

# INTELLIGENT MODEL FOR IMAGE-BASED RECOMMENDATION SYSTEM

by

Prerna Prateek

August, 2016

Director of Thesis: Dr. Nasseh Tabrizi

Major Department: Computer Science

## Abstract

Online shopping has developed in parallel with the Internet, and Recommendation Systems have played a pivotal role in its growth. The recommendations are usually provided in two ways: Content-based Filtering and Collaborative Filtering. Both forms of recommendations face the problem of Cold-Start due to an initial lack of information. To overcome this issue, Image-based Recommendation Systems are introduced in order to allow the users to locate products based on similarity of images when purchasing products in categories such as: clothes, shoes, home-decor, kitchen and dining utilities, jewelry, and accessories by mostly viewing images. In this thesis, a Hybrid Model of displaying similar images to that of the product being viewed was developed using Deep Features and Description-based Models. The Hybrid Model displayed a set composed of all images that belong to both Deep Features and Description-based Models.

Implementation and comparison of results were performed on 100,000 images of SBU  
Captioned Photo Dataset.



**INTELLIGENT MODEL FOR IMAGE-BASED RECOMMENDATION SYSTEM**

A Thesis

Presented To

The Faculty of the Department of Computer Science

East Carolina University

In Partial Fulfillment

of the Requirements for the Degree

**MASTER of SCIENCE in SOFTWARE ENGINEERING**

by

Prerna Prateek

August, 2016

© Prerna Prateek, 2016

**INTELLIGENT MODEL FOR IMAGE-BASED RECOMMENDATION SYSTEM**

by

Prerna Prateek

APPROVED BY:

DIRECTOR OF THESIS: \_\_\_\_\_  
Nassem Tabrizi, PhD

COMMITTEE MEMBER: \_\_\_\_\_  
Venkat N. Gudivada, PhD

COMMITTEE MEMBER: \_\_\_\_\_  
Sergiy Vilkomir, PhD

CHAIR OF THE DEPARTMENT OF COMPUTER SCIENCE:

\_\_\_\_\_  
Venkat N. Gudivada, PhD

DEAN OF THE GRADUATE SCHOOL:

\_\_\_\_\_  
Paul J. Gemperline, PhD

## **DEDICATION**

I dedicate my thesis to my husband, parents, mother-in-law, and father-in-law for always helping me persevere. I would also like to thank my loving siblings for inspiring me, as well as Dr. Nasseh Tabrizi for giving me the wonderful opportunity, becoming my mentor, and also extending help during difficult situations.

## **ACKNOWLEDGMENTS**

I am extremely grateful to my advisor, Dr. Nasseh Tabrizi, for the continued support, patience, and guidance for my thesis and related research. I feel very fortunate to have worked under such an immensely knowledgeable and intelligent person. I am thankful to him for showing me the correct path and for always giving me the best advice.

## TABLE OF CONTENTS

DEDICATION .....	iv
ACKNOWLEDGMENTS .....	v
LIST OF TABLES.....	vii
LIST OF FIGURES .....	viii
CHAPTER 1: INTRODUCTION .....	1
CHAPTER 2: IMAGE RETRIEVAL MODELS.....	5
2.1 Description-based Image Retrieval.....	5
2.2 Deep Features-based Image Retrieval.....	6
2.3 Hybrid Model for Image Retrieval.....	10
CHAPTER 3: THE EXPERIMENT .....	11
CHAPTER 4: AUTOMATIC TEXTUAL DESCRIPTION .....	20
CHAPTER 5: RESULTS .....	30
CHAPTER 6: CONCLUSION AND FUTURE WORKS .....	34
REFERENCES .....	35
Appendix A: Output Images from each models.....	38
Appendix B: Code for Experiment .....	58
Appendix C: Collection of verbs for Man-Dog combination .....	63
Appendix D: PASCAL 50 Sentences.....	64

## LIST OF TABLES

Table 1. Layers of SuperVision's AlexNet.....	9
Table 2. Actions and possible interactions of a person with bike .....	25
Table 3. The results of Automatic Textual Description [4]. .....	28
Table 4. Object Detection using Deformable Part Model.....	30
Table 5. Analysis of image #5 .....	32

## LIST OF FIGURES

Figure 1. Neural Networks of SuperVision's Alexnet.....	8
Figure 2. Architecture of Graphlab (source: Turi, Inc.).....	12
Figure 3. Diagramatic representation of image description system .....	22
Figure 4. Deformable Part Model of man riding bike.....	24
Figure 5. Steps of method for image description .....	27

