DS 440

Image Caption Generator

Final Report

Team 11: Congqi Lin, Weiting Yu

4-17-2023

Abstract

This report details the creation and outcomes of an image caption generator developed for a senior capstone project. The report focuses on the advantages and purpose of the application in assisting users in creating automatic, evocative captions for photos. In order to extract information from an image and produce text that accurately characterizes the image, the application uses deep learning techniques that combine computer vision and natural language processing. Convolutional neural networks (CNNs), recurrent neural networks (RNNs), and long short-term memory (LSTM) networks were all used in the construction of the image caption generator. The website can be opened when the front end is connected to the HTTP port successfully. Despite the constraints of time, finances, and technical limitations, the team has created the application and run it successfully. A translation function has also been added which is convenient for users to use. The team also tried to make some functions that can benefit people with visual disabilities to use the application.

Table of Contents

Abstract 1
Table of Contents 2
Introduction 4
Problem Statement 4
Motivation 4
Objective 4
End User Needs 5
Literature survey 6
Literature Review 6
Assessment of Available Solutions and Techniques 7
Pros and Cons 7
Requirement Specification 9
Market Requirement Analysis 9
Design Requirement Analysis 9
Constraints 10
Assumptions 10
Outcome Criteria 10
Risks 11
System Development 12
System Planning 12
Principle Operation 13
Design Process 14
System Design 15
Choice from design process 15
Front End and Deployment 17
Back End 17
Functional Decomposition 19
Testing 20
Testing 1 20
Testing 2 21
Testing 3 22

Testing 4 25
Conclusion 26
Discussion 26
Changes in Development 26
Challenges 27
Lesson Learned 27
Future Work 28
More friendly for people with blind and visually impaired 28
Enhance the performance for uncommon images 28
Interact with other applications 29
Reference 30

Introduction

Problem Statement

Although sharing images is becoming more and more popular, consumers sometimes still struggle to come up with the correct words or definitions for a picture. This problem is not straightforward for businesses to deal with. Especially for some businesses that rely on images. With the important development for the visual contents, it is greatly demanded that a tool showing accurate and correct captions for the image can be used. And also to improve social engagement on social media platforms. People from all over the world can also use this application efficiently. A tool is also needed for people with visual disabilities which can help them read the correct objects from the pictures.

Motivation

All team members are interested in AI drawing and its inverse use which gave us an idea for the image caption generator. After going through some research paperwork, we learned that the project is useful for different commercials and people. We would like to use our project to help people in a positive way so we decided to do this project. This is our motivation.

Objective

The objective of our project is to create a useful website that contains the generation and translation functions. The final web page will be connected with a public URL based on our successfully created front end and the model and function we created successfully in the backend. As result, a complete image caption generator will run fluently through the website. This includes a Streamlit front end and Python back end.

End User Needs

The target end-user include social media users, e-commerce retailers, and users with disabilities. This project is for helping people to make their lives easier. For social media users, the generator can quickly generate the keywords in their posts through social media. That way it would be convenient for them to create more engaging content. For e-commerce retailers, the generator can help them to provide more useful information about the products. Especially, if the consumer wants to find something online with a picture, the generator can produce the keywords immediately to help them shop well online. For users with a disability, this generator can help them understand the visual content with text information.

Literature survey

Literature Review

We found some websites and previous works for the literature review to brainstorm and inspire our project. The combination of CNN and RNN is always the best choice for image-related tasks. One of the most useful implementations for us was the Generating Image Caption demo made by Milhidaka on GitHub in 2021. In this demo, users could input an image and then would get the top 10 most relevant descriptions of the image. It used the Caffe model, which was a combination of CNN and RNN to train the model on the Microsoft Common Objects in Context (COCO) dataset. The dataset was large enough to train and needed a large amount of time. The outputs of this application were 10 complete sentences related to the content of the image. They might contain words like what the object in the image is and what color it is and the action of the object.

Another website was an application that could generate captions for an image from the phone. It was basically using the CNN model with LSTM, which was used as a sentence generator and translation, and CNN, which was used as image representation. This website is an implementation of Oriol Vinyals's paper in 2015 and also provides a neural style version, which is used to change the style of the image. The original paper was about comparing the accuracy by training the same model with different combinations of 5 different training data including MSCOCO and Flicker8K with several evaluation matrices.

Based on what we learned from previous works, we would use CNN and LSTM to train our model for our back end. This website could have the same and even better function as the websites mentioned in the literature review. We would also add more functions such as translation to our web application.

Assessment of Available Solutions and Techniques

This tool provides the basic descriptions for the content in the image. It is very easy to use and provides great accuracy results for common objects. There are a lot of ways for using image captions in our lives. It can be used through social media. If a user uploads a picture on social media, the generator can identify what appears in the media and show keywords from the media. People can understand quickly what appears in the pictures through social media. The social media company could add this function to create tags for pictures automatically. It can automatically generate captions for images uploaded to social media platforms to improve the user experience. Researchers in the lab can use it to summarize news images in articles or broadcasts to provide a quick overview of the content.

Pros and Cons

Based on what we learned from these two websites, we summarize the advantages and disadvantages. These two websites are both user-friendly and provide high accuracy for analyzing the image. The functions of these websites are complete in some areas. We believe these two websites have done a good job of generating captions. The first website has a cool function that shows the bar of progress. It can let people know that the application is running in which step. For the second website, we find that it has a translation function for the results.

The disadvantages of these two applications are also obvious. For the first website, the data used to train the model is a little old. Sometimes the result in texts might not be as accurate as we think. The model for image captions lacks diversity. It may repeat the captions. Image captions can also be biased toward certain objects and attributes. Because of these cons, we first may use a more up-to-date dataset for training. In addition, we may use the combination of CNN and LSTM, which are the main trend of the image caption, as our model. We also plan to add a new evaluation matrix to provide the most accurate result. For the second website, there are no alternative results even though the description is not very accurate. It focuses more on comparing the accuracy of different data rather than the practicability of the

application. When we get the results for our own project, if the accuracy is not high enough, in order to make a better prediction, we will create more results for our generator. Only one description is not very useful sometimes. We also plan to only select one dataset because it would be more time-saving and have almost the same results. We may also add translation functions to translate the results into other languages.

Requirement Specification

Market Requirement Analysis

According to the report from Marketsandmarkets, the image identifies market size was growing highly from 2020 to 2023 and is predicted to keep increasing till 2025. The market price increased from 26.2 billion to 53 billion at the same time. Based on this report, we find that there is great value in this market, which means our project has a great chance to develop well in the market.

Even though there is still high competition in the market because there are a lot of big technology companies, such as Google, Microsoft, etc., the market for image generators is not saturated. There is still a great chance for new developers like us to get into the market and create great value. The most important element for entering the market is the accuracy and practicality of our application. The more accurate our project is, the more chance that companies will use our product. Based on this, we need to adjust and modify the model as accurately as possible.

