Chest x-ray Report Generation and Chatbot

Project Report

Presented to
CMPE 258
Spring 2023

By
Team Ctrl Alt Del

Sanjana Kothari
Neetha Sherra
Ananya Joshi
Naga Bathula

May 20, 2023

ABSTRACT

Chest X-rays are a common diagnostic tool used in detection and evaluation of various pulmonary diseases. However, generating reports based on these images can be time-consuming and prone to human errors, leading to delays in diagnosis and treatment.

To address this problem, we chose to build an AI tool that would read an image and generate tags from the image features and generate a report for that image based on chest image features and tags. In this project, we use a dataset of chest X-ray images and their corresponding radiology reports. We preprocess them by applying various image processing and NLP techniques. We then fine tune deep learning models which will extract relevant information from the images and generate corresponding reports.The generated reports (impressions and findings) are evaluated for the hamming loss and semantic similarity with the ground truth reports.

This project could have significant implications for improving the efficiency of radiology reporting, leading to faster and more accurate diagnosis hence leading to improved patient outcomes.

Acknowledgments

We would like to express our gratitude and appreciation to our respected Professor Vijay Eranti, Computer Engineering Department, San Jose State University for his constant support and guidance through this project.

Table of Contents

1. Introduction
1.1 Project Goals and Objectives
1.2 Problem and Motivation
1.3 Data Sources
1.4 High-level Process flow

2. Report Tags Generation
2.1 Model
2.2 Methodology and Implementation
2.3 Metrics and Results

3. Report Findings and Impressions Generation
3.1 Model
3.2 Methodology and Implementation
3.3 Metrics and Results

4. Medical Chatbot
4.1 Model
4.2 Methodology and Implementation

5. Application Functioning and Hosting

6. Conclusion

Chapter 1 Introduction

1.1 Project goals and objectives

The objective of this project is to leverage the power of large language models for generating textual medical reports from chest x-ray images. By utilizing the multimodal capabilities of large language models, we aim to develop an application that takes a chest x-ray image as input and generates a medical report from the x-ray, in a similar fashion as a physician or medical practitioner would. In addition to that, we also provide a chatbot facility built using Microsoft's Semantic Kernel that can look at the generated report and answer user questions based on the report as well as answer other general queries about chest-related questions. Therefore, it is a complete package that gives the user the diagnosis and observations from chest x-ray as well as give the user the ability to get specific answers to their questions about the report or general chest-related conditions.

1.2 Problem and Motivation

Medical images are widely used in hospitals for the diagnosis and treatment of many diseases. The reading and interpretation of medical images are usually conducted by specialized medical professionals. They write textual reports to narrate the findings regarding each area of the body examined in the imaging study.

Impression: No acute cardiopulmonary abnormality.

Findings: There are no focal areas of consolidation. No suspicious pulmonary opacities. Heart size within normal limits. No pleural effusions. There is no evidence of pneumothorax. Degenerative changes of the thoracic spine.

MTI Tags: degenerative change

Fig. An exemplar chest x-ray report

To correctly read a chest x-ray image, the following skills are needed: (1) thorough knowledge of the normal anatomy of the thorax, and the basic physiology of chest diseases; (2) skills of analyzing the radiograph through a fixed pattern; (3) ability of evaluating the evolution over time; (4) knowledge of clinical presentation and history; (5) knowledge of the correlation with other diagnostic results. For less-experienced radiologists and pathologists, especially those working in the rural area where the quality of healthcare is relatively low, writing medical-imaging reports is demanding. For experienced radiologists and pathologists, writing imaging reports is tedious and time consuming. In sum, for both inexperienced and experienced medical professionals, writing imaging reports is unpleasant. This motivates us to develop an application to automatically generate medical image reports, specifically for chest x-rays.

1.3 Data Sources

Open-i has a collection of chest x-ray Images from the Indiana University hospital network. Data contains two folders, one for x-ray Images and the other for the XML report of radiography. For each report, there could be multiple images.

