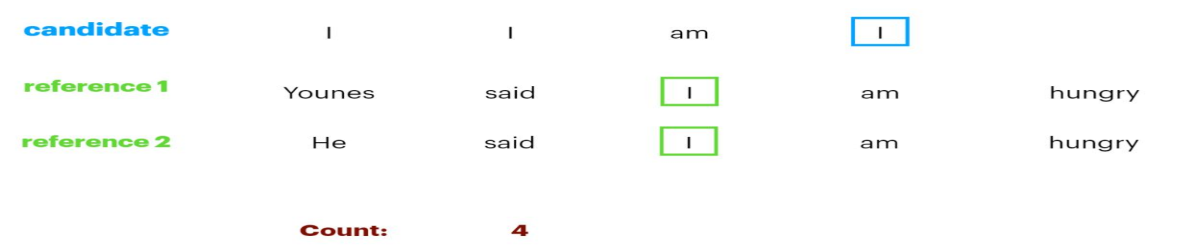
$$= 0.5 * 0.4$$

$$= 0.20$$

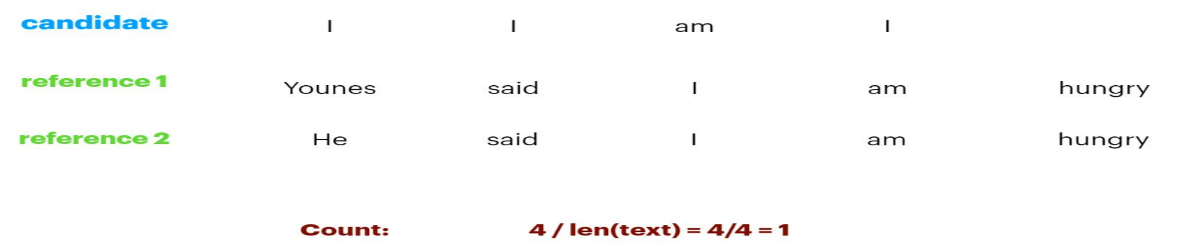


BLEU Score

- Compares each word in the predicted sentence and compare it to the reference sentence and returns score based on how many words were predicted that were in the original sentence.



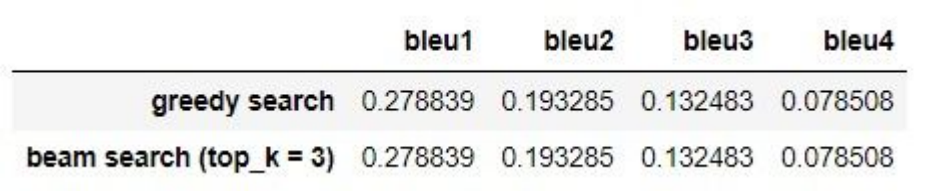
candidate	I	I	am	I	
reference 1	Younes	said	I	am	hungry
reference 2	He	said	I	am	hungry
	Count:	4			



candidate	I	I	am	I	
reference 1	Younes	said	I	am	hungry
reference 2	He	said	I	am	hungry
	Count:	4 / len(text) = 4/4 = 1			

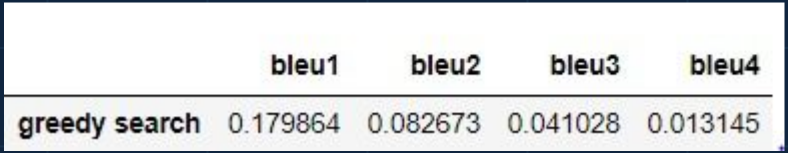
Model Scores

Baseline Encoder Decoder:



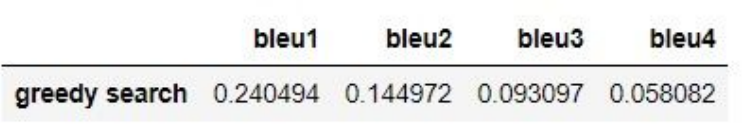
	bleu1	bleu2	bleu3	bleu4
greedy search	0.278839	0.193285	0.132483	0.078508
beam search (top_k = 3)	0.278839	0.193285	0.132483	0.078508

Attention Model:



	bleu1	bleu2	bleu3	bleu4
greedy search	0.179864	0.082673	0.041028	0.013145

CAM Model:



	bleu1	bleu2	bleu3	bleu4
greedy search	0.240494	0.144972	0.093097	0.058082

Predictions vs Labels

- Insert examples of predictions and labels, one wrong, one different, one similar.

10cm

05

Further Experimentations

10cm

Why transformer?

- Bottleneck when encoder tries to fit large amount of data.
- Parallelization.
- Sequence detection.

10cm

- Attention is the core block.
- Positional Encoding.