Design Requirement Analysis

While we are designing the application, we need to be sure that what components are necessary for the project at the beginning. We completely know the importance of the back end in our project so we worked on it first. We need to figure out how to modify and run the model successfully. We also need to find an appropriate dataset to fit into the model. Setting correct epochs is also important since our computer is not highly efficient enough for running the model.

The second step is to create a frontend page for our project. To make a web application, we have to set some basic functions like user input and user interaction. It is also important to make sure our work can be accessed by anyone anywhere. We also need to find a way that can run our model on the front end successfully.

Constraints

One constraint we recognize this time is the problem with the accuracy of the model. Since we need to train our model with a huge amount of image data and many epochs, they are really time-consuming and require a computer with high-quality performance. It is difficult for us now to solve this problem by ourselves. If we develop this product full-time, there would be a possible solution for us. In this situation, the coding part can be improved more professionally, especially for some details and parameters in the model.

Another constraint is time. This project is due before the end of the semester. Neither of us has experience in building the front end and making the deployment. In this case, time is very limited for us. We have to learn new approaches and knowledge as soon as possible to make sure our project is on schedule.

Assumptions

When we are preparing for our project, we assume that the market for AI drawing and image prediction will develop well and will be in high demand in the future. In the past years, intelligent applications did not play that much important role in people's daily lives. In the new era, we believe that this will be a great market for us to discover. This belief and assumption will be an important key to our success.

Outcome Criteria

There are a few outcome criteria for our project. One is the accuracy of the results. The results generated from the picture should be accurate and relevant. We can see if the results are accurate by human vision or using some evaluation matrix. The results of the content should also be diverse. Since there might be more than one object in the pictures, the caption should appear with more than one result. Also, speed is one of the outcome criteria. The time it shows the result can reflect the success of our application.

Risks

Inaccuracy is one of the risks of our project. As we mentioned before, image captioning is not always accurate. It has a high chance that giving the wrong results for the pictures. Another risk is the bias for our project. Since we train the model we used in a limited database, the model would have a preference from the database. If we put a different category's image to the generator, the caption might not accurate as what appears in the picture. If the user has no ability to distinguish whether the result is correct or not, there will be so many hidden dangers.

System Development

System Planning

To plan our project, we started by breaking down all required tasks into different categories. As our project was aimed at building a web application, we identified the need for a front end, a back end, and deployment. Because this was a class project, we also needed to submit assignments at each step. Our initial breakdown of work is shown in the following Gantt Chart, and it helped us to organize our tasks and responsibilities. As our project progressed, we made adjustments to the subtasks within each category while keeping the four main categories consistent.

1	Class Deliverables	Weiting, Congq	2023
1.1	Team formation		2023/1/15
1.2	Online Search		2023/1/22
1.3	Project brief		
1.4	Literature Review		2023/1/29
1.5	Paper Demo		2023/2/5
1.6	Project brief		2023/2/5
1.7	Project Specs		2023/2/5
1.8	Charts&Networks		2023/2/12
1.9	Midterm presentation		2023/2/19
2.1	Midterm peer review		2023/2/26
2.2	Implementation		2023/3/4
2.3	Enhanced implementation		2023/3/19
2.4	Testing		2023/3/26
2.5	Revise, Redo		2023/4/2
2.6	Results, conclusions, future work		2023/4/7
2.7	Final Report		2023/4/7
2	Front End		2023/3/1
2.1	Create home page		2023/3/19
2.2	Create map page		2023/3/19
2.3	Test front end		2023/3/19
3	Back End		2023/3/3
3.1	Collect data		2023/3/4
3.2	Data integration		2023/3/4
3.3	Train /test model		2023/3/4
3.4	Hyperparameter Tunning		2023/3/4
3.5	Model evaluation		2023/3/19
4	Deployment		
4.1	Obtain domain name		2023/3/19
4.2	Deploy frontend to domain name		2023/3/19
4.3	Deploy backend to web service		2023/3/19

Figure 1 Gantt Chart

The first category was the class deliverables, which was a symbol of a landmark of each step of our project. The first subtask was the team formation, which began on Jan 15th, 2023, and the last subtask was results, conclusions, and future work, which ended on April 7th, 2023. This category ran through the whole semester.

The second category was the front end, which was the website we designed. We used ngrok and Streamlit for all tasks in this category and planned to create the home page, map page, and testing.

The third category was the back end, which contained our dataset and model. Tasks in this category were much more complex and we spent much more time on them. We separated this category into 5 subtasks, which were collecting data, data integration, train/test model, hyperparameter tuning, and model evaluation.

The last category was deployment. As a beginner in website construction, we didn't have enough experience and professional knowledge. In this case, we need to do a lot of research for this category and spend a lot of time. We generally separated the task into three parts, which are the domain name, the connection between the front end and back end, and the connection between the front end and the domain.

Principle Operation

The principal operation begins when the user navigates to our website on their web browser. As shown in the operation flow figure, after entering the site, user can either upload their image by clicking the "Select Image" button on the computer or take a picture using their phone's camera by clicking the "Take Picture" button. With this image as the input, the user can click on the "Generate" button to make the whole process start. After the previous action, the image will be transformed into a given format in the back end and then sent to the model. The model will generate captions in this process and show them on the website. With the caption generated, the user can make further actions like copying the captions by

clicking on the “Copy” function and translating the captions into other languages by selecting the target language in the “Language” select box.