The data files can be downloaded from:

- https://openi.nlm.nih.gov/imgs/collections/NLMCXR_png.tgz
- https://openi.nlm.nih.gov/imgs/collections/NLMCXR_reports.tgz

1.3 High-level Process Flow

There are two major components of the project-

a. Report generation from chest x-ray image
b. Medical chatbot to answer queries based on generated report

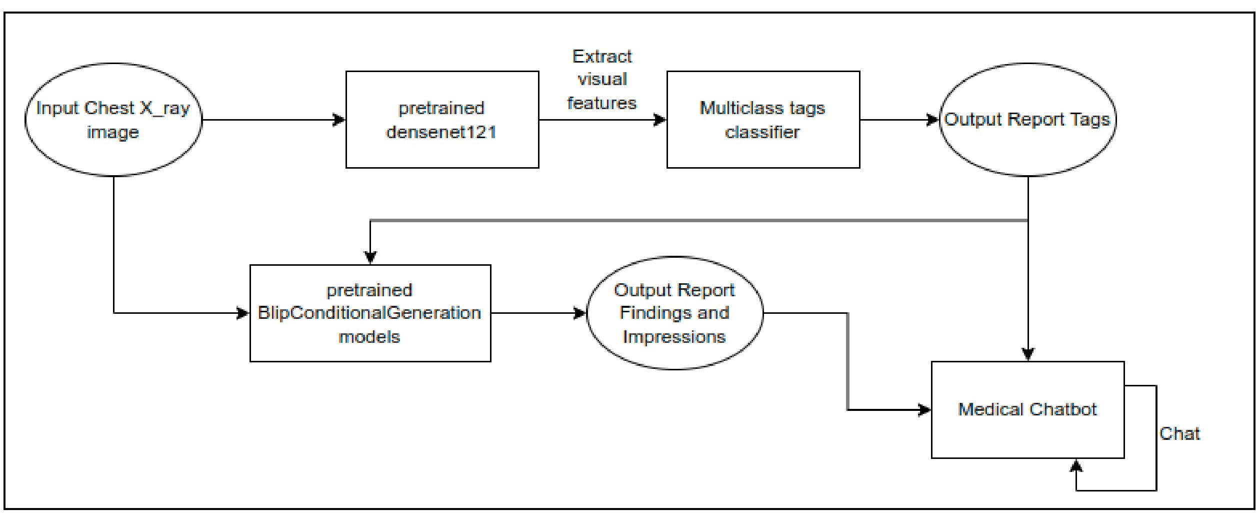


Fig. High level flow of report generation

Chapter 2 Report Tags Generation

2.1 Model

One of the parts of the x-ray report are the tags which list the critical keywords found in the findings section of the report. To generate these tags, we use the pretrained VGG-19 vision foundation model followed by a linear layer classifier for predicting the tags associated with the input x-ray image.

2.2 Methodology and Implementation

Tag generation involves two steps:

a. Extracting visual features from chest x-ray image
b. Using these visual features to predict the associated tags

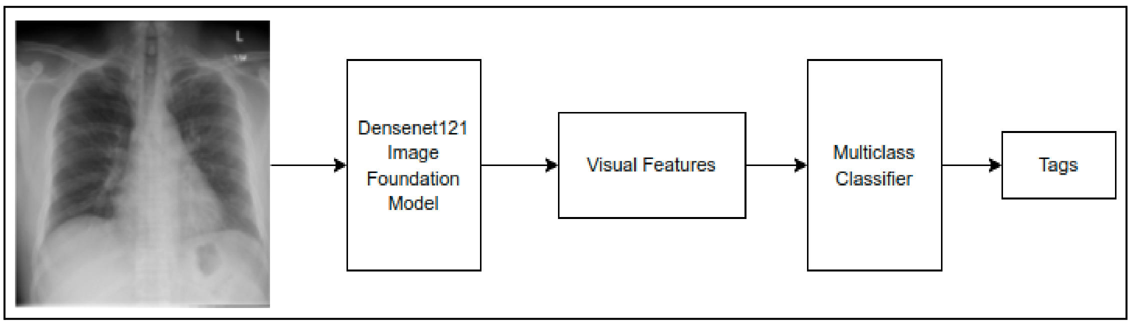


Fig. Tag Generation

The implementation steps are as follows:

a. Densenet121 is an image classification model consisting of 121 layers. This is an image classification model trained on ImageNet dataset. For our use case that involves medical images, we extract the representations from the convolution layers of Densenet121 and attach a AvgPooling and FullyConnected layer for classification. This architecture is now fine tuned using the chest x-rays by training the weights of the newly attached layers while keeping the weights of the pretrained layers frozen.
   Architecture -
   1. Input: The input to DenseNet-121 is an RGB image of size 224x224 pixels.
   2. Initial Convolutional Layer: The input image is passed through an initial convolutional layer with 7x7 kernel size, stride of 2, and padding of 3. This layer performs convolution on the input image to extract initial features.
   3. Max Pooling: Following the initial convolutional layer, a max

pooling layer with a kernel size of 3x3 and a stride of 2 is applied to reduce the spatial dimensions of the feature maps.