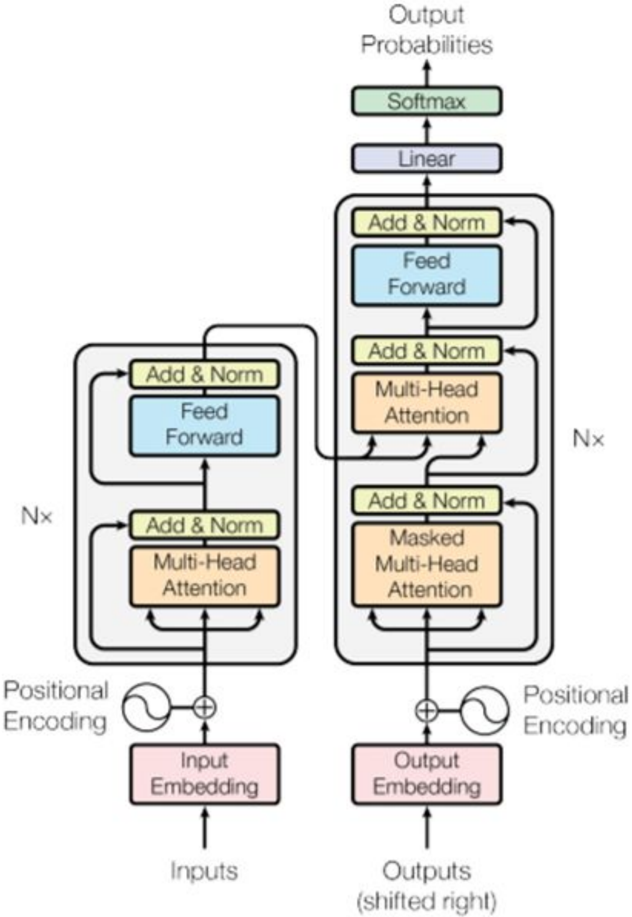
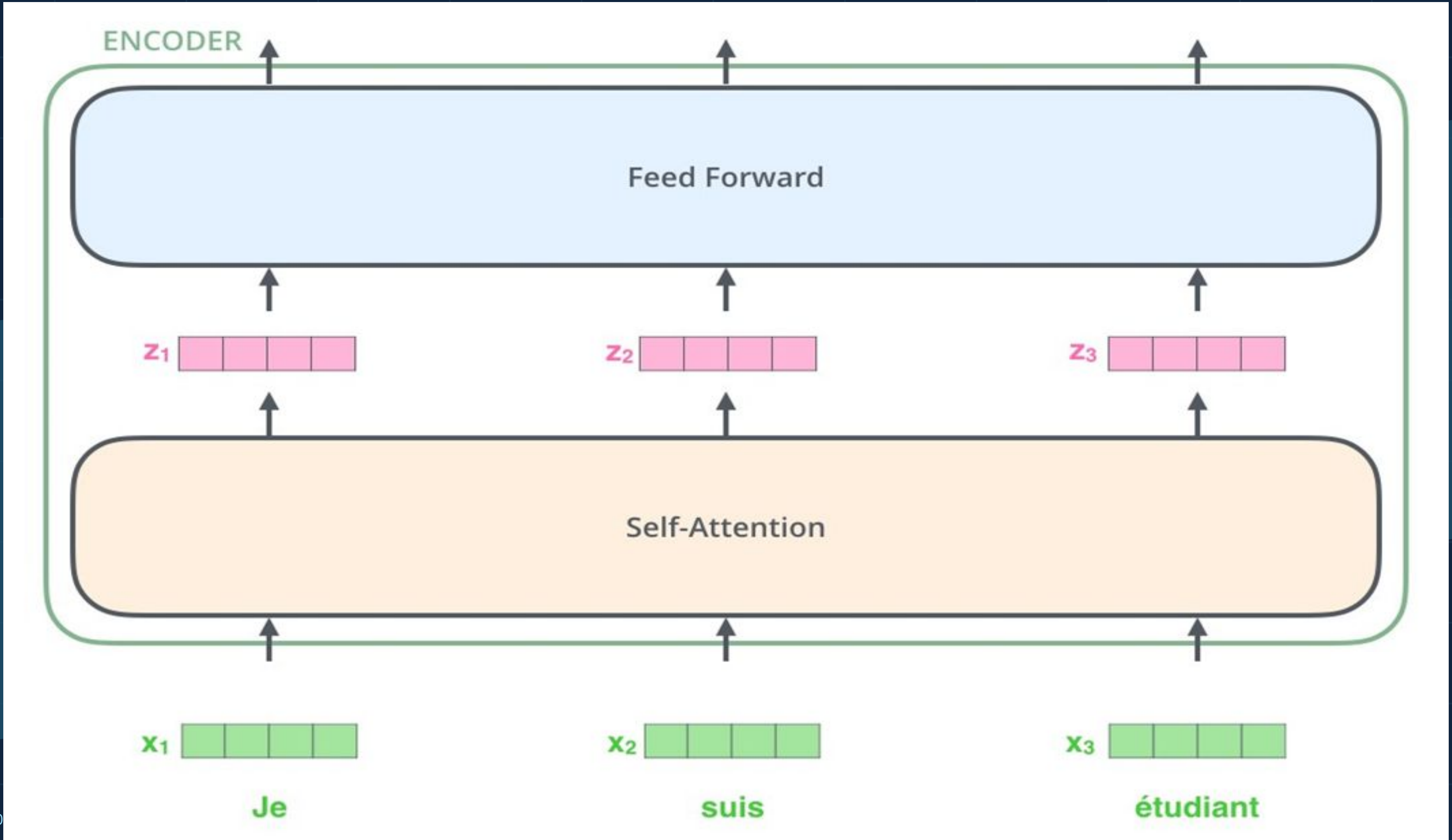
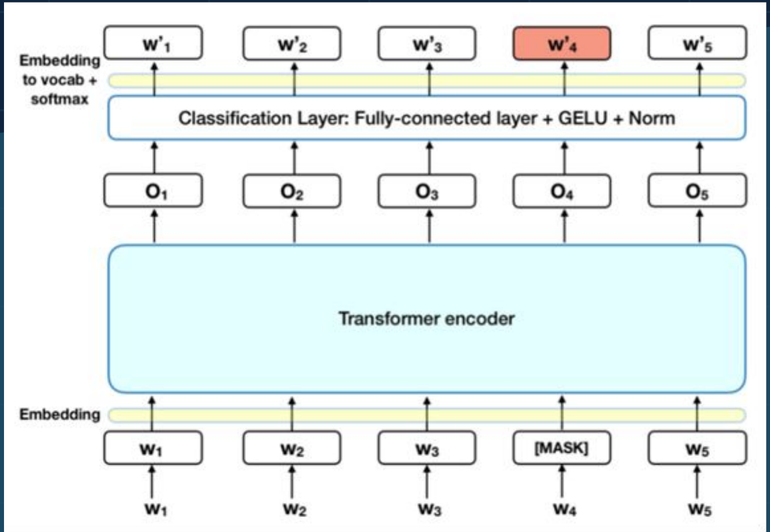


