Software Requirements Specification
Version 1.0

EyeTalker

Theme: Eyes4Blind

Category: Unleashing the Power of AIML for Intelligent Solutions

Contents

1.1 Background and Necessity for the Application

‘Eye’ being one of the most important sensory organs when being a dysfunction to any human, it makes the person feel handicapped in all ways. Such people are always dependent on their close family or friends who guide them on their locomotion and anything that they wish to see.

In recent times, the prevalence of multimedia content on social media platforms and digital media has surged. The generation of image captions, which involves automatically producing descriptions or explanations for images, has garnered considerable attention due to its potential applications in various domains such as human-computer interaction, image retrieval, and accessibility for visually impaired individuals.

Despite the numerous proposed approaches for image captioning, effectively generating descriptive captions that truly capture the content and context of an image remains a formidable challenge. Furthermore, the majority of existing image captioning applications primarily focus on generating textual captions, disregarding the potential advantages of generating audio captions for individuals with visual impairments.

1.2 Proposed Solution

Image captioning coupled with audio is an ideal solution catering to the requirements of individuals who face challenges in visual skills. By employing a voice-based image caption generator, the application can generate both textual descriptions and corresponding voice output.

'EyeTalker' aims to develop an image captioning application using Artificial Intelligence and Machine Learning techniques to assist visually impaired individuals in perceiving visual content. By leveraging the power of machine learning algorithms, the application can generate textual descriptions of images, enabling visually impaired individuals to understand, and engage with the visual world.

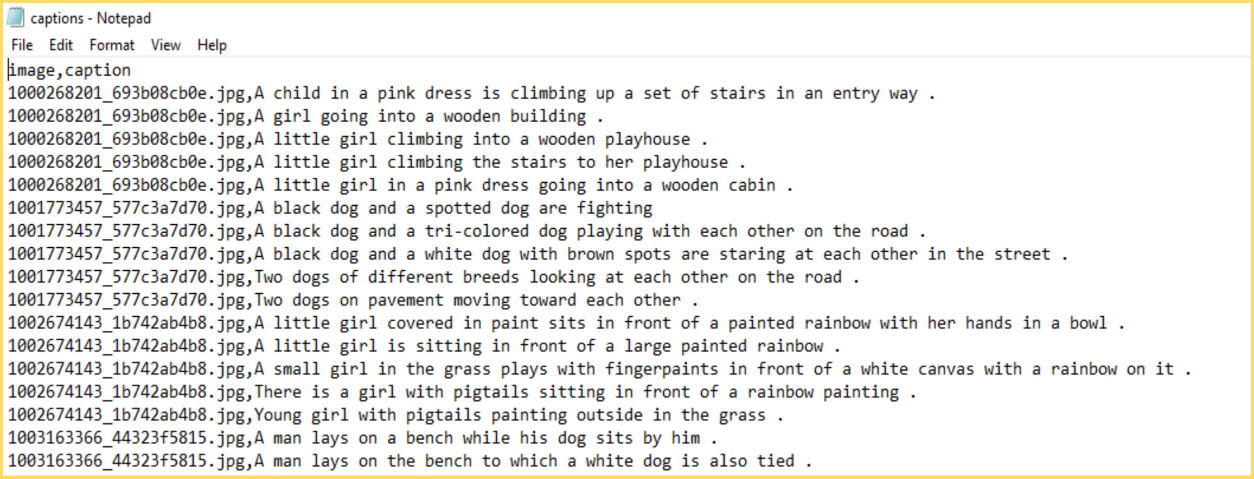
The proposed solution comprises a three-stage model consisting of feature extraction, encoder model, and decoder model. This model is designed to optimize the process of generating descriptive text captions and audio output from images.

1. **Feature Extraction:** The primary role of this model is to extract features from an image in order to train the provided data. During training, the image's features are taken as input to the model.

2. **Encoder:** The encoder plays a crucial role in the training process by converting the input images into captions. It produces a vector output that serves as an input to the decoder array.