## CHAPTER 1: INTRODUCTION

Computer Vision has provided solutions to many existing challenges and can be used in improving Recommendation Systems. The interest of this thesis, in regard to learning and contributing to the computer vision domain of artificial intelligence, was due to the fact that the most advanced machines and computers struggle in vision aspect. Even though autonomous driving cars have been developed and have been undergoing continuous modification, they are still far behind human ability of vision and recognition. Developers strive to achieve an autonomous driving car that can match a human's vision performance in order to distinguish between what should be avoided (say a "huge rock") or what could be run over (say a "crumpled paper"). More and more types of advanced cameras have been developed, but facilitating visually impaired individuals with vision, or aiding them to know what is in front of them, has yet to be accomplished. Even if Unmanned Aerial Vehicles have been flying over all types of geographical regions to capture videos, any natural calamity recorded cannot automatically forewarn people in the areas affected. It requires humans to watch captured videos and alert people to take precautionary measures. Advanced surveillance cameras capture videos, but it cannot prevent anticipated accidents, which can happen in everyday life. For instance, if a child comes into contact with an electrical socket and receives a shock, any person next to him/her will try to protect the child, as human vision is interpreted by brain; machine vision lacks this important characteristic and is not intelligent enough to rescue an individual. The development needs to be inspired from the brain's functionality. Humans are extremely efficient at the task of describing images' content; with nothing more than a gaze, the human brain can compose an entire story of the image. The brain understands an image and has evolved to find subtle similarities between two or more images. It can easily find most similar images from a set. A Neural Network mimics the brain, and it

operates like non-linear parallel information-processing systems, which rapidly perform computations such as image recognition.

Recommendation systems are nothing but an automated form of a “sales associate” who not only shows what product a customer is purchasing, but also the related ones which he/she could purchase. This ability to recommend personalized content, based on past behavior, is incredible, as it brings customers delight and provides them a reason to keep returning to the website.

A very common approach to the design of recommender systems is collaborative filtering [1], which has its implementation based on collecting and analyzing a large amount of information on users’ behaviors, activities, or preferences and predicting what users will like based on their similarity to other users. Another common approach when designing recommender systems is content-based filtering [2], whose methods are based on a description of the item and a profile of the user’s preference. The underlying methodology is that algorithms try to recommend items that are similar to those a user liked/selected in the past (or is examining in the present).

There is a concerning issue in functioning of both collaborative filtering and content-based filtering when the system cannot draw correct inferences for users or items about which it has not yet gathered sufficient information. This is the case when past behavior is unavailable, given a lack of the customer’s previous purchase history, and is known as cold-start problem [3].

Capturing an image is becoming easier, as images on the Internet have been increasing tremendously. Now, it is very easy to find images for vast collection of commodities within a market. By taking advantages from the availability of many images, and utilizing the Deep Features, and Captions of the image (or the suggested description generation method [4], when

captions are unavailable), we developed a model which can be used to resolve potential Cold-Start problem.

Deep Convolutional Neural Networks [5] are used to extract features from the images, called Deep Features, to train Nearest Neighbor Models for retrieving images. In some cases, images do not have descriptions, so images are labelled with descriptions so that the Term Frequency-Inverse Document Frequency (TF-IDF) of Descriptions can be used to train Nearest Neighbor Models. Similar images can be obtained by querying with the images' description and the images' Deep Features, and common images generated from both processes will then be displayed. This way allows for advantages from images retrieved from both methods to be available.

The contributions of this thesis are:

- Implementation and evaluation of image retrieval using Deep Features
- Implementation and evaluation of image retrieval using descriptions' TF-IDF
- Development of a Hybrid Model
- Analysis on 100,000 images from SBU Captioned Photo Dataset
- Textual description of images for correcting wrong image descriptions.

Underlying methodologies are discussed in Chapter 2.

Chapter 3 discusses the experimental steps taken by the Hybrid Model for generating similar images with the related works in field of each steps.

During our analysis, we encountered a case where human generated descriptions were improper for certain images, which caused the method to not perform as desired. In Chapter 4, a method for generation of image description for all types of images is explored.

In addition, Chapter 5 will discuss the comparisons of said generated results.

Finally, Chapter 6 summarizes the main point of the thesis and discusses the scope for future research and implementation.

