# *Chest Diagnosis Tool*

*Introduced by:*

*1- Mariam Khaled*
*2- Abdelrahman Tarek*
*3- Hassan Mohamed*
*4- Abdullah Shawwaf*
*5- Mohamed Magdy*
*6- Sarah Abdelmaoty*

# Table of Contents

# Introduction

## Executive Summary

This project is concerned with medical image captioning on chest X-Rays via creating an image captioning deep learning model which can write automatic medical reports.

## Business Problem

The goal is to predict the impressions of the medical report attached to the images. The process of writing medical reports usually takes around 5–10 minutes per report. In a day the doctors have to write medical reports that number in 100s which can take a lot of their time. 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.

## Abbreviations and Terminologies

| Terminology | Definition |
|---|---|
| CAM | Chained Context Aggregation Module used mainly for image segmentation. |
| BERT | Bidirectional Encoder Representations from Transformers (BERT) is a transformer-based machine learning technique for natural language processing (NLP) pre-training developed by Google. |
| GPT | Generative Pre-trained Transformer is an autoregressive language model that uses deep learning to produce human-like text. There are GPT-2 and GPT-3. |

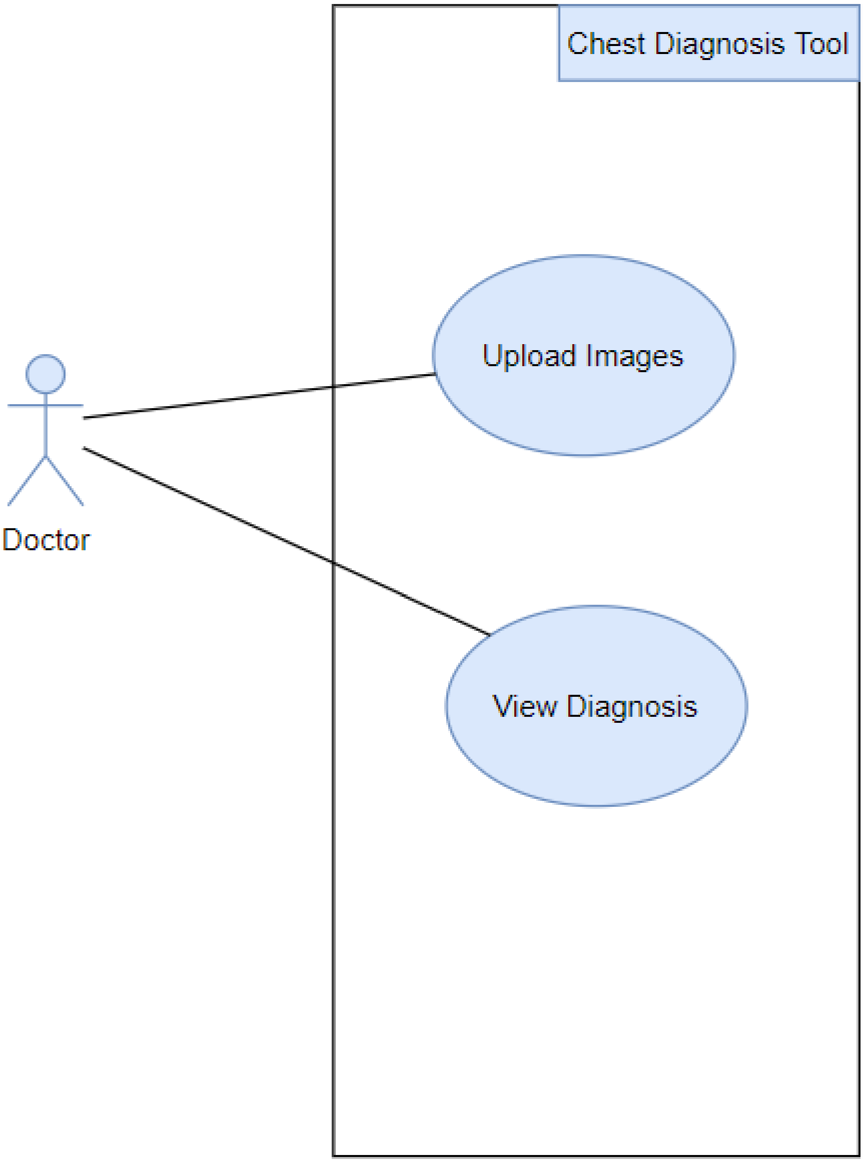
## References

- towardsdatascience.com/image-captioning-with-keras-teaching-computers-to-describe-pictures-c88a46a311b8
- jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/