Figure 1: The Transformer - model architecture.

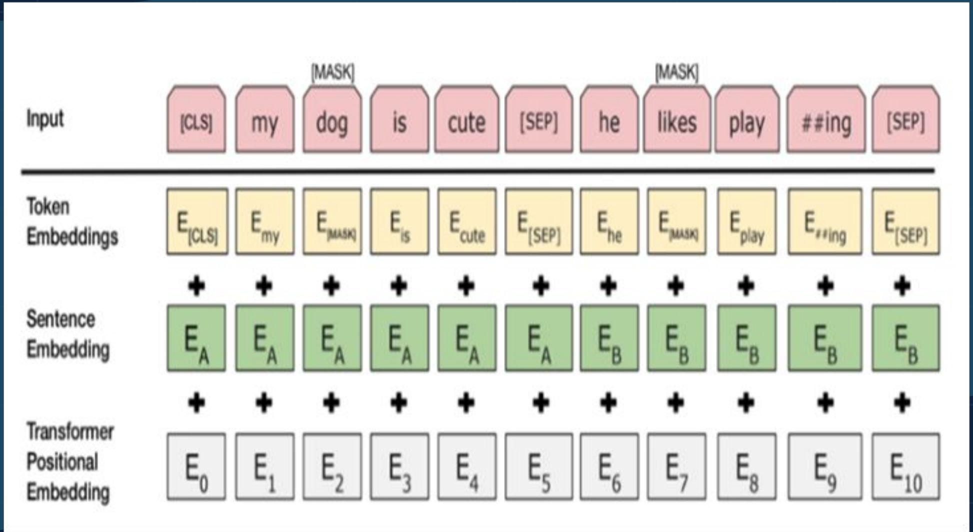
10cm


ENCODER
Feed Forward
z_1
z_2
z_3
Self-Attention
x_1
x_2
x_3
Je
suis
étudiant

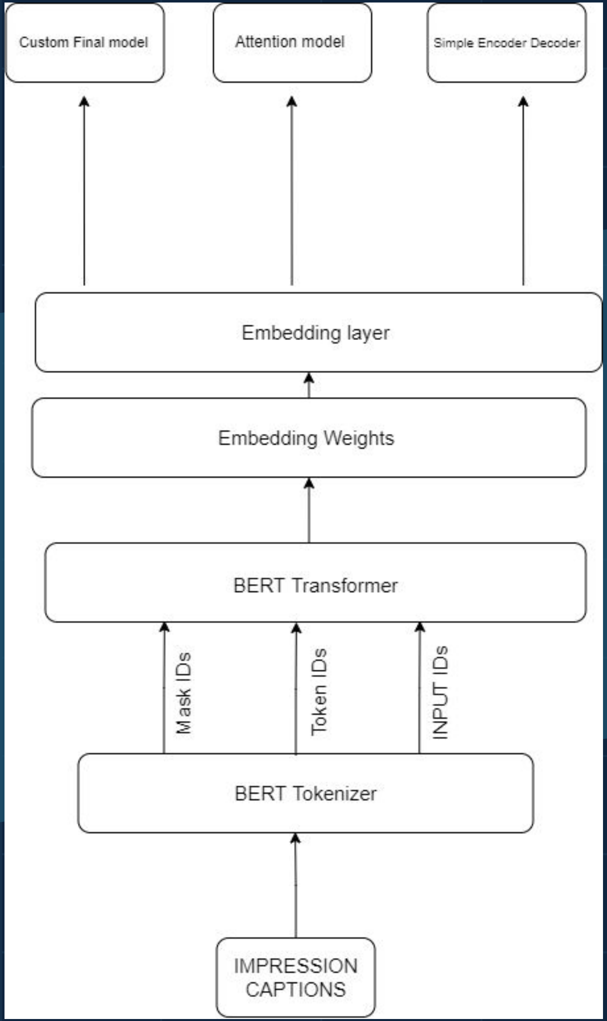
BERT for Word Embeddings


w'₁
w'₂
w'₃
w'₄
w'₅
Embedding to vocab + softmax
Classification Layer: Fully-connected layer + GELU + Norm
O₁
O₂
O₃
O₄
O₅
Transformer encoder
Embedding
w₁
w₂
w₃
[MASK]
w₅
w₁
w₂
w₃
w₄
w₅

Masked Language Modeling

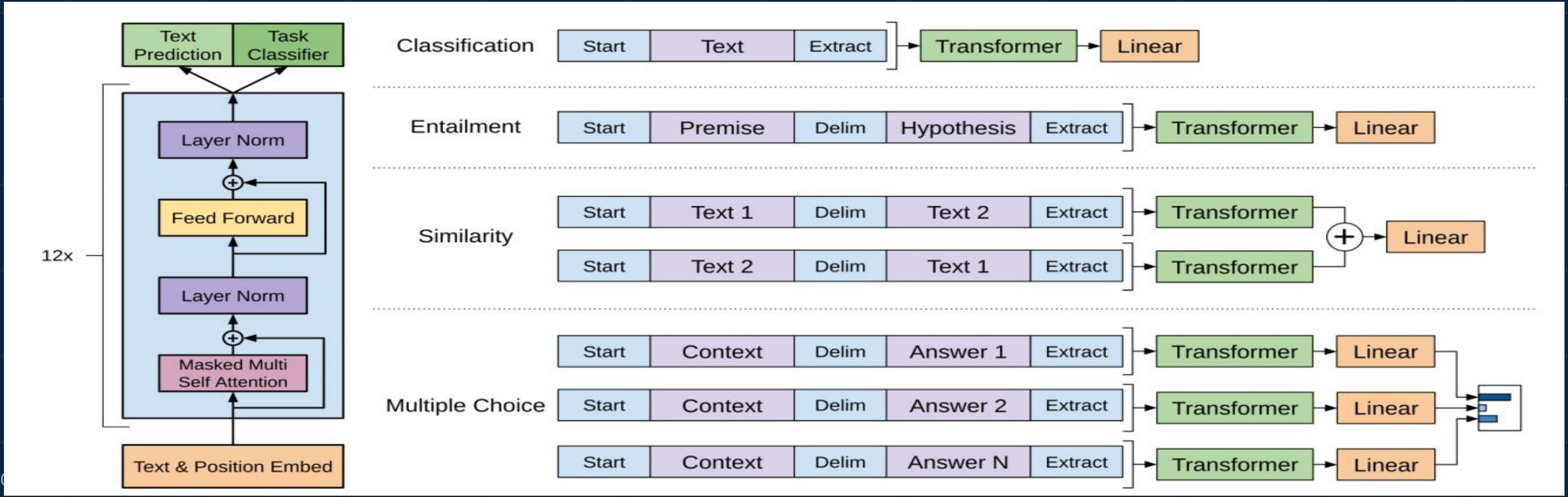


Next Sentence prediction


Custom Final model
Attention model
Simple Encoder Decoder
Embedding layer
Embedding Weights
BERT Transformer
Mask IDs
Token IDs
INPUT IDs
BERT Tokenizer
IMPRESSION CAPTIONS