## CHAPTER 2: IMAGE RETRIEVAL MODELS

In this chapter, the underlying methodology of all the models is elaborately discussed. The models which were implemented and evaluated are:

- Description-based Model
- Deep Features Model

The model which was developed and evaluated during this study is:

- Hybrid Model

### 2.1 Description-based Image Retrieval

The method uses descriptions associated with images for image retrieval, so the model created for retrieving images is named as Description-based Model.

In the implementation of image retrieval using description, a powerful technique which can be used to figure out what a description is about, was required. For information retrieval, TF-IDF weighing came out as a plausible solution. TF-IDF stands for term frequency-inverse document frequency, and the TF-IDF weight is a statistical measure which evaluates how important a word is to a document in a collection or corpus. Term frequency measures how frequently a certain word occurs in a document, and inverse document frequency measures how special a certain word is in a collection of documents.

$$\text{TF-IDF}(w,d) = \text{tf}(w,d) * \log(N/f(w))$$

where “ $tf(w,d)$ ” is the number of times word “ $w$ ” appeared in document “ $d$ ”, “ $f(w)$ ” is the number of documents word “ $w$ ” appeared in, “ $N$ ” is the number of documents, and we use the natural logarithm.

In the algorithm for implementing Description-based Model, Nearest Neighbors approach was used. This means that using the description of any image and finding it's similarity to other descriptions have to be retrieved. In order to use the Nearest Neighbors method, a way to compare descriptions was needed. This was done by representing descriptions as vectors in a high-dimensional space and then quantifying how similar two descriptions are by calculating a distance metric. In making vectors for distances TF-IDF features were used.

Once a description is queried, a new TF-IDF unit vector is obtained, which allows us to easily calculate cosine similarities. Cosine similarity is the cosine of the angle between two vectors. A pair of vectors pointing in the same direction will have a cosine similarity of 1, while vectors pointing in opposite directions will have a cosine similarity of 0. By representing the entire collection of description as a matrix with  $n$  rows (each row representing a vector), we can do a dot product with the queried description vector, and we will get an  $n$ -dimensional vector of cosine similarities. Looking for the largest values in the cosine similarity vector helps to find Nearest Neighbors, and aggregating the tags of these Nearest Neighbors based on rank determines which ones to suggest.

## 2.2 Deep Features-based Image Retrieval

Artificial intelligence is immensely powerful, and there has been substantial growth from the earliest research in computer vision, in the 1950s. Neural networks and Deep Learning empowered it like never before and have taken this technology to a different level. The aim of

Computer Vision is to imitate the functionality of human eye and brain components responsible for a humans' sense of sight. Lots of advancements have been seen in the field of computer vision in the past 5 years. One of the most stated advancements is the Convolution Neural Networks [6], done in 2012 for ImageNet Challenge(<http://www.image-net.org/challenges/LSVRC/2012/>) [5] by SuperVision's AlexNet.

Object Detection is considered to be the most basic application of computer vision, and due to several factors such as variations in viewpoints, occlusions, change in illumination, and background clutter, it is often very challenging. Convolutional Neural Networks are the most superior method as of today for Object Recognition. A Convolutional Neural Network typically consists of 3 types of layers: Convolution Layer, Pooling Layer, and Fully Connected Layer. In Convolutional Neural Network, the image is broken into overlapping image tiles. It is very similar to the sliding window search in which the sliding window is passed over the entire original image with overlap. The output from sliding window is saved as separate tiny image tiles. Then, every single image tile is fed into a Neural Network to see what it contains. For all the image tiles from the same original image, weights on Neural Network is kept the same, i.e. In other words, every image tile is treated equally. If something interesting appears in any given tile, then that tile is marked. In order to prevent arrangement loss of original tiles, results from processing each tile are saved in a grid, in the same arrangement as that of the original image. In other words, it started with a large image and ends with a slightly smaller array, with information about interesting sections of the original image. To reduce array size further, down samples are created by using an algorithm called max pooling. The concept here involves locating something interesting. When found in any of the input tiles that makes up each grid rectangle, it just keeps the most interesting

bit. This is a method that reduces the size of the array while keeping its essence. Then, output is passed through fully connected Neural Network to find if it is match or not.

Figure 1. is an illustration of architecture of SuperVision's Convolutional Neural Networks, called AlexNet.