- towardsdatascience.com/working-with-hugging-face-transformers-and-tf-2-0-89bf35e3555a
- arxiv.org/abs/2002.12041v3
- tensorflow.org/text/tutorials/nmt_with_attention
- keras.io/guides/transfer_learning/
- tensorflow.org/guide/keras/custom_layers_and_models
- arxiv.org/pdf/1706.03762.pdf
- towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270
- towardsdatascience.com/gpt-3-a-complete-overview-190232eb25fd

# System Users

## Doctors

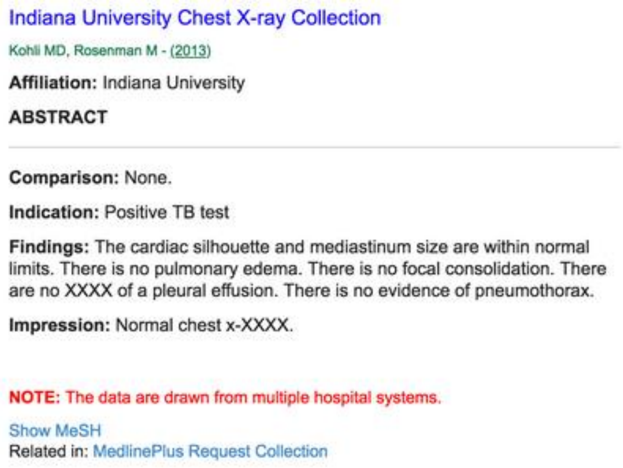
They are our main users. It can help them reduce the time they spend daily examining the X-rays and writing the medical reports.



# Dataset

## Data Source

We based our work on a publicly available dataset from Indiana University which consists of chest X-ray images and reports in XML format which includes information regarding the comparison, indication, findings, and impression of the X-ray images.



Indiana University Chest X-ray Collection

Kohli MD, Rosenman M - (2013)

**Affiliation:** Indiana University

**ABSTRACT**

**Comparison:** None.

**Indication:** Positive TB test

**Findings:** The cardiac silhouette and mediastinum size are within normal limits. There is no pulmonary edema. There is no focal consolidation. There are no XXXX of a pleural effusion. There is no evidence of pneumothorax.

**Impression:** Normal chest x-XXXX.

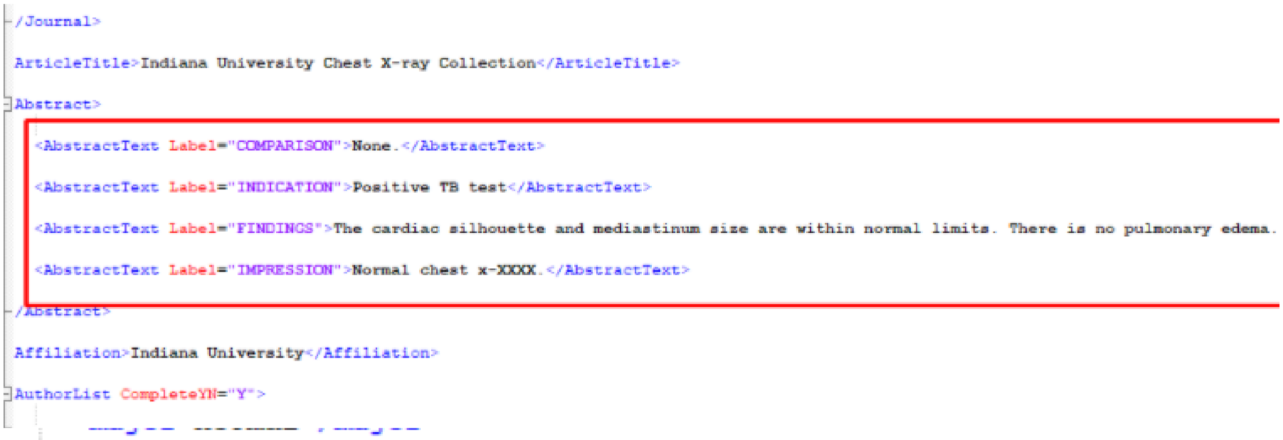
**NOTE:** The data are drawn from multiple hospital systems.