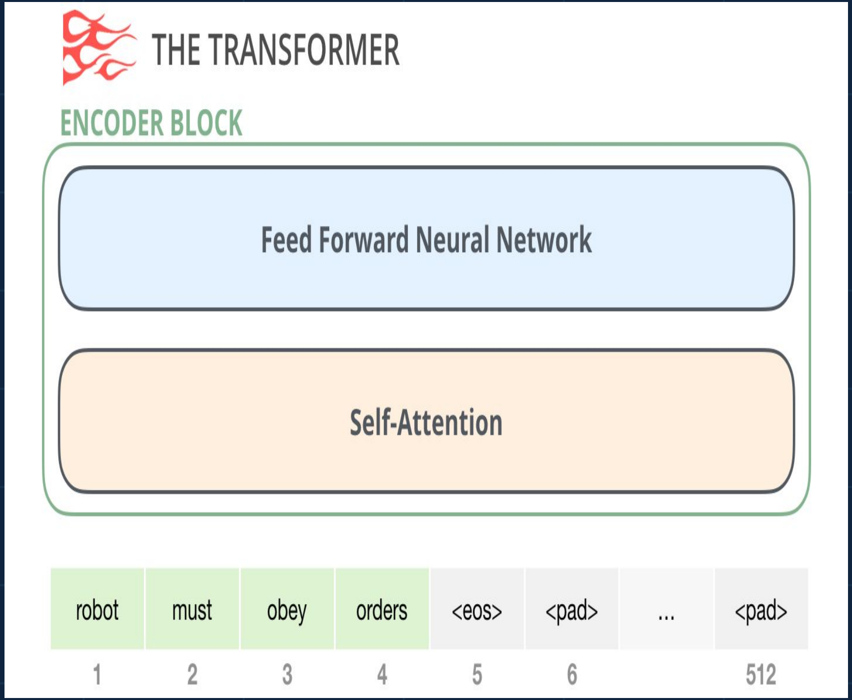
GPT

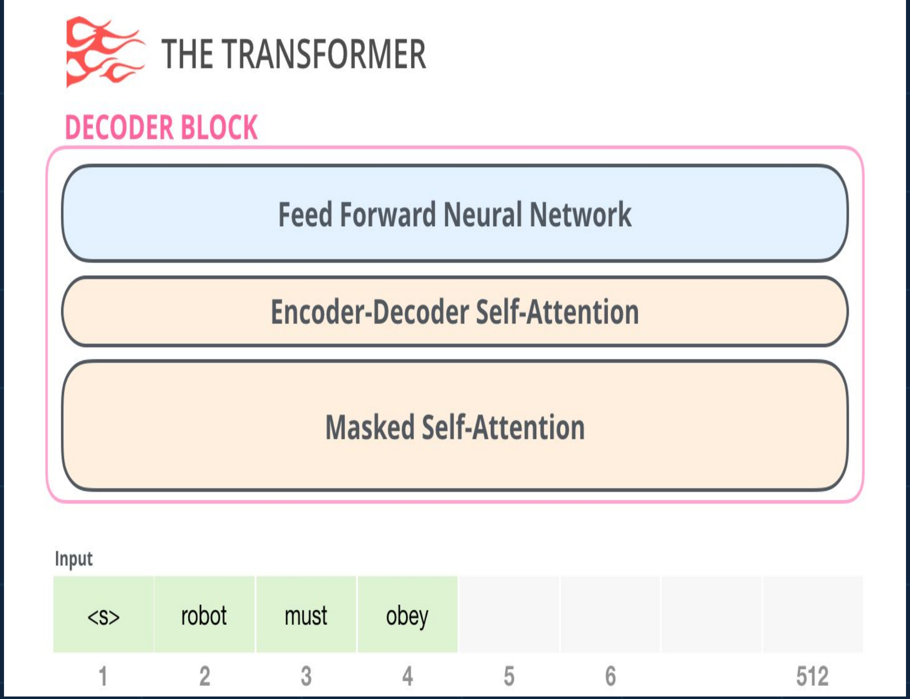
- The GPT model is developed by OpenAI.
- GPT is a 12 layer, 12 attention head, transformer decoder, but explore how to take advantage of massive unlabeled text datasets to fine-tune them on limited supervised datasets.
- GPT can work on 12 supervised learning tasks.