4. Dense Blocks: DenseNet-121 consists of four dense blocks. Each dense block consists of a series of densely connected convolutional layers. Within each dense block, the feature maps from all preceding layers are concatenated along the channel dimension and passed as input to subsequent layers. This dense connectivity pattern allows feature reuse and promotes information flow throughout the network.
5. Transition Layers: Between the dense blocks, transition layers are inserted to reduce the number of channels and the spatial dimensions of the feature maps. Each transition layer consists of a 1x1 convolutional layer followed by a 2x2 average pooling layer. The 1x1 convolutional layer helps in reducing the number of channels, and the average pooling layer reduces the spatial dimensions.
6. Global Average Pooling: After the last dense block, a global average pooling layer is applied. It computes the average value for each channel across the spatial dimensions, resulting in a fixed-length vector representation of the features.
7. Fully Connected Layer: The global average pooled features are then fed into a fully connected layer with a softmax activation function. This layer maps the features to the desired number of classes and produces the final classification probabilities.

b. We extract the visual features from the last convolutional layer of the network and apply average pooling to reduce the dimensions.
c. These features are then passed to a linear layer multi-label classifier that performs classification for tags. We have a total of 210 tags in the whole corpus and for every image we take the top ‘n’ tags (where n = number of original tags for that image) with the highest probability assigned by the softmax function.

Fine tuning methodology:

a. The densenet121 model is fine tuned on 6512 x-ray images from the training set for 100 epochs.
b. Before passing them to the network, we perform random cropping, horizontal flipping and normalization on the images.
c. We also have a validation set of 403 images to test the performance of the

network at each epoch. The validation images are passed through a similar transformation process except that the images are not flipped.

d. We use the Adam optimizer and BCE loss function while fine tuning.

2.3 Metrics and Results

Tag generation is a multi-label classification problem. Each x-ray image will have 'n' number of tags associated with it out of the possible 210 tags. For this problem, we evaluate the performance of the classification model using hamming loss.

The Hamming loss is the fraction of labels that are incorrectly predicted.The value of hamming loss lies between 0 and 1 and a smaller value is preferred.