Show MeSH

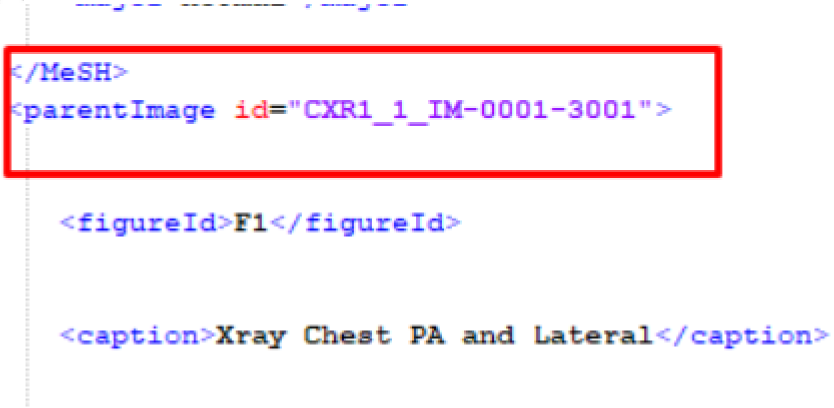
Related in: MedlinePlus Request Collection

# Data Preprocessing

Each XML report has the following tags: comparison, indication, findings, impression and parentImage. 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.

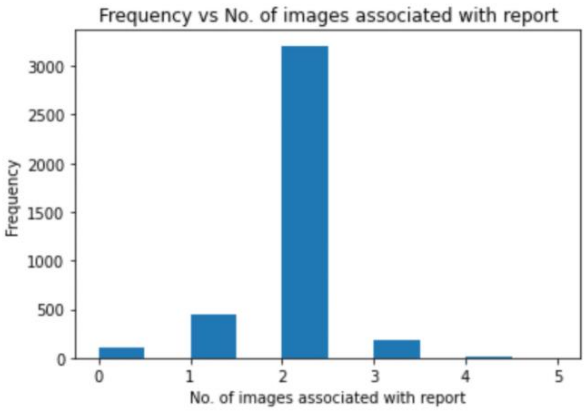


```
/Journal>
ArticleTitle>Indiana University Chest X-ray Collection</ArticleTitle>
Abstract>
  <AbstractText Label="COMPARISON">None.</AbstractText>
  <AbstractText Label="INDICATION">Positive TB test</AbstractText>
  <AbstractText Label="FINDINGS">The cardiac silhouette and mediastinum size are within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

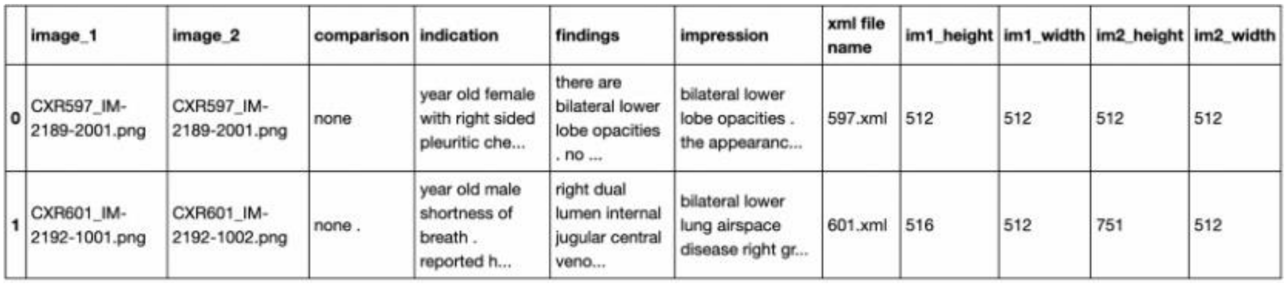


