Software Requirements Specification

for

Image Captioning Using Deep Learning

Version 1.0 approved

Prepared by Chinmay Paralkar, Royston Pinto, Kajal Sapkal

<organization>

<date created>

Table of Contents

Table of Contents ii

Revision History ii

1. Introduction 1

1.1 Purpose 1

1.2 Document Conventions 1

1.3 Intended Audience and Reading Suggestions 1

1.4 Product Scope 1

1.5 References 1

2. Overall Description 2

2.1 Product Perspective 2

2.2 Product Functions 2

2.3 User Classes and Characteristics 2

2.4 Operating Environment 2

2.5 Design and Implementation Constraints 2

2.6 User Documentation 2

2.7 Assumptions and Dependencies 3

3. External Interface Requirements 3

3.1 User Interfaces 3

3.2 Hardware Interfaces 3

3.3 Software Interfaces 3

3.4 Communications Interfaces 3

4. System Features 4

4.1 System Feature 1 4

4.2 System Feature 2 (and so on) 4

5. Other Nonfunctional Requirements 4

5.1 Performance Requirements 4

5.2 Safety Requirements 5

5.3 Security Requirements 5

5.4 Software Quality Attributes 5

5.5 Business Rules 5

6. Other Requirements 5

Appendix A: Glossary 5

Appendix B: Analysis Models 5

Appendix C: To Be Determined List 6

# Introduction

## Purpose

The purpose of this document is to give a detailed description of the requirements for the Image Captioning software. It will illustrate the purpose and complete declaration for the development of system. It will also explain system constraints, interface and interactions with other external applications. This document is primarily intended to be proposed for approval and a reference for developing the first version of the system for the development team.

## Document Conventions

PC- Personal Computer

LSTM- Long Short Term Memory

CRF- Conditional Random Field

## Intended Audience and Reading Suggestions

This project for the Image Captioning Using deep learning system and it is restricted within the college premises. This has been implemented under the guidance of college professors. This project is useful for the Image Captioning. If you load the any image its generate the caption for it.

## Product Scope

This project is used to generate a caption for a given image. It uses InceptionV3 model in Keras which is pre-trained for Image Recognition on dataset imagenet. For image captioning we are using Flickr-8k dataset containing 8000+ images.

## References

https://cs.Stanford.edu/people/karpathy/cvpr2015.pdf

https://arxiv.org/abs/1411.4555

https://arxiv.org/abs/1703.09137

https://arxiv.org/abs/1708.02043

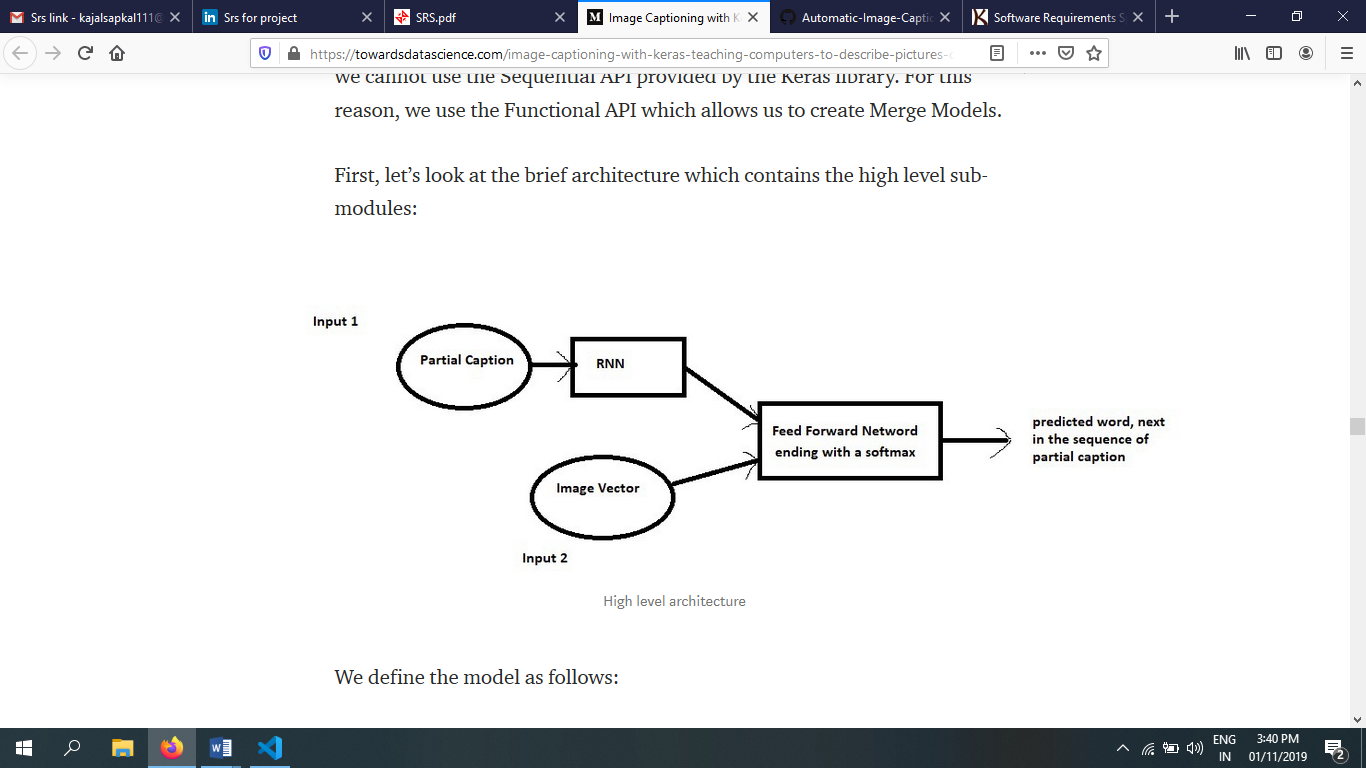
https://machinelearningmastery.com/develop-a-deep-learning-caption-generation-model-in-python/