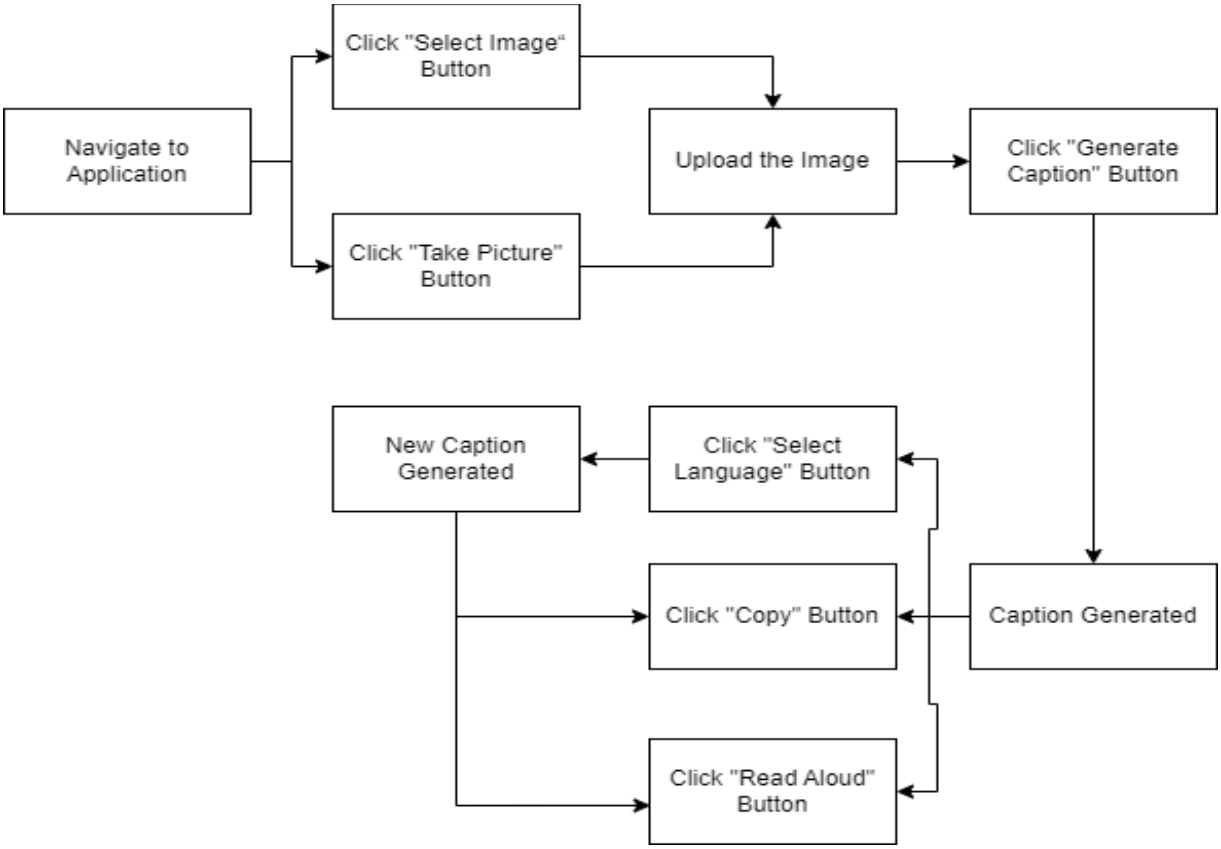

Figure 2 Operation Flow

Design Process

As a beginner in website construction, we didn’t have much experience and professional knowledge about how to build the front end and how to keep the connection. We generally needed three components to keep our web application working, which are front end, back end, and deployment. We did a lot of research on these components and tried to find out several possible options. Our front end should be a page that allowed inputs from users and showed the output from the model. Based on our experience and

research, we could use some traditional methods including JavaScript and HTML, or some new ways like Streamlit to make the user view. We also tried to learn how to use VUE, which was a very popular framework for the front end. For the backend, we were looking for a method that could easily deal with large-size data and perform well in image-related tasks and deep-learning models. For the deployment, we needed to get a domain name and connect our back end and front end with the web service. In this component, the cost was also an important factor we consider about.

System Design

Choice from design process

After carefully reviewing our project purpose and research results, we decided to find a way that could directly build the website page on Python. With this clear objective, we decided to use Streamlit and Ngrok for building the front end and making deployment. Streamlit was an open-source Python library that was very easy to use. It perfectly satisfied our requirement that we could design the website in Python and load the model. Additionally, it could create interactive widgets by simple syntax, update the app automatically with the change in code, provide a variety of built-in widgets for user input like a dropdown menu, support popular data science libraries like NumPy and Scikit-learn, and integrate with different data sources easily. Streamlit set up our web page, while Ngrok made the deployment. By using Ngrok, we could create a secure tunnel between our local computer and the Ngrok server. After that, our local web server would be exposed to the internet with a public URL, which can be used by anyone anywhere to access our web application. For the backend, our first choice was Python with Google Colab. We were familiar with them and had a lot of experience with them. Python provided many useful packages and libraries in data processing, deep learning, and translation. We could easily do all steps on it. It also had a function called pickle that could store our model after training as a “. pkl” file. It was very convenient

for us to connect the model with the web application directly. In this case, we didn't need to train the model each time we used it. With the Google Colab, we could store our training and testing data, which contained over 8000 images, in the Google drive and didn't need to upload them to Python every time. It also provided the GPU that could make the whole process run faster and allowed us to modify the model better with more epochs and tuning steps during training.

The steps we followed to develop each part of the application are shown in Work Breakdown Structure below.

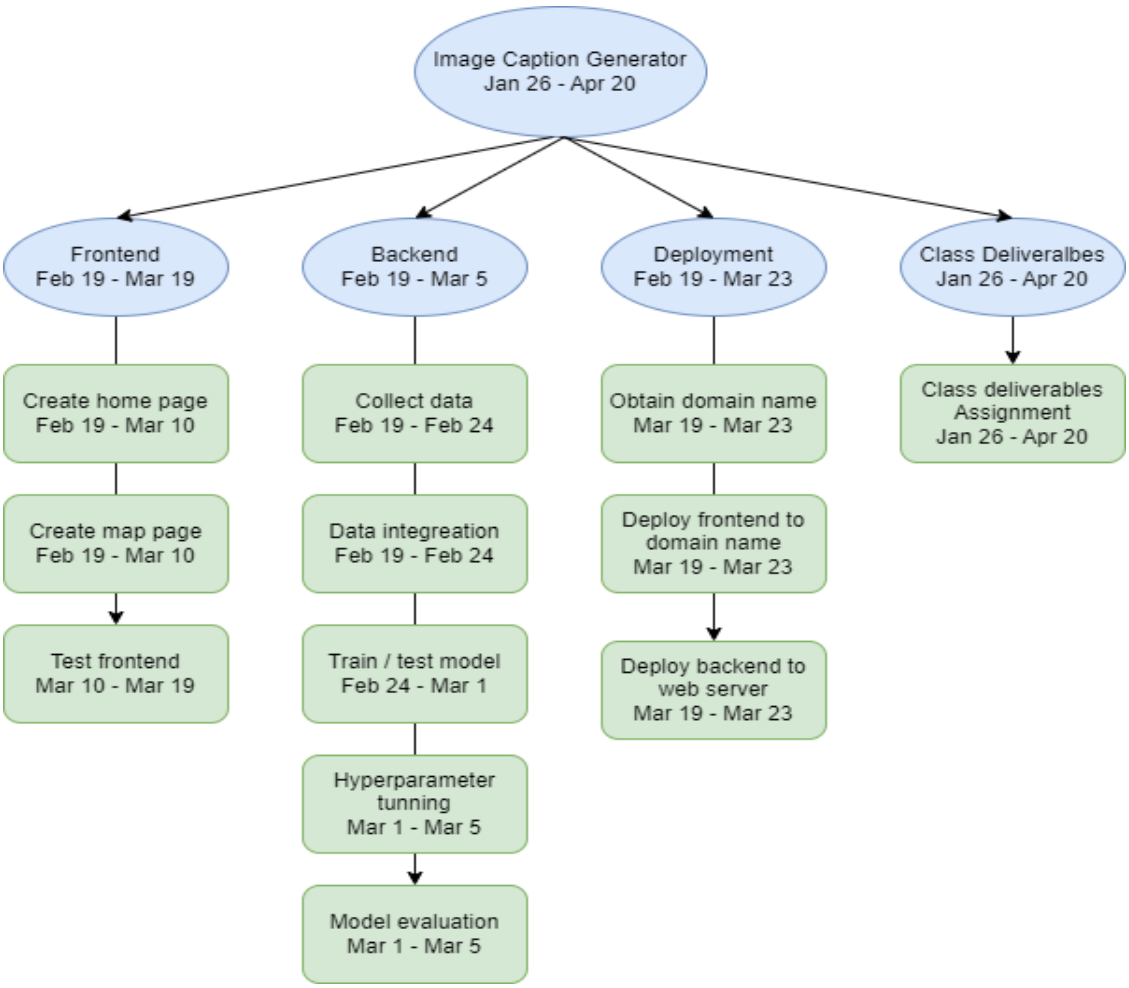

Figure 3 Work Breakdown

Front End and Deployment

App.ipynb contains all contents for our front end and deployment. It is both the entry point and the core part of the user interface. Most of the code in this file is based on pyngrok and streamlit. After importing these two into the notebook, with access to Google Drive, we could easily build our web application.