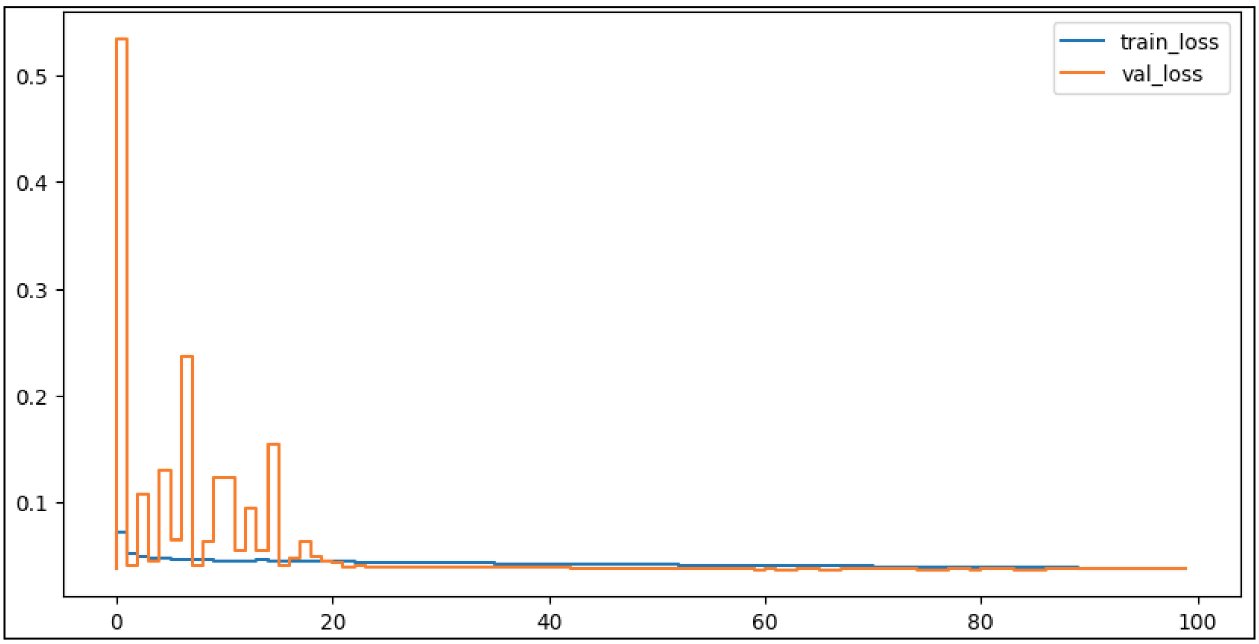


Fig. Training and validation loss for tags model over 100 epochs

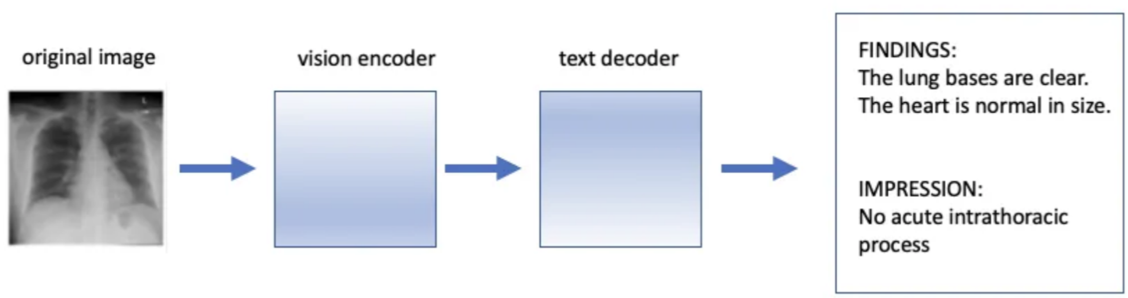
Chapter 3 Report Findings and Impressions Generation

3.1 Model

In addition to tags, a chest x-ray report also consists of findings and impressions. The impression section provides diagnosis and the findings section lists the observations regarding each body part examined in the imaging study. For the purpose of generating impressions and findings, we use the BlipForConditionalGeneration model from Hugging Face which is a transformer architecture based on the Generative Pre-Trained Transformer (GPT) architecture. For our purpose of generating text from the image of the chest x-ray, we use the [generate-cxr]() model which is modified for taking into account both image and text inputs to generate output text. Internally, it uses a BlipProcessor to achieve this.

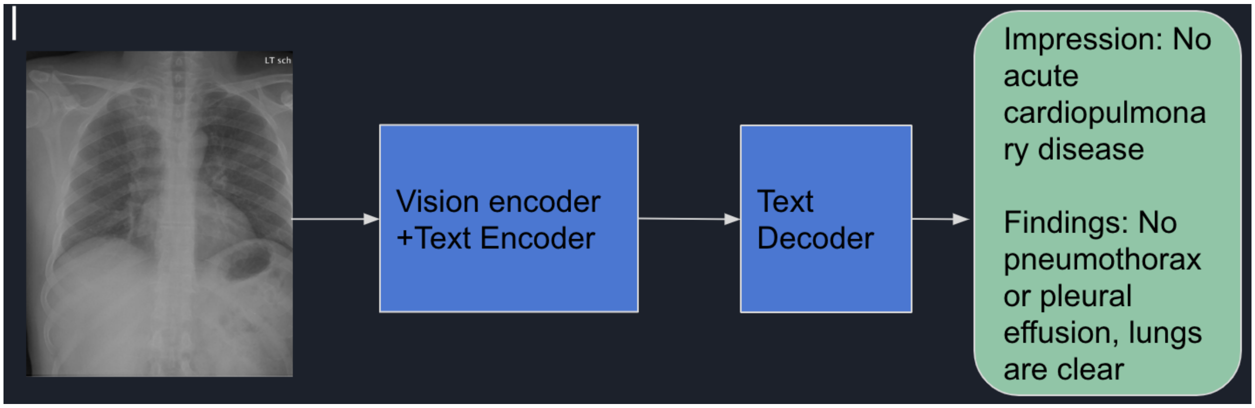
3.2 Methodology and Implementation

Most of the previous method in this area of research has only used visual features and all the previous architectures to solve this problem look like below



To realistically describe what a radiologist is doing when they write a report a deep learning model needs to accept the same inputs. This means the conditional generation of radiology reports should include both image and text inputs and have a text output.

A new transformer architecture particularly suited for this type of multi-modal problem was just released by SalesForce (Li et al. 2022). BLIP has a dual text and vision encoder paired with a text decoder. This allows it to continue generating new text for a radiology report from a given prompt's starting point.



The focus again in this module is on fine tuning a large language model, i.e. the BlipForConditionalGeneration model. We define the hyperparameters for training and use our dataset for tuning the model. The dataset is passed through the BlipProcessor for processing the image and tags associated with it so that the image-tags pairs can be used for generating findings and impressions of the image. Finally, the model is fine tuned for 5 epochs over 50 examples.

3.3 Metrics and Results

To measure the semantic similarity between the generated impressions and the ground truth impressions, as well as the generated findings and ground truth findings, we embed them using the medium sized corpora from spacy and then perform semantic similarity. A higher similarity score suggests that the predicted and actual report components match closely.