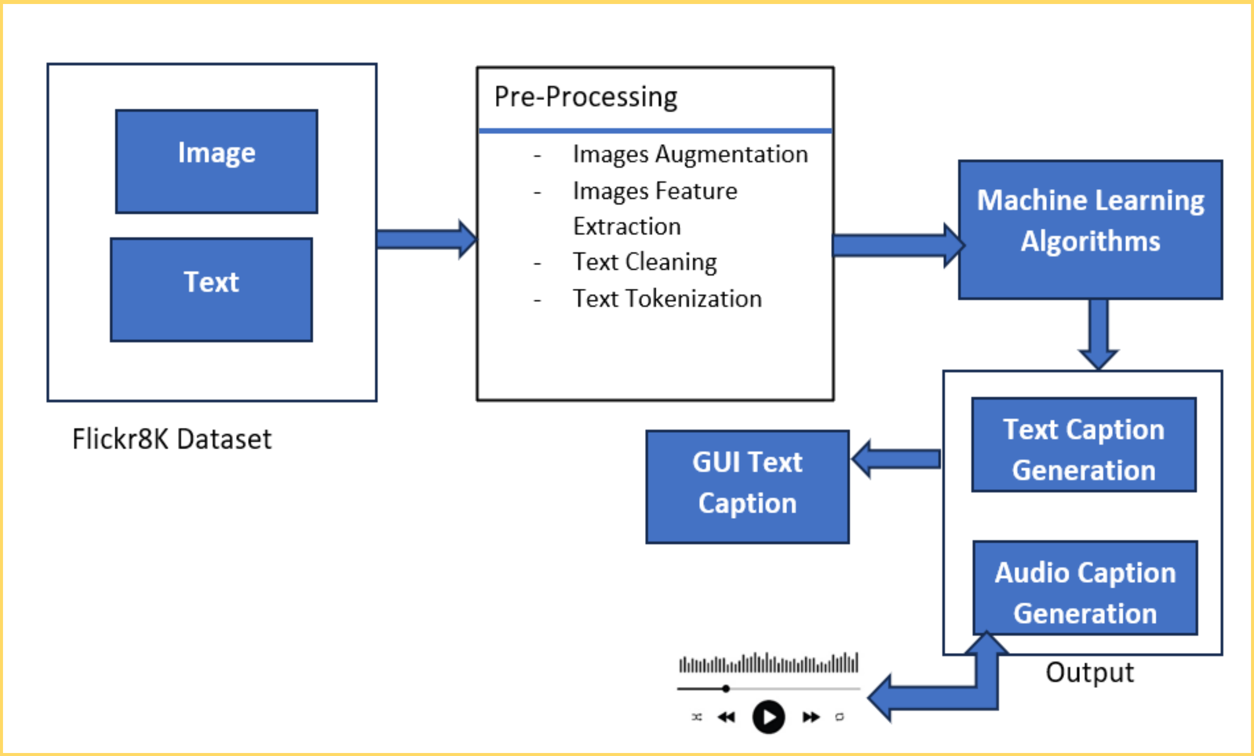
3. **Decoder:** The main responsibility of this model is to extract features, encode the data, and generate the predicted output, which includes words for a given image and the corresponding sentence up to that point. The decoder component takes input from the feature extraction and encodes both the vector outputs. The model takes the image and input application as parameters and generates predicted words as its output, representing the text captions and audio generated up to that point in time.