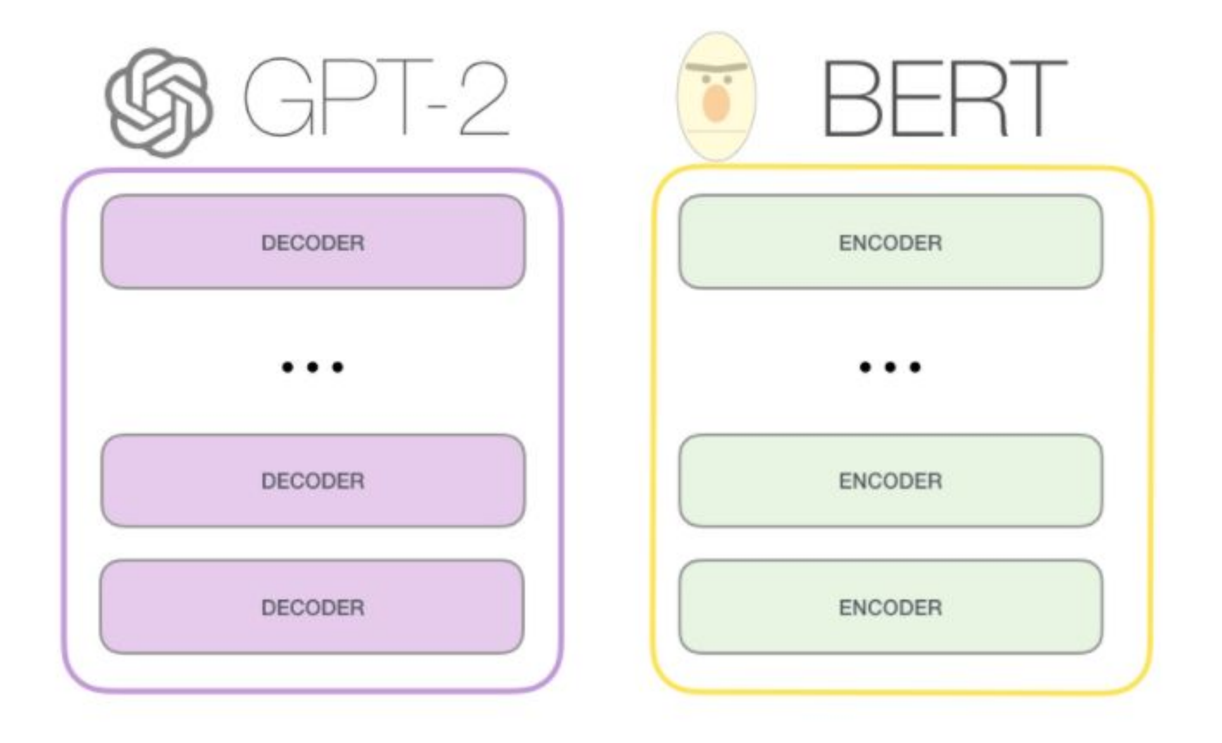
GPT vs BERT

GPT is built out of Decoders only, BERT is built out of Encoders only.