https://www.youtube.com/watch?v=yk6XDFm3J2c

https://www.appliedaicourse.com/

# Overall Description

## Product Perspective



**Hardware :** System Model : HP Laptop 15-bs0xx. PC credentials : Processor Intel(R) Core(TM) i3-7020U CPU @ 2.30GHz, 2300 Mhz, 2 Core(s), 4 Logical Processor(s), 64GB RAM.

**Software :** Visual Studio Code IDE for programming, Python

Data Structures, Libraries called Pandas, Numpy, Skilearn, Keras,

CNN model InceptionV3. We use Flickr-8k dataset provided by the

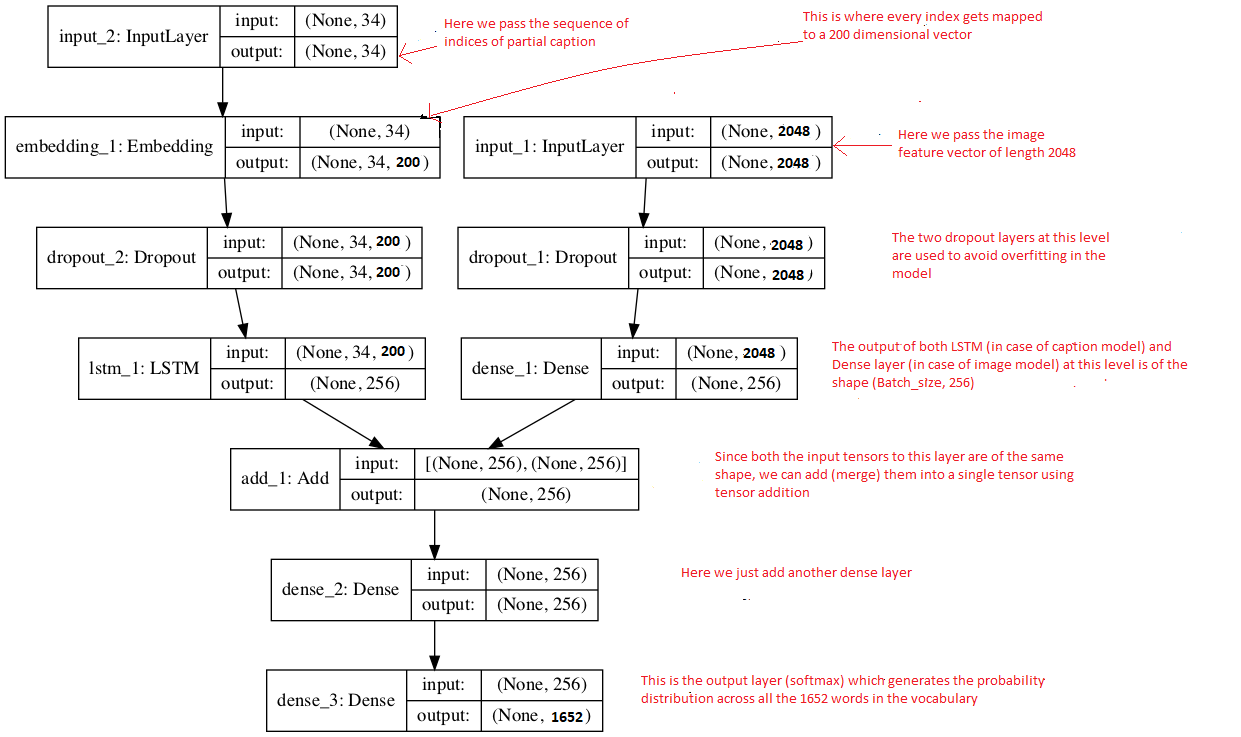
reference.