To implement all functions we design and keep their work on the web page, we use the %%writefile command to write the contents of the back end to a ".py" file which is better for the front end need. In this section, many functions in streamlit are used to design the website: set_page_config() for setting the page title, header() for displaying the topic of the website, file_uploader() for allowing user upload images, button() and selectbox() for adding interactions for the user, and image() for showing the input.

For the deployment, a ngrok authtoken is needed. This authtoken allows us to set a public URL with a selected HTTP port. This URL will be the link to access our web application. After that, by using streamlit, the ".py" file we get from the previous step and our local tunnel will be connected with this public URL and thus our web application can be used by anyone anywhere.

Back End

Backend. ipynb contains all contents for our back end. The Flicker8k from Kaggle is the dataset we use. There are generally 3 components in this section: image representation, caption generation, and translation. All models in this section are provided by the TensorFlow library.

In image representation, all images in the dataset are loaded, converted to numpy array, reshaped and preprocessed for the model, and extracted features by VGG16 model, a convolutional neural network that is 16 layers deep. All features are stored in a pickle for future use.

In caption generation, all text data are preprocessed by converting to lowercase, deleting digits, special characters, etc., deleting additional spaces, and adding start and end tags. All these preprocessed captions are stored in a list. The tokenizer is fitted on this list as a processing method and stored in a pickle. After

that, all data are split into two parts: 80% for training and 20% for testing. For the model, there are two models in this part: the encoder model with image feature layers and sequence feature layers and the decoder model. LSTM is also used in this step. The model plot below shows the concatenation of the inputs and outputs into a single layer. The final model is trained with the training data and 30 epochs and then saved in a pickle. The final step is using the tokenizer and model to generate the caption by adding a start tag for the generation process, encoding the input sequence, padding the sequence, predicting the next word, getting an index with high probability, converting the index to word, and appending word as input for generating next word.

In translation, the translate library provides the function of translator. By defining a translate_text() function, we can easily translate the captions into English, Japanese, Korean, Portuguese, Spanish, Chinese, and Chinese Traditional.

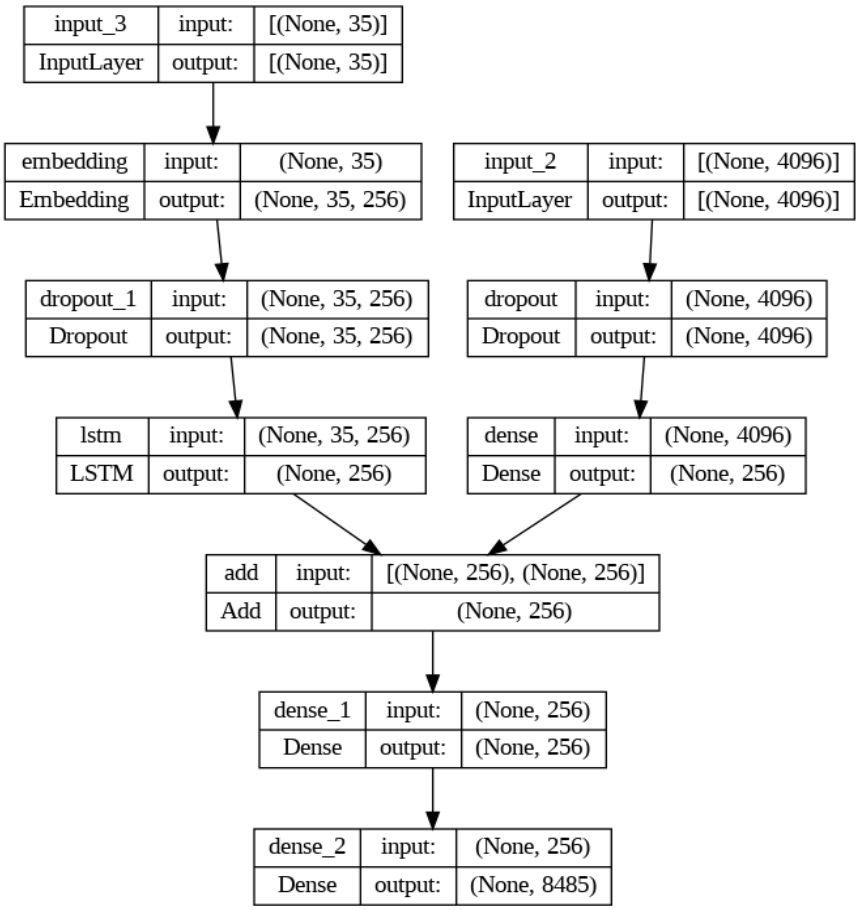

Figure 4 Model Plot

Functional Decomposition

The main purpose of this web application is to generate captions for the image. As a result, all functions work for this goal and there is only one use case. No matter what operating system the user uses to access the application it will go through the same processes. Once the image is uploaded, all stored models, functions, and tokenizers will be loaded. In the first step, the image will be preprocessed and transformed into a feature vector. This feature vector will be sent to the model. Both the encoder and decoder will run in the back end to generate the captions for the image and then translate them to a given language. Finally, both the image itself and the caption will be shown in the application.