```
</MeSH>
<parentImage id="CXR1_1_IM-0001-3001">

  <figureId>F1</figureId>

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

We plotted a histogram to find the minimum and maximum number of images associated with each report:



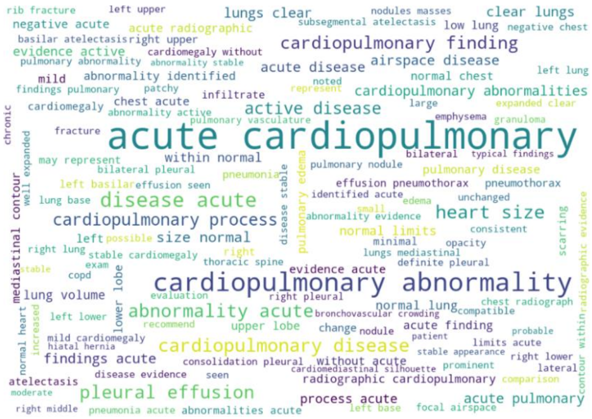
Since it was found that two images was the most frequent case, we took two images as input. We also extracted the information part of the comparison, indication, findings and impression tags of the report using regex. For images more than two, we will create new datapoints with new image and same info. The extracted dataframe was as follows:

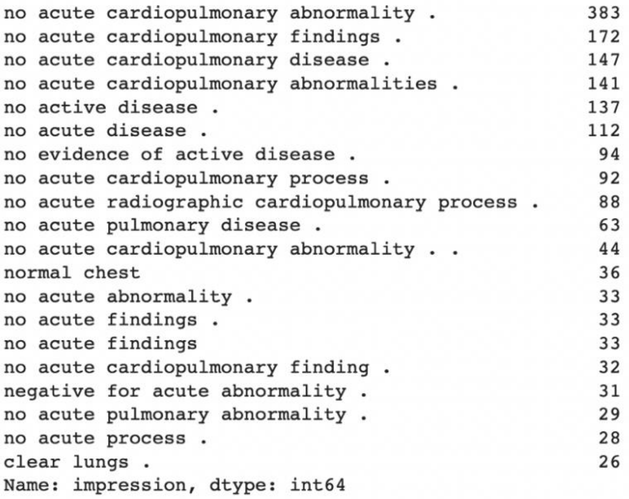


| | image_1 | image_2 | comparison | indication | findings | impression | xml file name | im1_height | im1_width | im2_height | im2_width |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | CXR597_IM-2189-2001.png | CXR597_IM-2189-2001.png | none | year old female with right sided pleuritic che... | 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 dataframe, all the datapoints which had image_1 and impression value as null were removed from the dataframe, then all the missing values found in image_2 were filled with the same data path of that of image_1. Since the pretrained models are using square-sized images we chose 224*224 (same as that of input to VGG16) as the specified size of the images.

We also created a word cloud for the impressions values to examine the distribution of the impressions, which was the following:





```
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
```

From the above value counts we can see that the top 20 most frequently occurring phrases had the same meaning thereby the same information suggesting one type of impression is dominating this data. So we applied a set of upsampling and downsampling to the data so that the model doesn't overfit.