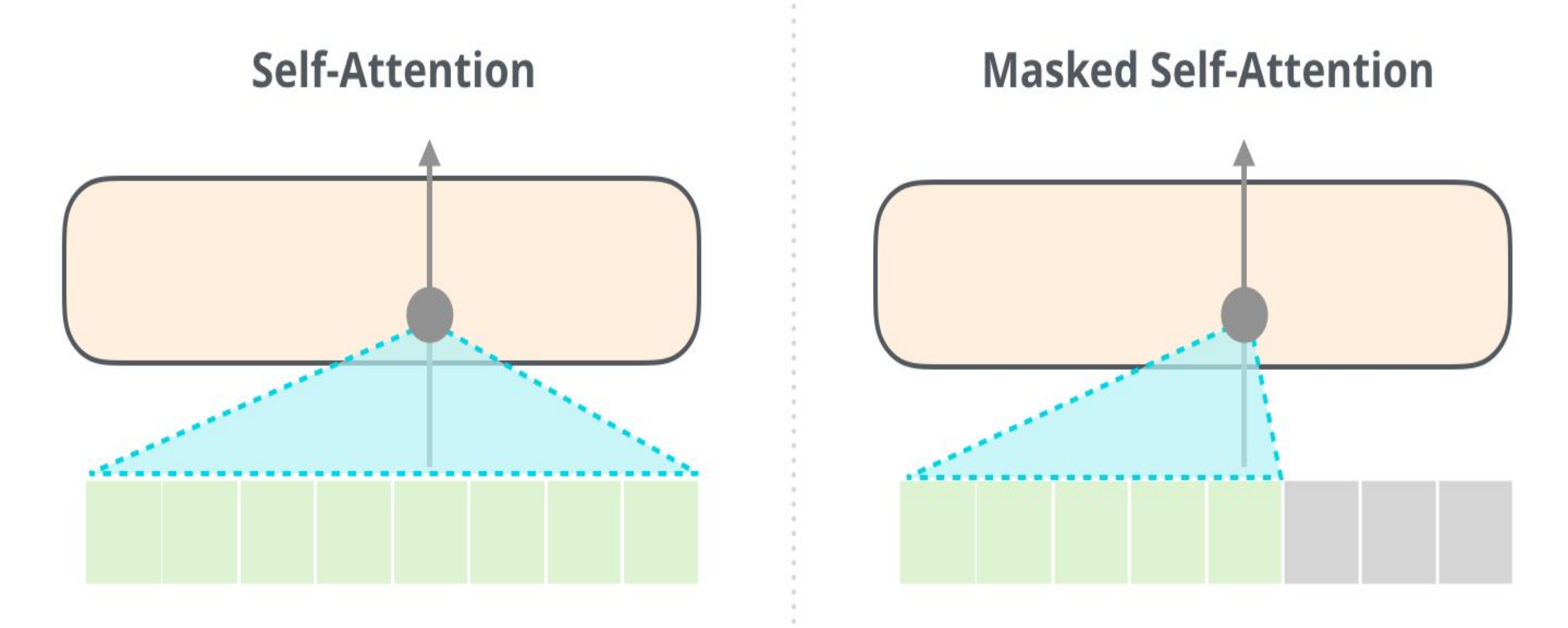
GPT vs BERT


GPT-2
BERT
DECODER
...
DECODER
DECODER
ENCODER
...
ENCODER
ENCODER

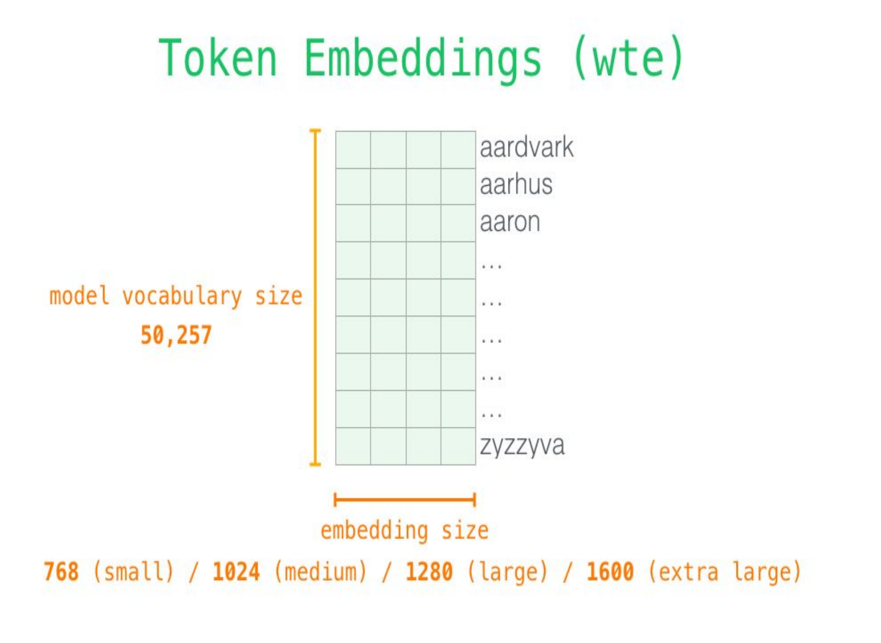
10cm

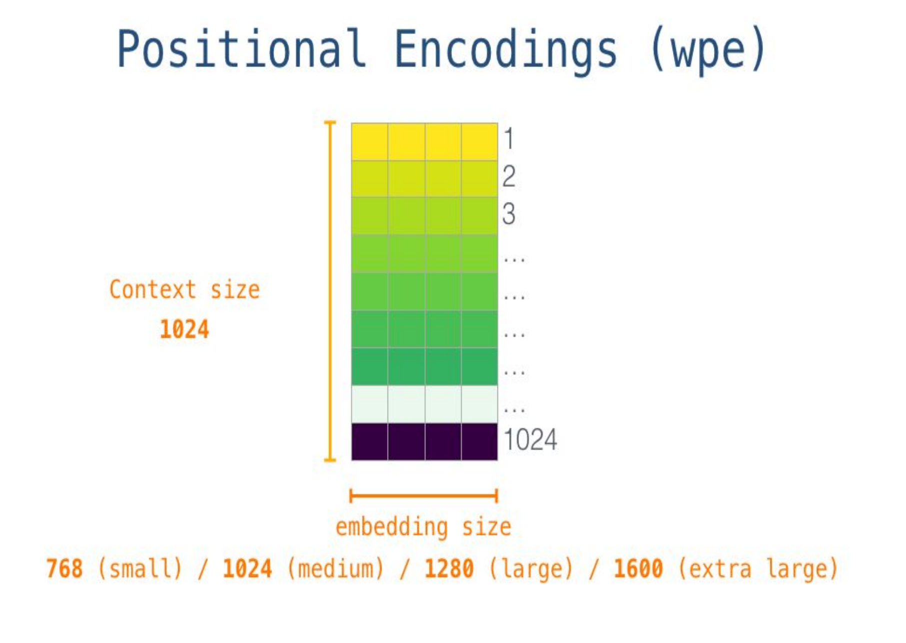
GPT vs BERT

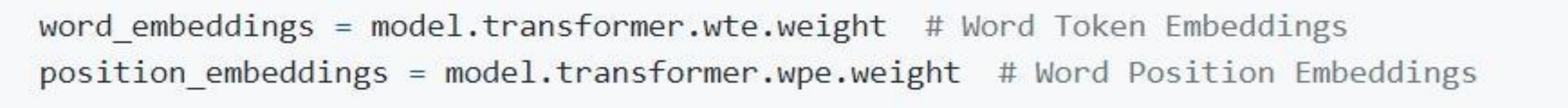
The GPT2, and some later models are auto-regressive in nature. BERT is not.
That is a trade off. In losing auto-regression, BERT gained the ability to incorporate the context on both sides of a word to gain better results. XLNet brings back autoregression while finding an alternative way to incorporate the context on both sides.