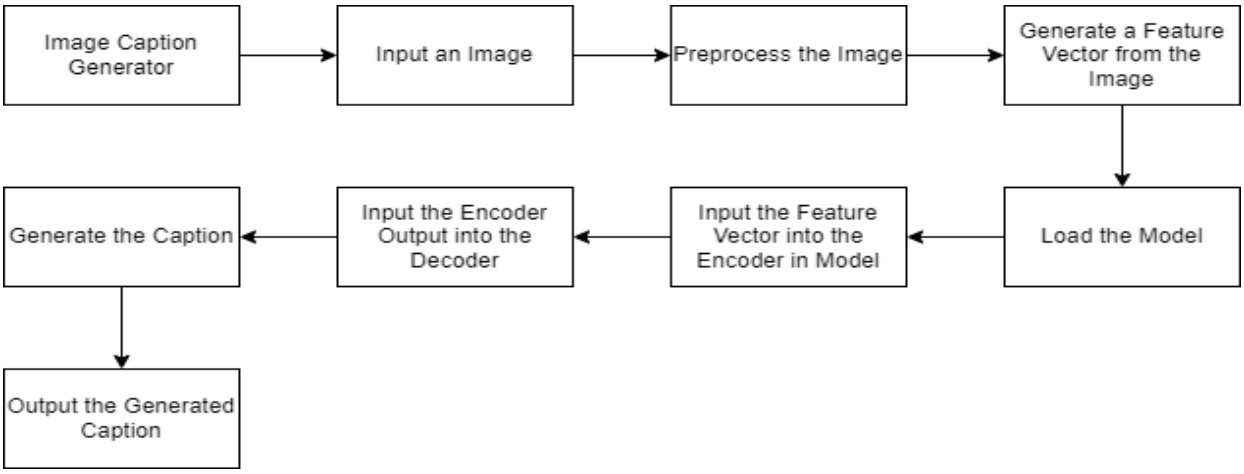

Figure 5 Technical Flow

Testing

Testing 1

We try to test our website on different operating systems. We select MacOS, Windows, IOS, and Android as our testing systems. The result shows that our website works well on Android and Windows. However, for IOS, it saves the image as JPEG automatically, which is not allowed on our website. As a result, there will be no input and the whole process can't work. The screenshot below shows the test result in IOS.

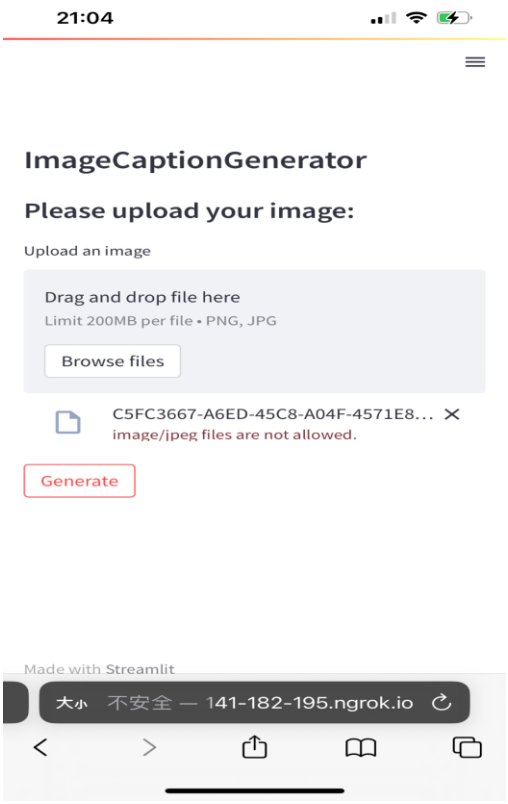

Figure 6 Failing Test Result in IOS

To solve this problem, we rewrite the input_buffer and change the requirement for the upload file in the front end code so that images in "jpg", "png", and "jpeg" can all be uploaded. The screenshot below shows the test result after modifying.

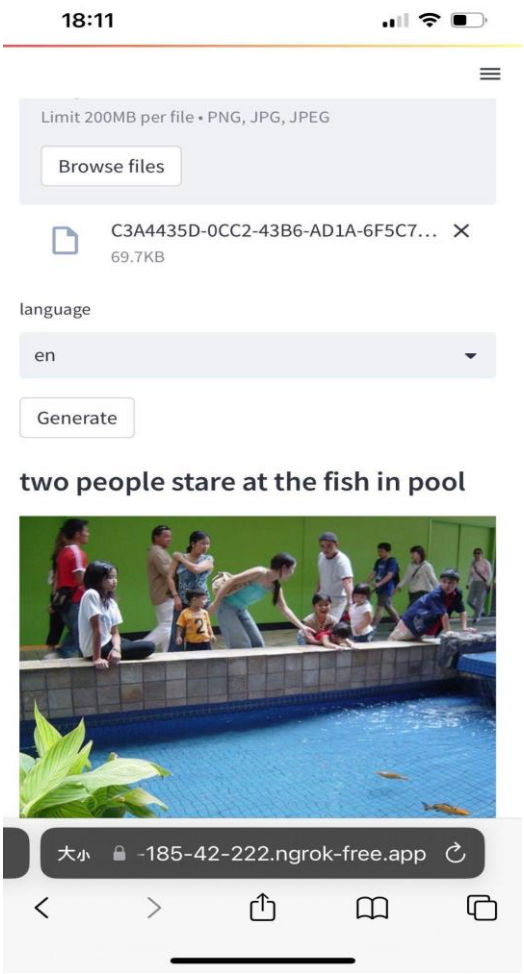

Figure 7 Successful Test Result in IOS

Testing 2

We use the streamlit to build our front end and use the ngork to connect the front end with the HTTP port. We have to keep our back end code running during we access the web application. In this attack, we simulate the case that there are problems in our bac kend. The result shows that the tunnel is not found and we can't access our web application.

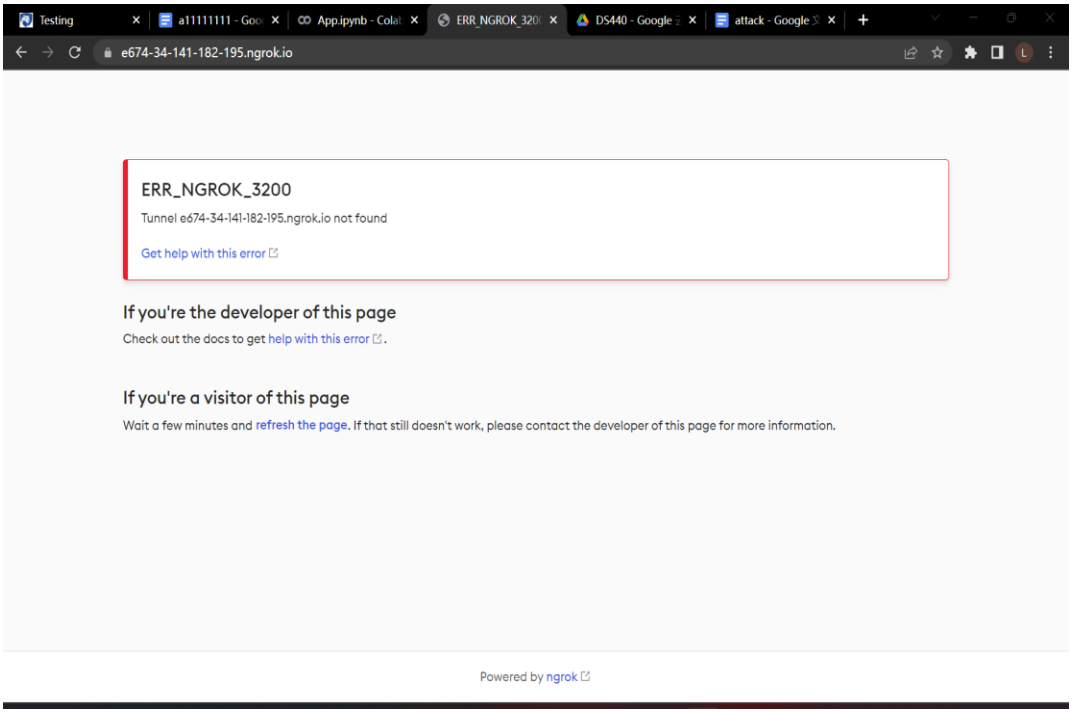

Figure 8 Failing Test Result

Testing 3

We send multiple requests to the website at the same time. We let 20 other users select random pictures and use it as input to the website; We intended to test whether our website and model can take such workload.