Hint: The Flickr8k dataset is downloaded from Kaggle for implementation purpose as follows:



```
image,caption
1000268201_693b08cb0e.jpg,A child in a pink dress is climbing up a set of stairs in an entry way .
1000268201_693b08cb0e.jpg,A girl going into a wooden building .
1000268201_693b08cb0e.jpg,A little girl climbing into a wooden playhouse .
1000268201_693b08cb0e.jpg,A little girl climbing the stairs to her playhouse .
1000268201_693b08cb0e.jpg,A little girl in a pink dress going into a wooden cabin .
1001773457_577c3a7d70.jpg,A black dog and a spotted dog are fighting
1001773457_577c3a7d70.jpg,A black dog and a tri-colored dog playing with each other on the road .
1001773457_577c3a7d70.jpg,A black dog and a white dog with brown spots are staring at each other in the street .
1001773457_577c3a7d70.jpg,Two dogs of different breeds looking at each other on the road .
1001773457_577c3a7d70.jpg,Two dogs on pavement moving toward each other .
1002674143_1b742ab4b8.jpg,A little girl covered in paint sits in front of a painted rainbow with her hands in a bowl .
1002674143_1b742ab4b8.jpg,A little girl is sitting in front of a large painted rainbow .
1002674143_1b742ab4b8.jpg,A small girl in the grass plays with fingerpaints in front of a white canvas with a rainbow on it .
1002674143_1b742ab4b8.jpg,There is a girl with pigtails sitting in front of a rainbow painting .
1002674143_1b742ab4b8.jpg,Young girl with pigtails painting outside in the grass .
1003163366_44323f5815.jpg,A man lays on a bench while his dog sits by him .
1003163366_44323f5815.jpg,A man lays on the bench to which a white dog is also tied .
```

The sample architecture can be as follows:



Sample Architecture of the Application

1.3 Purpose of the Document

The purpose of this document is to present a detailed description of the AI and ML based application interface for generating text and audio captions from images titled 'EyeTalker'.

This document explains the purpose and features of the AI and ML based application and the constraints under which it must operate. This document is intended for both stakeholders and developers of the application.

1.4 Scope of Project

The scope of the project is to incorporate a Text-to-Speech (TTS) engine to transform text-based captions from images. This TTS engine produces high-quality and natural-sounding audio descriptions that effectively convey the image's essence.

A desktop application featuring a Graphical User Interface (GUI) that enables users to upload images will generate both audio and text captions for the uploaded images.

1.5 Constraints

The project's effectiveness may be influenced by factors such as:

- **The Complexity of the Image**: Some images are more complex than others and this can make it difficult to generate accurate and informative captions. For example, an image of a busy street scene will be more complex than an image of a single object.

- **The Quality of the Image**: The quality of the image can also affect the accuracy of the captions. A low-quality image may be difficult to interpret and this can lead to inaccurate or incomplete captions.

- **The Context of the Image**: The context of the image can also be important, as it can help to provide additional information that is not explicitly visible in the image. For example, an image of a person holding a book may be more easily understood if it is known that the person is a student.

1.6 Functional Requirements

The primary objective of this application is to produce precise and varied captions that effectively represent the content and context of the image. Following are the functional requirements of the application:

i. **Data Collection and Data Cleaning –** Use the Flickr8k dataset downloaded from Kaggle which consists of 8,000 images and is suitable to build better models. Data cleaning plays a crucial role in handling textual data by ensuring its quality and consistency. In this project, perform specific data cleaning operations such as removal of vowels, conversion to lowercase, elimination of digits, trimming of trailing 's', and removal of special characters.

ii. **Extract Feature Vector –** This approach is commonly known as transfer learning, which allows to leverage pre-trained models that have already been trained on large datasets. Utilize these models to extract valuable features and incorporate them in your project.

iii. **Load Dataset for Training –**

Load the processed dataset for training the model. Convert the loaded images and text captions into a format compatible with the trained model being used for caption generation. This may involve encoding the images using techniques like Convolutional Neural Networks (CNNs) or transforming the text captions into numerical representations using methods like word embeddings.

iv. **Tokenizing Vocabulary –** To commence the process of generating text captions, it is necessary to train a Recurrent Neural Network (RNN) using appropriate data. In this phase, the RNN utilizes the vector inputs produced by the CNN model for decoding purposes.

v. **Create Data Generator –** To resemble it as supervised learning provide input-output pairs to the model, which then predicts actions based on previous predictions. This approach is like YouTube's recommendation application, which suggests videos based on user interests and previous viewing history. By using labeled datasets, the algorithm is trained to generate accurate predictions for new data.

vi. **Define the Model –** Define the model by using the Keras library from the functional API/Python library.

vii. **Train the Model –** Train the model by using various epochs value. The loss value should decrease with each iteration.

viii. **Test the Model –**

To test, the model provides the image path as input, which undergoes various functions to generate a description based on user input. Once the description is created, it proceeds to the audio creation function, which converts the generated caption into an audible output.