# Models

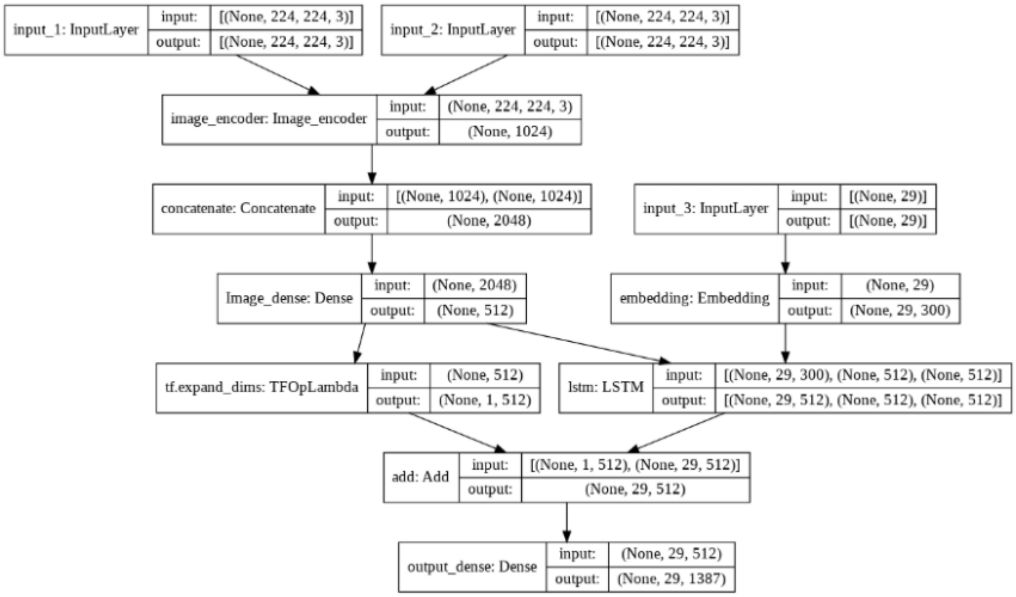
For the modelling part we've created three main models. Each one had a similar encoder but a different decoder architecture.

## Encoder

We used the CHEXNET pretrained model to extract backbone features from the two images used as input to feed it to the decoder. The CHEXNET model is a DenseNet121 layered model which is trained on millions of chest x-ray images for the classification of 14 diseases.

## Simple Encoder Decoder

This model was our baseline model. It's a simple implementation of an image captioning model. The architecture is shown as follows:


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)

The two images pass through the Image_encoder layer "CHEXNET" and concatenate the two outputs and then pass it through the Dense layer. The padded tokenized captions will be passed through an embedding layer where we will be using pretrained Glove vectors (300 dimensions) as the initial weights for the layer. This will be set as trainable and then it is passed through LSTM where the initial state of the LSTM is taken from the output of Image_dense layer. These are then added and then passed through output_dense layer where the numbe of output will be the vocabulary size with softmax activation applied on top.

## Attention Decoder Model

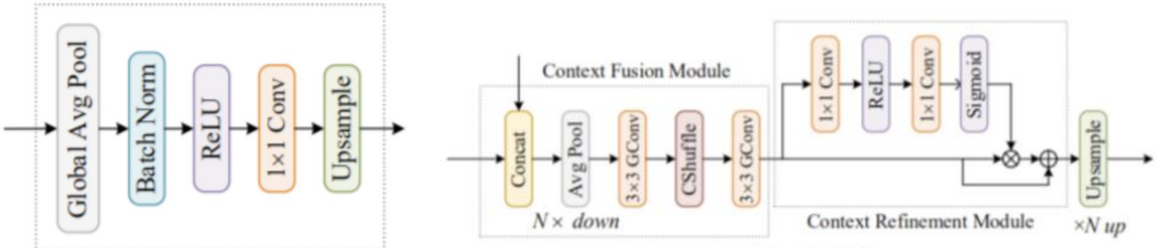
The encoder part was CHEXNET, same as the simple encoder decoder architecture. For the decoder part, we adopted the global attention mechanism:

$$\mathrm{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \begin{cases} \boldsymbol{h}_t^\top \bar{\boldsymbol{h}}_s & \textit{dot} \\ \boldsymbol{h}_t^\top \boldsymbol{W}_a \bar{\boldsymbol{h}}_s & \textit{general} \\ \boldsymbol{v}_a^\top \tanh\left(\boldsymbol{W}_a [\boldsymbol{h}_t; \bar{\boldsymbol{h}}_s]\right) & \textit{concat} \end{cases}$$

The attention mechanism permits the decoder 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.