We see that the model takes a similar amount of time to generate the predictions as we only send one request. So, we think our model has the capacity to handle multiple requests.

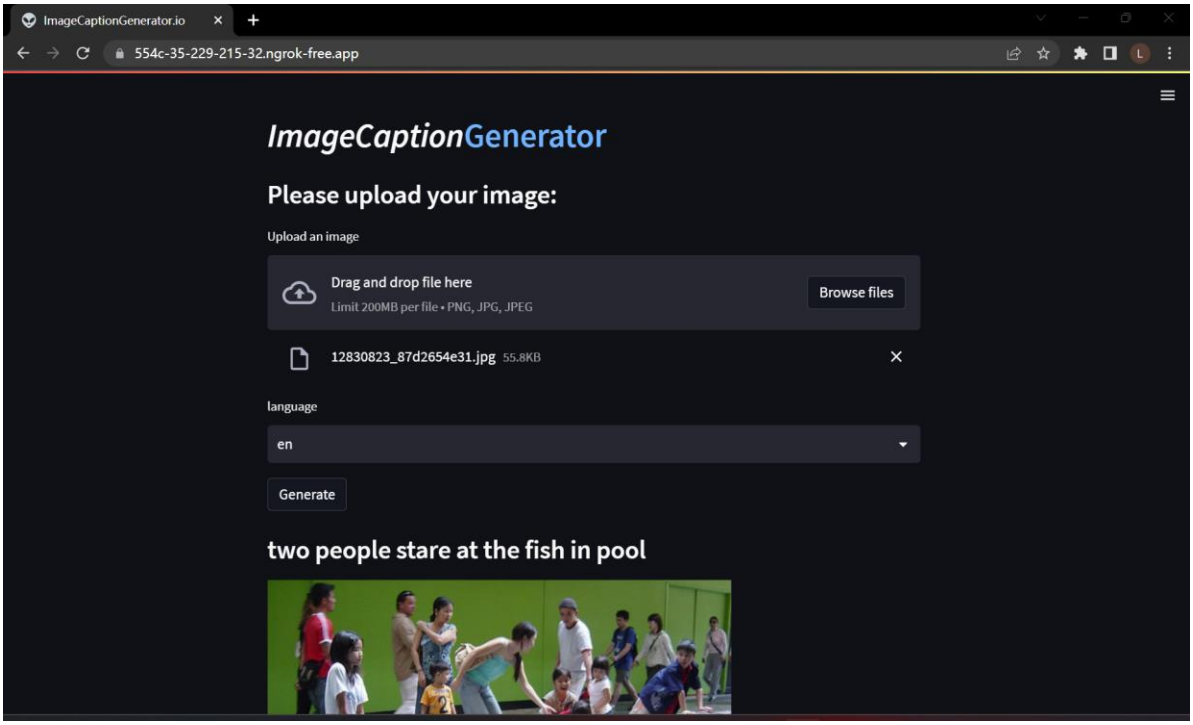

Figure 9 Successful Test Result in English

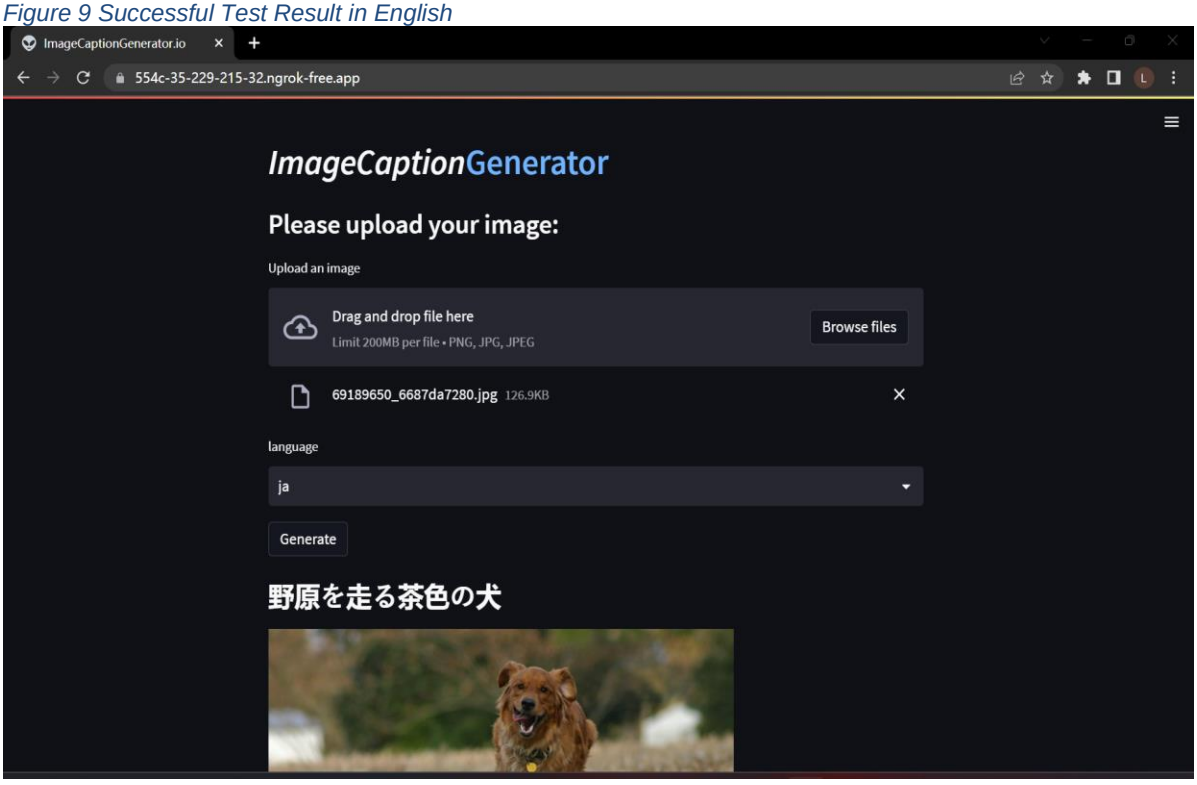

Figure 10 Successful Test Result in Japanese

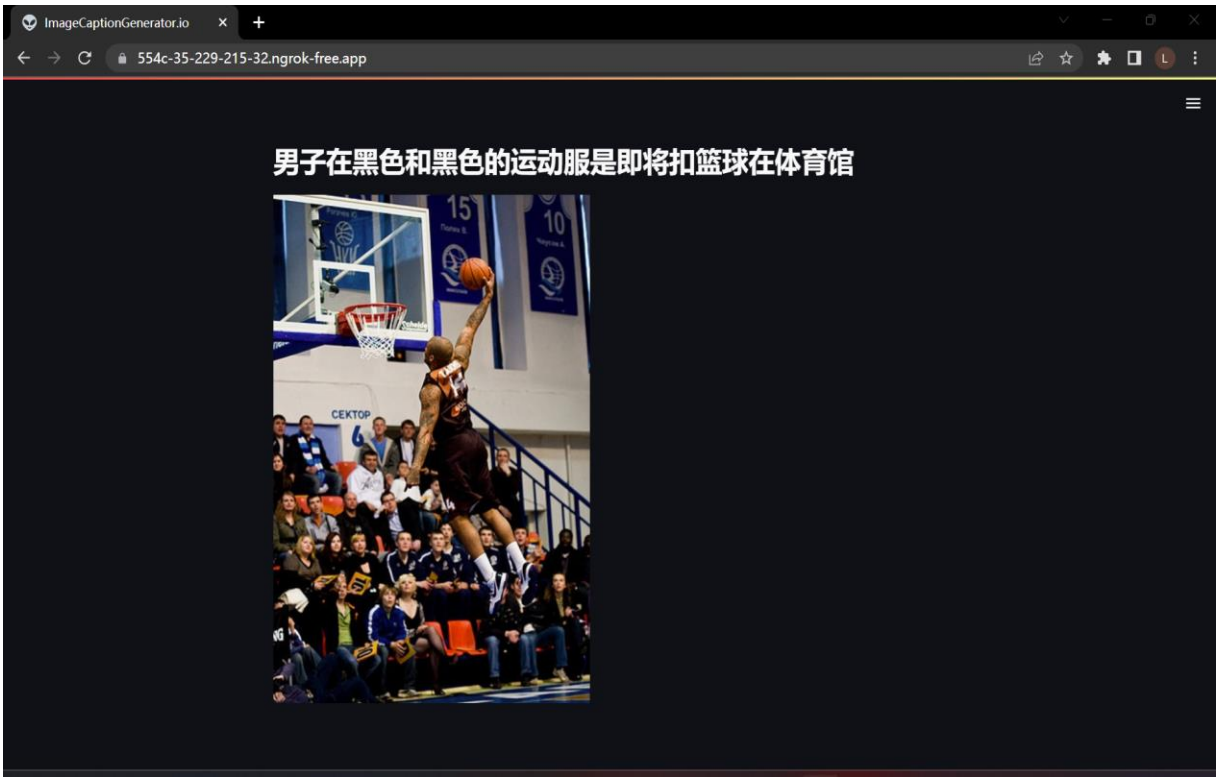

Figure 11 Successful Test Result in Chinese

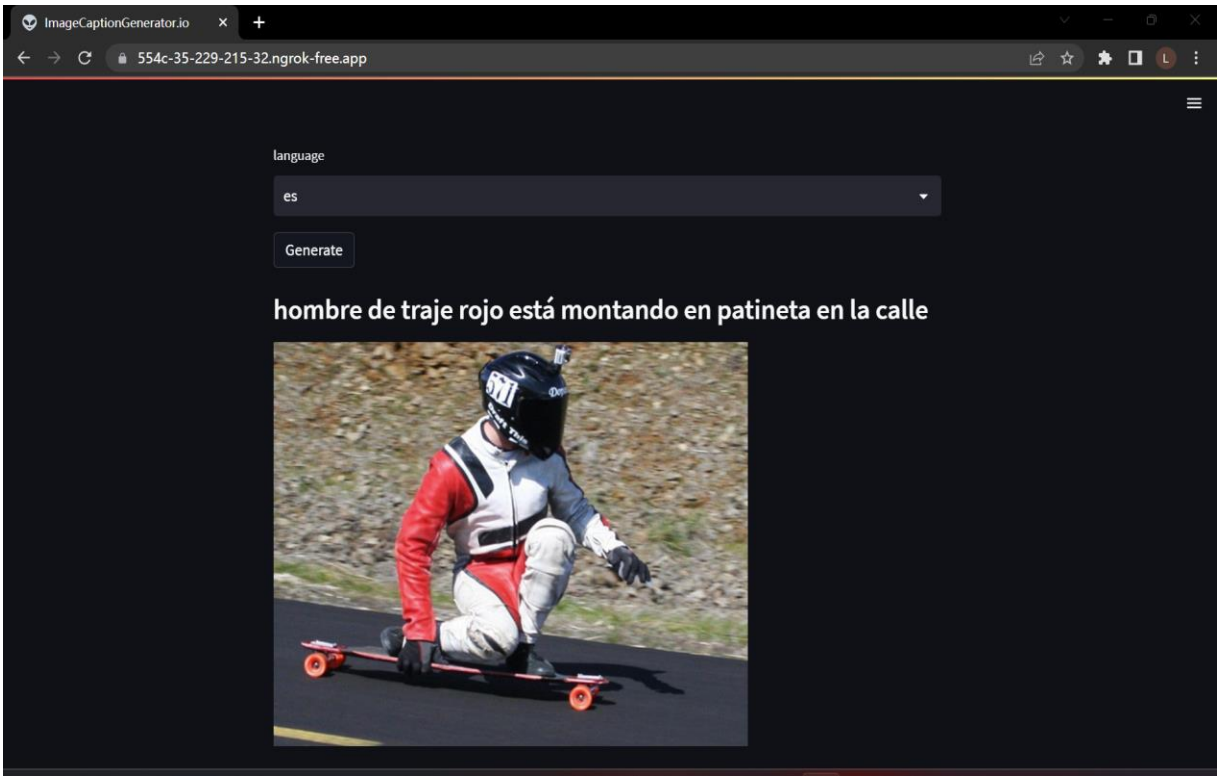

Figure 12 Successful Test Result in Spanish

Testing 4

Our model is trained based on the Flicker8k dataset, which are all pictures of common objects in our daily life. In this case, we want to test the accuracy of our web application for generating captions for uncommon objects, especially for text and charts. The result shows that our application successfully generates captions for the input image, but its results are not accurate at all. The following screenshot shows the result of our testing.