Chapter 4 Medical Chatbot

4.1 Model

As a useful addition to medical report generation based on image features and tags, we have built and integrated a chatbot which takes the generated report as input and can answer basic questions on the report to the patient. It has handy answers for any immediate questions that a patient has to ask regarding his report.

4.2 Methodology and Implementation

We used semantic kernel's capabilities to populate the memory for the chat and build a context on which the bot can answer questions that a patient asks. Semantic kernel majorly helped us to populate the bot memory with previous questions and the report and maintain the context for that session.

Semantic Kernel (SK) is a lightweight SDK that lets you easily mix conventional programming languages with the latest in Large Language Model (LLM) AI "prompts" with templating, chaining, and planning capabilities out-of-the-box.

When a user asks something, it is driven to a dynamically informed outcome with the orchestration capabilities of the kernel. Starting from a user's ask to getting what they want can be represented as a flow of connected parts:

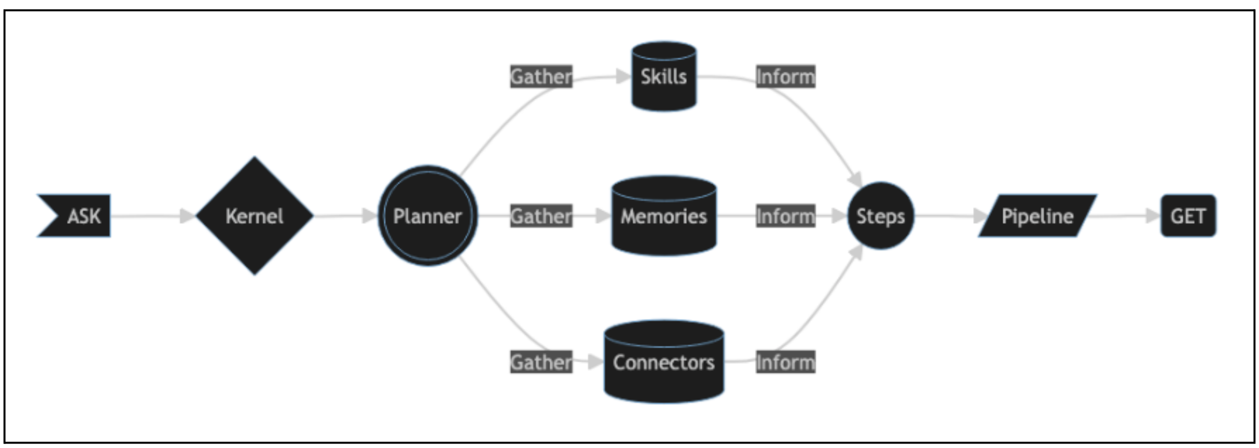


Fig. Semantic Kernel pipeline

A user's query is sent to a semantic kernel. The kernel orchestrates this query, the planner in the above diagram basically breaks down the query and gathers available resources that are in skills(typical questions - prompts), memories(context) and connects them. Basically it leverages the available resources from the populated memory. It then executes the steps in the pipeline to get the response for the query and sends it to the user.

Chapter 5 Application Functioning and Hosting

Currently, the application is hosted on Streamlit, an open source framework for machine learning projects.