## AlexNet (Krizhevsky et al. 2012)

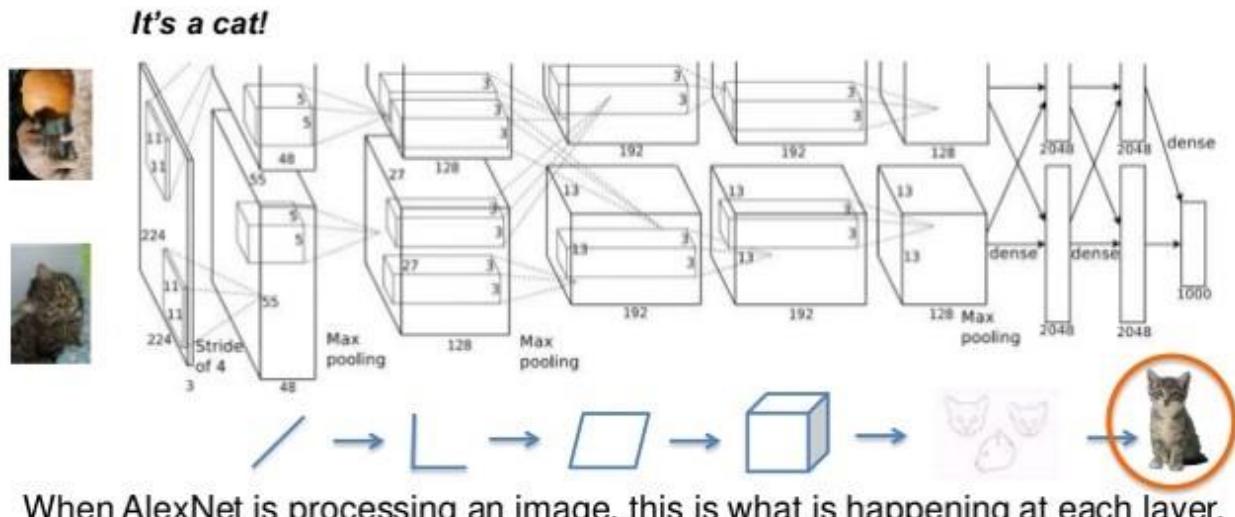


Figure 1. Neural Networks of SuperVision's Alexnet

Convolutional Neural Network [5] was very specific for ImageNet Challenge. Deep Features of images were extracted using Convolutional Neural Network and Nearest Neighbors Model for identifying images based on similarity was created.

Deep Convolutional Neural Network, in Figure 2 displays later layers for specific tasks, and work in [7] used Deep Features extracted from layers of SuperVision's AlexNet [5] for new visual recognition task. ImageNet pre-trained Model in Graphlab was loaded, and features created in its last layer were implemented into a simple classifier to make predictions. The Model was named `deep_learning_model`.

Details about different layers are described in Table 1:

*Table 1. Layers of SuperVision's AlexNet*

<b>Layer 0: Input image</b>
<ul style="list-style-type: none"> <li>• Size: 227 x 227 x 3</li> </ul>
<b>Layer 1: Convolution with 96 filters, size 11x11, stride 4, padding 0</b>
<ul style="list-style-type: none"> <li>• Size: 55 x 55 x 96</li> <li>• <math>(227-11)/4 + 1 = 55</math> is the size of the outcome</li> <li>• 96 is depth because 1 set denotes 1 filter and there are 96 filters</li> </ul>
<b>Layer 2: Max-Pooling with 3x3 filter, stride 2</b>
<ul style="list-style-type: none"> <li>• Size: 27 x 27 x 96</li> <li>• <math>(55 - 3)/2 + 1 = 27</math> is size of outcome</li> <li>• depth is same as before, i.e. 96 because pooling is done independently on each layer</li> </ul>
<b>Layer 3: Convolution with 256 filters, size 5x5, stride 1, padding 2</b>
<ul style="list-style-type: none"> <li>• Size: 27 x 27 x 256</li> <li>• Because of padding of <math>(5-1)/2=2</math>, the original size is restored</li> <li>• 256 is depth because of 256 filters</li> </ul>
<b>Layer 4: Max-Pooling with 3x3 filter, stride 2</b>
<ul style="list-style-type: none"> <li>• Size: 13 x 13 x 256</li> <li>• <math>(27 - 3)/2 + 1 = 13</math> is size of outcome</li> <li>• Depth is same as before, i.e. 256 because pooling is done independently on each layer</li> </ul>
<b>Layer 5: Convolution with 384 filters, size 3x3, stride 1, padding 1</b>
<ul style="list-style-type: none"> <li>• Size: 13 x 13 x 384</li> <li>• Because of padding of <math>(3-1)/2=1</math>, the original size is restored</li> <li>• 384 is depth because of 384 filters</li> </ul>
<b>Layer 6: Convolution with 384 filters, size 3x3, stride 1, padding 1</b>
<ul style="list-style-type: none"> <li>• Size: 13 x 13 x 384</li> <li>• Because of padding of <math>(3-1)/2=1</math>, the original size is restored</li> <li>• 384 is depth because of 384 filters</li> </ul>
<b>Layer 7: Convolution with 256 filters, size 3x3, stride 1, padding 1</b>
<ul style="list-style-type: none"> <li>• Size: 13 x 13 x 256</li> <li>• Because of padding of <math>(3-1)/2=1</math>, the original size is restored</li> <li>• 256 is depth because of 256 filters</li> </ul>
<b>Layer 8: Max-Pooling with 3x3 filter, stride 2</b>
<ul style="list-style-type: none"> <li>• Size: 6 x 6 x 256</li> <li>• <math>(13 - 3)/2 + 1 = 6</math> is size of outcome</li> <li>• Depth is same as before, i.e. 256 because pooling is done independently on each layer</li> </ul>
<b>Layer 9: Fully Connected with 4096 neuron</b>
<ul style="list-style-type: none"> <li>• In this later, each of the <math>6 \times 6 \times 256 = 9216</math> pixels are fed into each of the 4096 neurons and weights determined by back-propagation.</li> </ul>
<b>Layer 10: Fully Connected with 4096 neuron</b>
<ul style="list-style-type: none"> <li>• Similar to layer #9</li> </ul>
<b>Layer 11: Fully Connected with 1000 neurons</b>
<ul style="list-style-type: none"> <li>• This is the last layer and has 1000 neurons because IMAGENET data has 1000 categories to be predicted.</li> </ul>

## 2.3 Hybrid Model for Image Retrieval

The Hybrid Model uses both images and descriptions, i.e. intersection (overlapping images) of both retrieval methods.

The k-Nearest Neighbor classifier had been the simplest image classification algorithm, as it relies on the distance between feature vectors. The k-NN algorithm classifies unknown data points by finding the most common class among the k-closest examples. Each data point in the k closest examples give a vote, and the category with the most votes is eventually taken.

The Nearest Neighbor Model trained on features extracted from Deep Features and Nearest Neighbor Model trained in TF-IDF generated 1000 images (this number was chosen, considering SuperVision's AlexNet had 1000 neurons in the fully connected last layer).

As both were retrieving similar images to the same queried image, a considerable number of images belonging to both models were present. The Hybrid Model utilizes strengths of both models, as there could be many images with Description-based Model and Deep Features Model when retrieving images alone could have missed, but those images can be retrieved by both models working together.

Results generated on five images, using all models of image retrieval, are attached in Appendix A.

## CHAPTER 3: THE EXPERIMENT

In this chapter, a step-by-step procedure in the development of the Hybrid Model, technology details, and related work to the adopted methods are discussed.

Technology details:

- Execution Environment: Linux
- Programming Language: Python
- The web-based, interactive, and computational environment, Jupyter Notebook, was used for writing programs, and the code written in it is included in Appendix B for further reference.

The experiment was conducted using following steps:

Step 1: Machine learning tool, namely Graphlab [8] with urllib (module which provides a high-level interface for fetching data across the World Wide Web) and urllib2 (module which defines functions and classes, which help in opening Universal Resource Locators in a complex world) were imported to the local machine. Graphlab is a Python package that enables users to perform end-to-end, large-scale data analysis. It is a graph-based, high performance, distributed computational framework, written in C++, so it quickly performs large-scale and high-performance data analysis. It has libraries for data transformation, manipulation, and model visualization. Graphlab also is comprised of scalable machine learning toolkits and contains several elements, which can improve existing machine learning models. The package exposes the underlying algorithms and parameters used, the models can be modified as per requirements.

Figure 2. depicts the architecture of Graphlab.