GPT As Embedding Layer


Token Embeddings (wte)


Positional Encodings (wpe)
1
2
3
...
...
...
...
...
1024
Context size
1024
embedding size
768 (small) / 1024 (medium) / 1280 (large) / 1600 (extra large)



```
word_embeddings = model.transformer.wte.weight  # Word Token Embeddings
position_embeddings = model.transformer.wpe.weight  # Word Position Embeddings
```

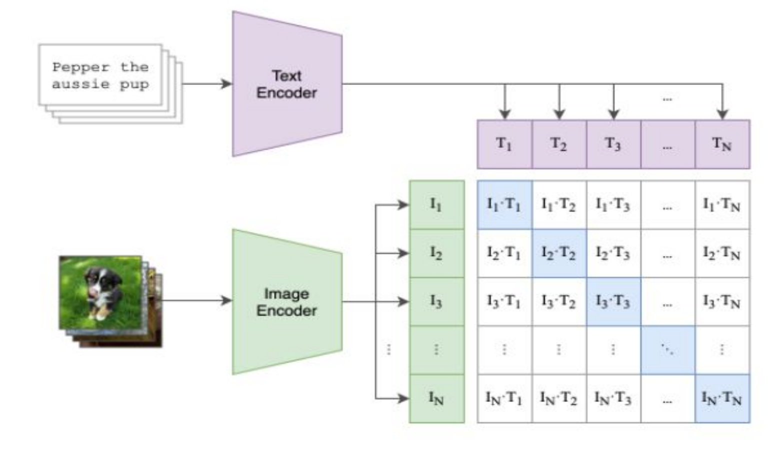
06

Future Works

10cm

Future Work

- Using more advanced models. (BART - GPT3 - Visual Bert)
- Fine-tune chex-net weights using “CheXpert”, which is a dataset containing 200K images.
- Collecting more data.
- Add class weights during training, to solve the imbalance.
- Use Simple decoder with CAM model.
- CLIP: Contrastive language-image Pre-Training:
- Mobile Application.

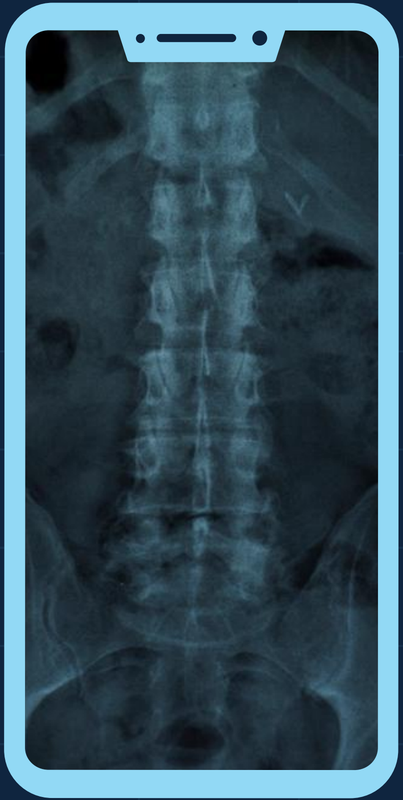


Pepper the aussie pup
Text Encoder
Image Encoder
T1 T2 T3 ... TN
I1 I2 I3 ⋮ IN
I1·T1 I1·T2 I1·T3 ... I1·TN
I2·T1 I2·T2 I2·T3 ... I2·TN
I3·T1 I3·T2 I3·T3 ... I3·TN
IN·T1 IN·T2 IN·T3 ... IN·TN

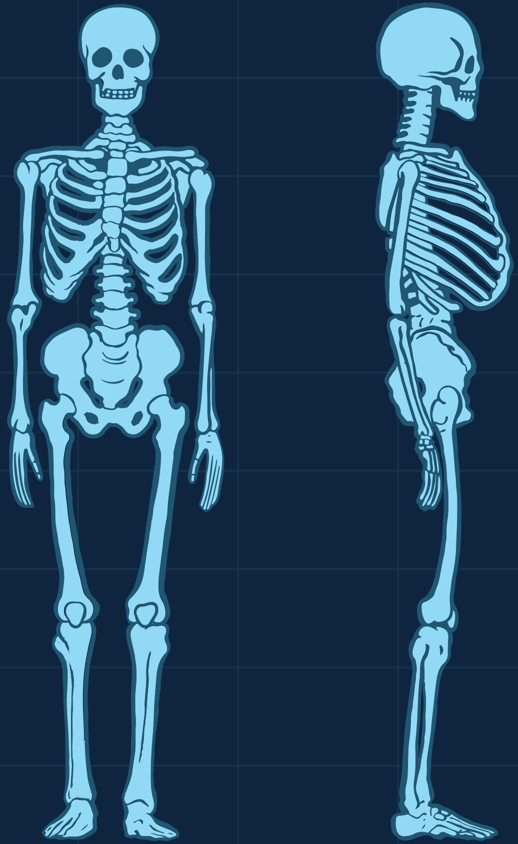
10cm

Mobile app

You can replace the image on the screen with your own work. Right-click on it and then choose "Replace image" (in Google Slides) or "Change Picture" (in PPT) so you can add yours



10cm



10cm