## CAM Encoder Model

For the encoder part for this model, 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 structure of the global flow and the context flow blocks is as follows:



a) Global Flow

c) Context Flow

The outputs from encoder will be sent to global flow, then outputs from both the chexnet and global flow will be concatenated and sent to context flow. After that, the output from the global flow and context flow will be summed and then sent to decoder after reshaping, and applying batch norm and dropout. The decoder in this model uses attention, same as the previous attention model.

# Embedding Techniques

## GPT for Word Embeddings

Since transformers have offered state-of-the-art solutions recently in many NLP problems, we decided to use GPT-2 transformer to extract word embeddings instead of Glove which we used previously in our three main architectures.

## BERT for Word Embeddings

We decided to try Bert, which is another type of transformers to extract word embeddings.

# Inference Methods

We used two inference approaches: greedy search and beam search with each model we used, to compare the results and find out which approach best works out.

## Greedy Search

A simple approximation that selects the most likely word at each step in the output sequence. This approach has the benefit that it is very fast, but the quality of the final output sequences may be far from optimal.

## Beam Search

The beam search algorithm selects multiple alternatives for an input sequence at each timestep based on conditional probability. The number of multiple alternatives depends on a parameter called Beam Width B. At each time step, the beam search selects B number of best alternatives with the highest probability as the most likely possible choices for the time step.

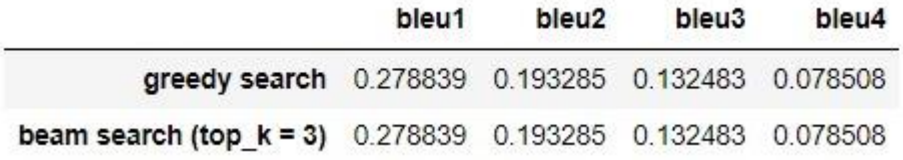
# Evaluation Metrics

## BLEU score

We used BLEU score to evaluate the performance of the models and the quality of the caption generated. 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.

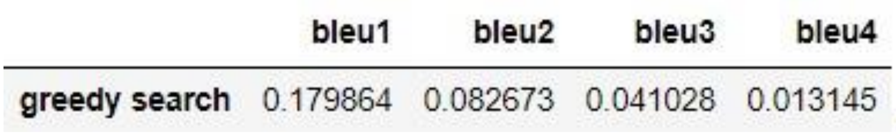
# Results

## Simple Encoder Decoder



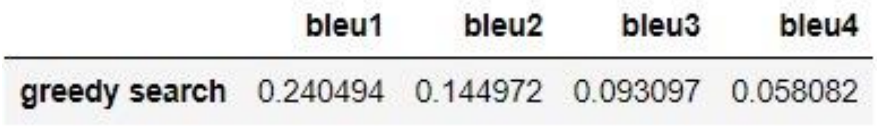
| | bleu1 | bleu2 | bleu3 | bleu4 |
|---|---|---|---|---|
| greedy search | 0.278839 | 0.193285 | 0.132483 | 0.078508 |
| beam search (top_k = 3) | 0.278839 | 0.193285 | 0.132483 | 0.078508 |

## Attention Decoder



| | bleu1 | bleu2 | bleu3 | bleu4 |
|---|---|---|---|---|
| greedy search | 0.179864 | 0.082673 | 0.041028 | 0.013145 |

## CAM Encoder



| | bleu1 | bleu2 | bleu3 | bleu4 |
|---|---|---|---|---|
| greedy search | 0.240494 | 0.144972 | 0.093097 | 0.058082 |