1.7 Non-Functional Requirements

There are several non-functional requirements that should be fulfilled by the application as follows:

1. **Maintainable:** The application's code should be well-structured, modular, and maintainable, allowing for future updates, bug fixes, and enhancements. It should adhere to coding best practices and documentation standards.

2. **Accessible:** The application should be designed with accessibility in mind, ensuring that users with disabilities can use it effectively. This includes providing support for screen readers, keyboard navigation, and alternative text for images.

3. **Secure:** The application should implement appropriate security measures to protect user data, including image uploads, and generated captions. It should follow best practices for data privacy, authentication, and authorization.

4. **Performance Efficiency:** The application should make efficient use of application resources, optimizing memory usage, and minimizing processing requirements.

5. **Usable**: The application should have an intuitive and user-friendly interface, allowing users to easily navigate, upload images, and access generated captions. It should provide clear instructions and feedback to guide users throughout the process.

6. **Reliability:** The application should be reliable and stable, minimizing crashes or unexpected behavior. It should handle errors gracefully and recover from failures gracefully.

7. **Internationalization:** The application should support multiple languages and locales, allowing users from different regions to use it effectively.

These are the bare minimum expectations from the project. <u>It is a must to implement the functional and non-functional requirements given in this SRS</u>. Once they are complete, you can use your own creativity and imagination to add more features if required.

1.8 Interface Requirements

1.8.1 Hardware

Intel Core i5 Processor or higher
8 GB RAM or higher
Color SVGA
500 GB Hard Disk space
Mouse
Keyboard

1.8.2 Software

Technologies to be used:

1. **Frontend**: HTML5 or any other frontend programming languages
2. **Data Store**: CSV/TXT
3. **Programming/IDE**: Python 3.x, Jupyter Notebook 7.x, Anaconda 3.x, Google Colab
4. **Libraries**: TensorFlow, Keras, Pytorch or other Python libraries, NLP APIs and libraries

1.9 Project Deliverables

You need to design and build the project and submit it along with a complete project report that includes:

- Problem Definition
- Design specifications
- Diagrams such as User Flow Diagram/User Journey Map
- The source code, including. ipynb files for Jupyter notebook and Google Colab, should be shared via GitHub. Appropriate access permissions should be granted to users to allow testing for Jupyter notebook and Google Colab.
- Dataset of .csv files
- Sample of 10-20 images used in the Project for testing as a zip file
- Project Installation Instructions
- .exe file, Jupyter Notebook/Google Collab for desktop application testing
- Link of GitHub for accessing the uploaded project code. (Link should have public access.)

The consolidated project must be submitted on GitHub with a ReadMe.doc file listing assumptions (if any) made at your end. Ensure that you provide the GitHub URL where the project has been uploaded for sharing. The repository on GitHub should have public access. Documentation is a very important part of the project; hence, all crucial aspects of the project must be documented properly.

Specify the location of the folder/zip containing the sample images as the input. This allows the code to access and process the images during the caption generation process. The folder structure and naming conventions must align with the instructions provided in the documentation for testing the images. Specify the path if the image testing is within the machine learning code itself.

Provide the executable (.exe) file of the desktop application for testing purposes. The application should allow image uploads through the Graphical User Interface (GUI) and enable us to evaluate the generated output. In addition, you must submit a video clip showing the actual working of the application.

Over and above the given specifications, you can apply your creativity and logic to improve the application.

*~~~ End of Document ~~~*