## Product Functions

<Summarize the major functions the product must perform or must let the user perform. Details will be provided in Section 3, so only a high level summary (such as a bullet list) is needed here. Organize the functions to make them understandable to any reader of the SRS. A picture of the major groups of related requirements and how they relate, such as a top level data flow diagram or object class diagram, is often effective.>

## User Classes and Characteristics

The below plot helps to visualize the structure of the network and better understand the two streams of input:



The text in red on the right side are the comments provided for you to map your understanding of the data preparation to model architecture.

The **LSTM (Long Short Term Memory)** layer is nothing but a specialized Recurrent Neural Network to process the sequence input (partial captions in our case). To read more about LSTM, click [**here**](http://colah.github.io/posts/2015-08-Understanding-LSTMs/#_blank).

If you have followed the previous section, I think reading these comments should help you to understand the model architecture in a straight forward manner.

Recall that we had created an embedding matrix from a pre-trained Glove model which we need to include in the model before starting the training:

model.layers[2].set\_weights([embedding\_matrix])

model.layers[2].trainable = False

Notice that since we are using a pre-trained embedding layer, we need to **freeze** it (trainable = False), before training the model, so that it does not get updated during the backpropagation.

Finally we compile the model using the adam optimizer

model.compile(loss=’categorical\_crossentropy’, optimizer=’adam’)

Finally the weights of the model will be updated through backpropagation algorithm and the model will learn to output a word, given an image feature vector and a partial caption. So in summary, we have:

Input\_1 -> Partial Caption

Input\_2 -> Image feature vector

Output -> An appropriate word, next in the sequence of partial caption provided in the input\_1 (or in probability terms we say **conditioned** on image vector and the partial caption)

## Operating Environment

The project is working on Windows 10 in Visual Studio Code IDE. The code is written in Python language with Keras modules.

## Design and Implementation Constraints

<Describe any items or issues that will limit the options available to the developers. These might include: corporate or regulatory policies; hardware limitations (timing requirements, memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; parallel operations; language requirements; communications protocols; security considerations; design conventions or programming standards (for example, if the customer’s organization will be responsible for maintaining the delivered software).>

## User Documentation

<List the user documentation components (such as user manuals, on-line help, and tutorials) that will be delivered along with the software. Identify any known user documentation delivery formats or standards.>

## Assumptions and Dependencies

Let us assume that this is an Image captioning using deep learning system and it is used in the following application:

This system generate automated caption for specific image.

# External Interface Requirements

## User Interfaces

Software used are Visual Studio Code, CV2 for using on functions of images.

## Hardware Interfaces

Operating System: Windows 10, Processor Intel(R) Core(TM) i3-7020U CPU @ 2.30GHz, 2300 Mhz.

## Software Interfaces

Visual Studio Code IDE for programming, Python

Data Structures, Libraries called Pandas, Numpy, Skilearn, Keras,

CNN model InceptionV3. We use Flickr-8k dataset provided by the

reference.

## Communications Interfaces

This project supports all types of Images. We are using deep learning approach for the image generation.

# System Features

<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>

## System Feature 1

<Don’t really say “System Feature 1.” State the feature name in just a few words.>

4.1.1 Description and Priority

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>

4.1.2 Stimulus/Response Sequences

<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

4.1.3 Functional Requirements

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use “TBD” as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

REQ-1:

REQ-2:

## System Feature 2 (and so on)

# Other Nonfunctional Requirements

## Performance Requirements

<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>

## Safety Requirements

If there is extensive damage to a wide portion of the database due to catastrophic failure, such as a disk crash, the recovery method restores a past copy of the database that was backed up to archival storage and reconstructs a more current state by reapplying or redoing the operations of committed transactions from the backed up log, up to the time of failure.

## Security Requirements

Security systems need database storage just like many other applications. However, the special requirements of the security market mean that vendors must choose their database partner carefully.

## Software Quality Attributes

* **AVAILABILITY:** The Specific Image should be available for users.
* **CORRECTNESS:** The image should give the relevant correct caption for it.
* **MAINTAINABILITY:** Maintain the correct images for it.
* **USABILITY:** The image should satisfy a maximum number of users needs.

## Business Rules

<List any operating principles about the product, such as which individuals or roles can perform which functions under specific circumstances. These are not functional requirements in themselves, but they may imply certain functional requirements to enforce the rules.>

# Other Requirements

<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>

Appendix A: Glossary

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>

Appendix B: Analysis Models

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>

Appendix C: To Be Determined List

<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>