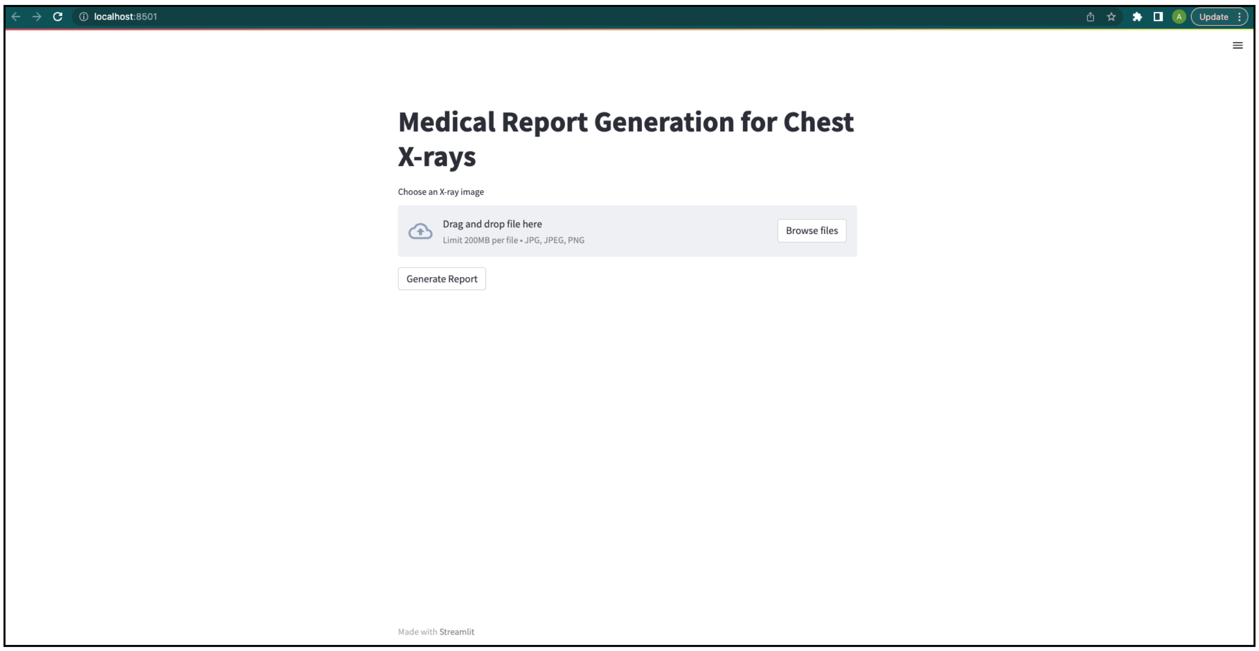


Fig. Streamlit app running

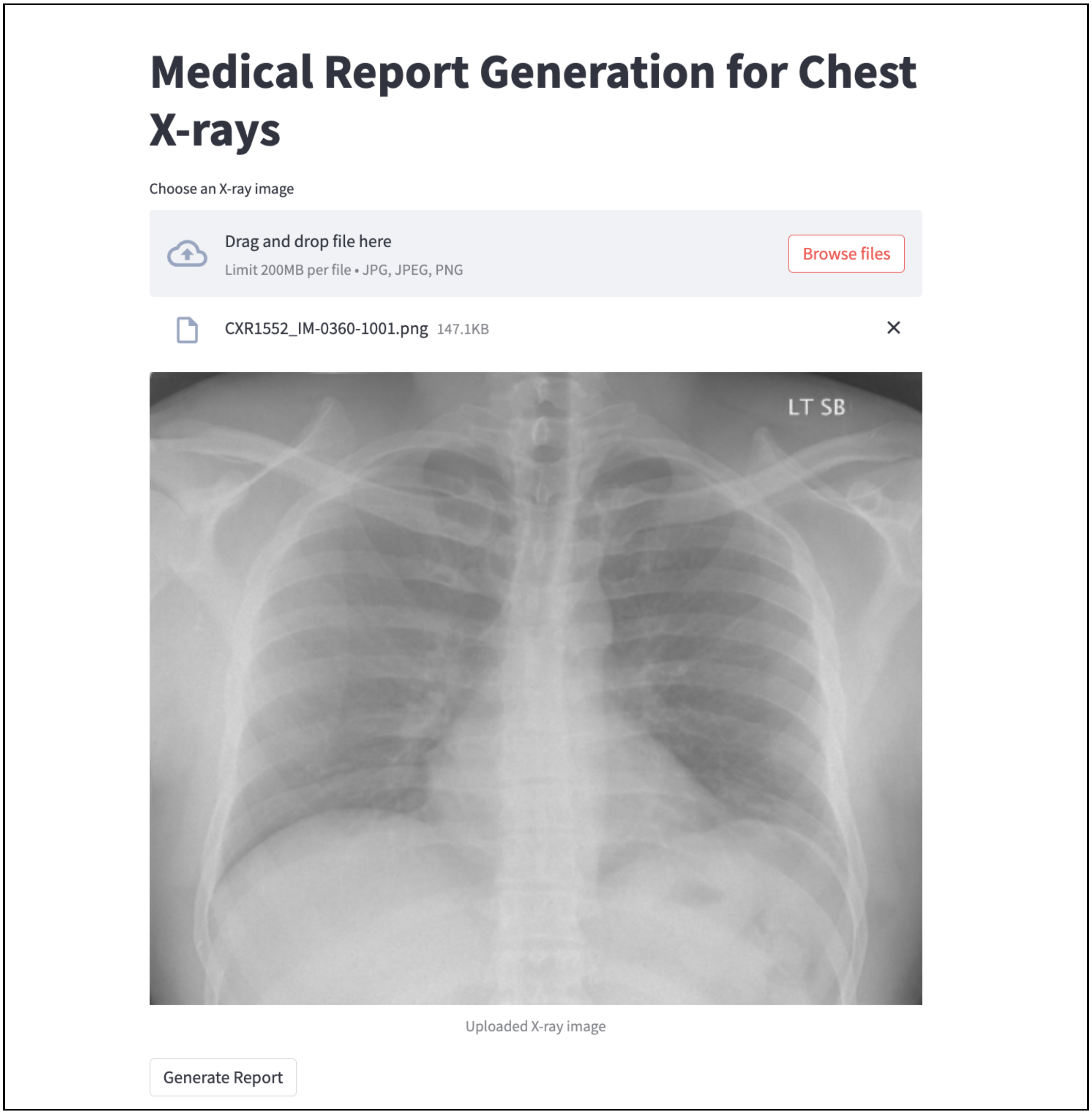


Fig : Uploaded a chest x-ray image and clicked on generate report

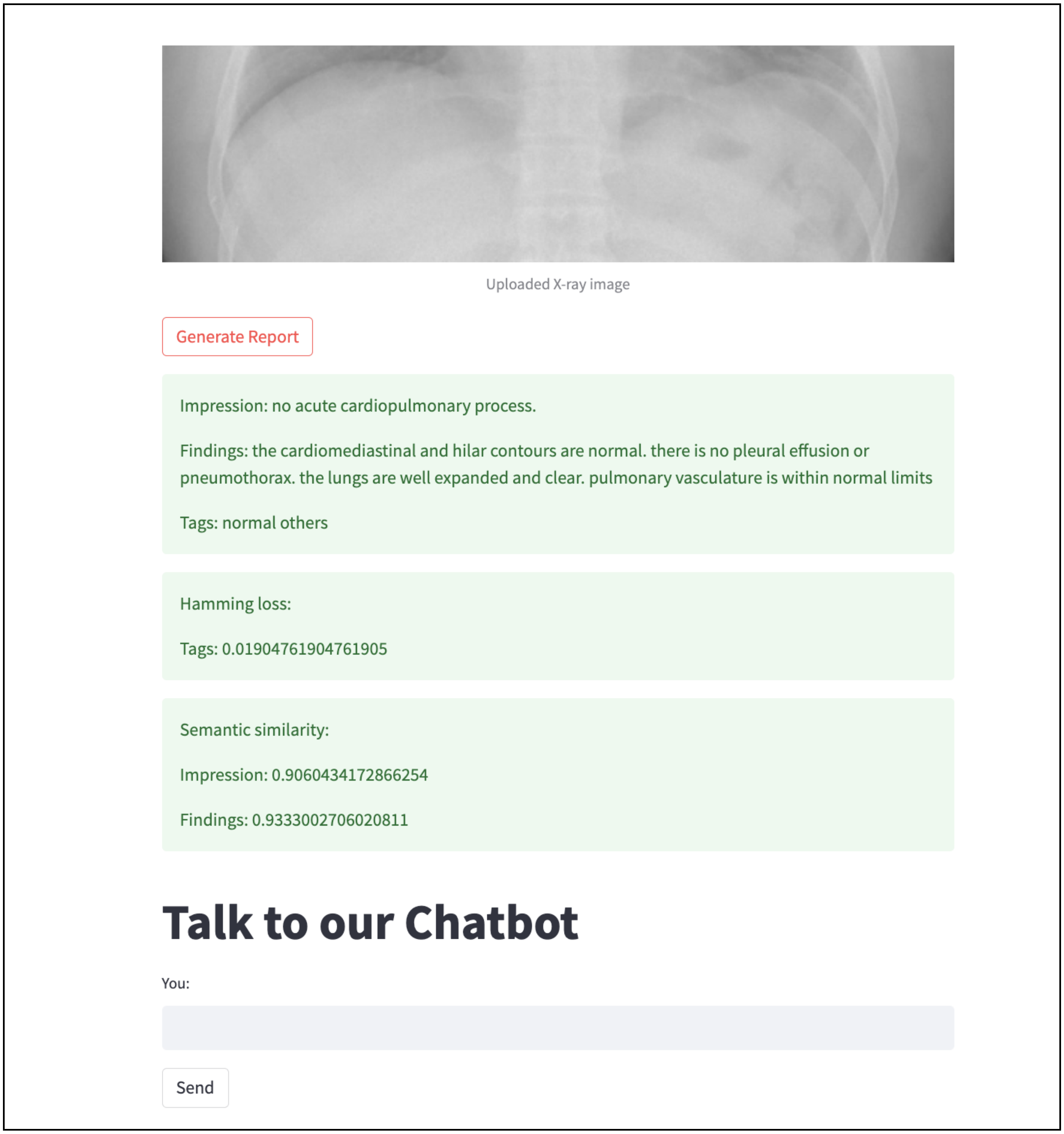


Fig. Report is generated along with the metrics and chatbot is activated

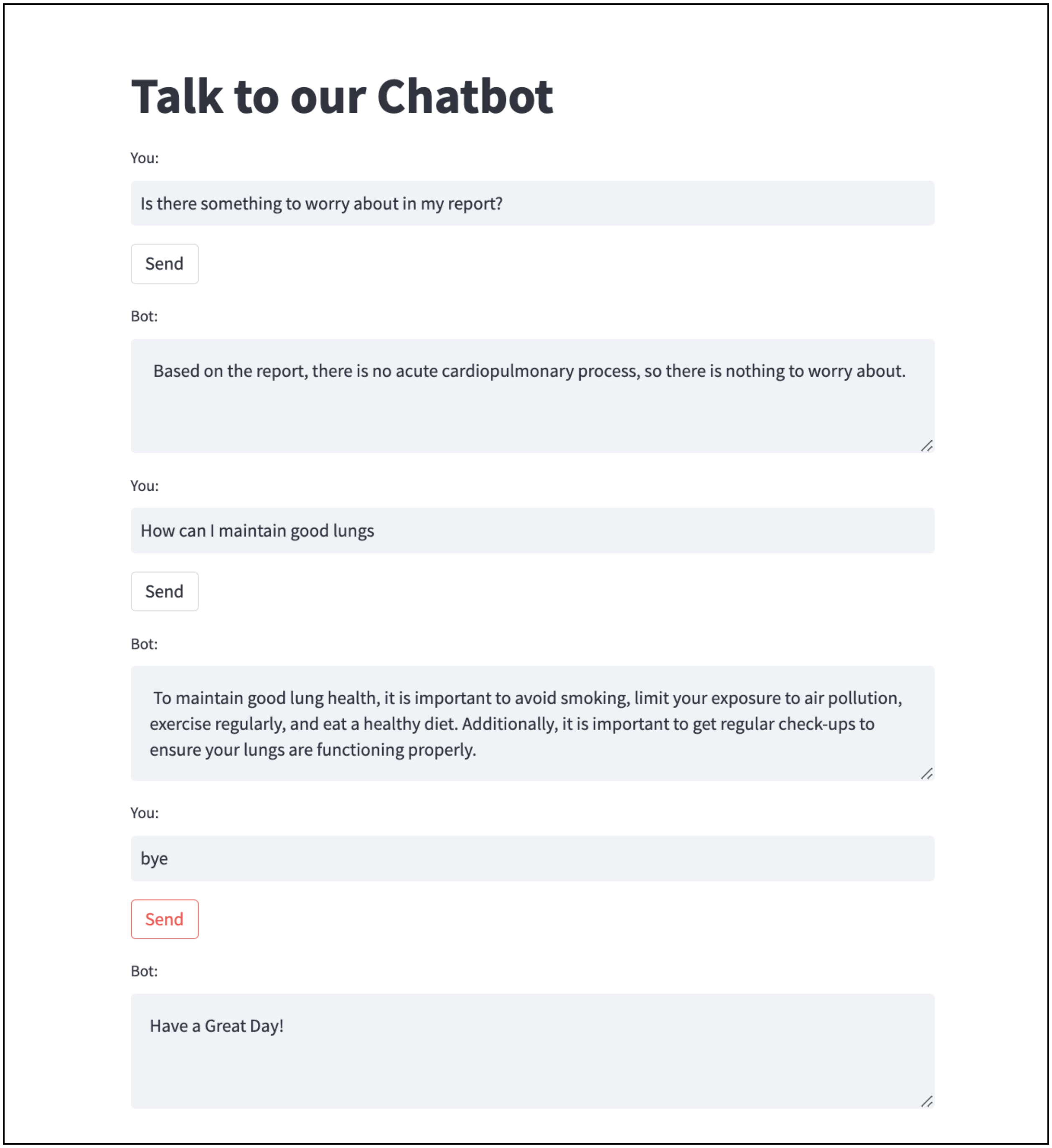


Fig : Chatbot is already loaded with the generated report and answers patient queries based on the report generated.

Chapter 6 Conclusion

In conclusion, the goal of the project was to generate reports from chest x-rays where the only available information one has is the chest x-ray. By leveraging the developments in large language models for both text and vision, we aim to reduce the burden on physicians of creating these reports manually. Additionally, using a semantic kernel for building a chatbot that can help a patient in answering questions about their report can be a big boost especially in scenarios where medical professions are not immediately available for translating the report to the common man.