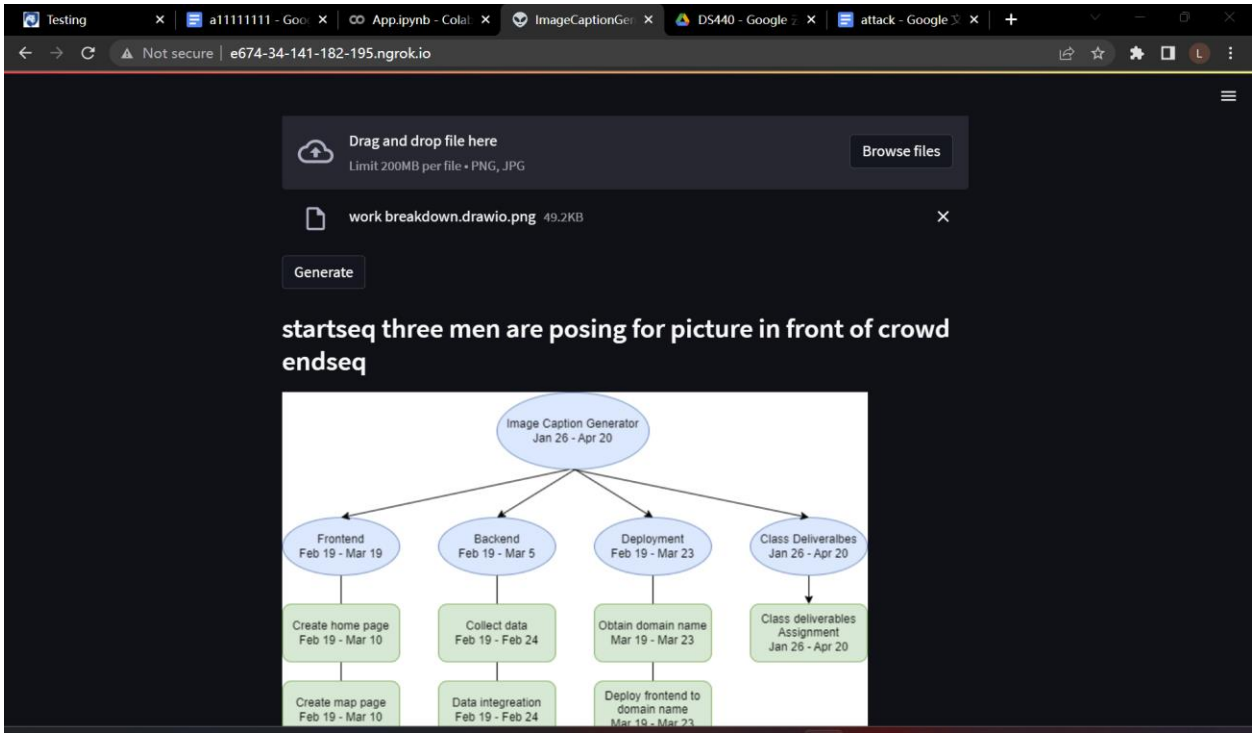

Figure 13 Failing Test Result

Conclusion

In conclusion, our project is an image caption generation web application that could accurately describe the content of an image in natural language. We used the Flickr8K dataset to train our model, VGG 16 model to extract features from the dataset, and LSTM to predict the caption with a translation function. To make our model accessible to end-users, we created a web application using Ngrok and Streamlit. Captions generated by our application were accurate to the image of common objects in our daily life.

Discussion

Changes in Development

The first change we made is the sequence of functions. Our original sequence was generating captions in English first, selecting the language, and finally translating the captions. However, when we tried to implement these functions, we found the application would run twice and the captions would also show twice, which is time-consuming and inefficient. As a result, we changed the sequence by setting selecting the language first.

Another change we made is the way to accomplish specific tasks. Our original plan was to use some advanced API to implement the function of translation. However, most of them are not free to use. In this case, we use some functions in Python to complete this task. It works almost as well as those APIs. We also changed some details like the design of the website to make it more user-friendly and the tool we used to deploy the application to reduce the cost.

Challenges

The most immediate challenge we faced was the lack of experience in front end and deployment. It cost us more time and attention than we expected. Our back end's implementation was also influenced by the choice for this part. It was a long trial-and-error process.

Another challenge we faced in our project was the limited budget. If we had enough funding, we could have incorporated more advanced APIs in our application to reduce the loading time and increase the accuracy of the translation. Additionally, we could train the model with more images without being restricted by GPU. We could also set more epochs and hyperparameter tunning steps to increase the accuracy of our model and reduce overfitting. We could also have conducted more rigorous testing, including automated tests to simulate heavy traffic on our application, without being restricted by usage limits.

Lesson Learned

One of the things we learned from this project was not all things would follow the plan even though the plan was perfect. We thought we have enough experience with the model and had assumed some problems when we made the plan. However, when we run epochs for the model, it already cost us two hours and then suddenly crashes because of the restriction of the GPU. We had to find a backup plan as soon as possible to keep going.

Another thing we learned was the power of Python. We never thought before that Python could be used to build the front end and make the deployment and how it could be so easy to get started. This experience inspired us to explore various libraries and modules in Python to accumulate knowledge and experience.

Future Work

More friendly for people with blind and visually impaired

Our application is very easy for normal people to use. However, for people with visual disabilities, we could add some assistive functions, such as screen readers and magnifiers. We can also add some text labels or provide audio descriptions for our application. Another method could be allow the virtual assistant like Siri to interact with our application. We could create interfaces that are not only accessible but also simple and pleasurable to use for those who are blind in this way.

Enhance the performance for uncommon images

Our model is trained by about 8000 images of daily common objects. The result shows that the model performs pretty well for similar images but has a poor performance for uncommon images like text. In this case, we could collect more images as our training data to improve the model. We could also investigate advanced techniques like transfer learning and multimodal learning in order to enhance our application to generate various types of images.

Another possible approach was to create customized models for various image categories. For example, a model could be trained specifically to generate captions for medical images. We could combine language models and domain-specific knowledge into the training process and provide captions with more accuracy and relevance by concentrating on specialized models.

Interact with other applications

As people depend on numerous applications to carry out tasks and manage their digital life, the interaction between each application is becoming more and more crucial. We could integrate API or develop a plug-in function for our application. These methods enable easy switching between several apps and the sharing of data between them. With this improvement, photo editing apps and social media platforms could automatically generate interesting captions for images uploaded by users. Additionally, e-commerce platforms could automatically generate detailed and informative captions for product images. Educational apps could also automatically generate captions for teaching material in a time-saving way.

Reference

Milhidaka (2021) *Milhidaka/chainer-image-caption: Image caption generator using chainer, python 3 and resnet feature version, GitHub*. Available at: https://github.com/milhidaka/chainer-image-caption (Accessed: April 18, 2023).

Vinyals, O. *et al.* (2015) *Show and tell: A neural image caption generator, – arXiv Vanity*. Available at: https://www.arxiv-vanity.com/papers/1411.4555/ (Accessed: April 18, 2023).

Image recognition market (2021) *Market Research Firm*. MarketsandMarkets. Available at: https://www.marketsandmarkets.com/PressReleases/image-recognition.asp (Accessed: April 18, 2023).

Image recognition market size, share and global market forecast to 2027 (2021) *MarketsandMarkets*. Available at: https://www.marketsandmarkets.com/Market-Reports/image-recognition-market-222404611.html (Accessed: April 18, 2023).