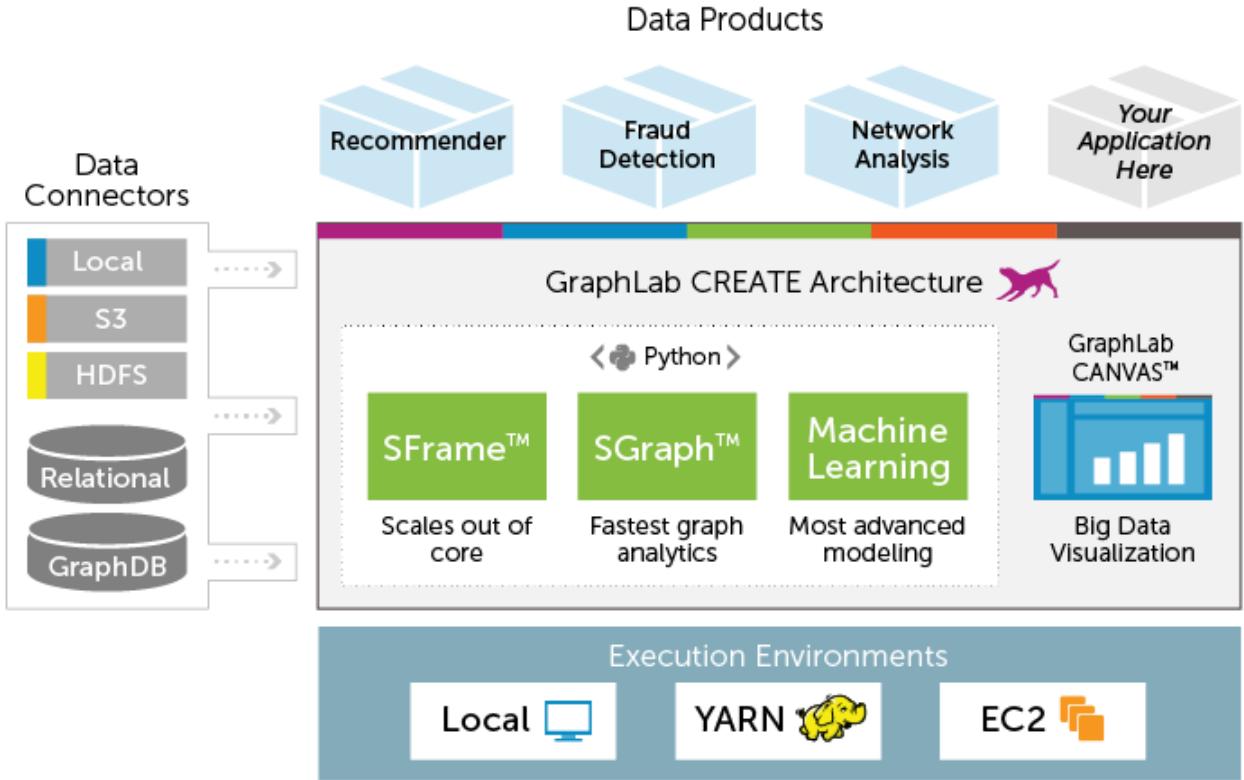


Figure 2. Architecture of Graphlab (source: Turi, Inc.)

The package was used via Jupyter Notebook, which gave access to two scalable data structures called SFrame and SGraph for analysis of tabular and graph data sets respectively. In the experiment, SFrames for storing images and captions with their necessary details were used. SFrame is similar to Pandas or R data frames, and most of the syntaxes are exactly the same or can be easily translated from one to the other. It is an efficient, disk-based tabular data structure, which is not limited by RAM. It is scalable, so SFrame handles large data sets for analysis and data processing.

Previous works on image retrieval when done on smaller datasets resulted in spurious results. Conversely, increasing the number of images in the dataset resulted in significant improvements in the quality of results. The experimental analysis required a large number of images with descriptions, so for this criteria, the SBU Captioned Photo Dataset

(<http://tlberg.cs.unc.edu/vicente/sbucaptions>) was the best suited. This is a large data set containing one million images from Flickr ([www.flickr.com](http://www.flickr.com)) with associated captions written by users. SBU Captioned Photo Dataset has files namely:

- SBU\_captioned\_photo\_dataset\_urls.txt: This file contains one million URLs corresponding to each image within the dataset. On downloading all the images from these URLs, it will result in a number less than one million, because images are owned by users of Flickr and can be removed at any time.
- SBU\_captioned\_photo\_dataset\_captions.txt: This file contains one million captions corresponding to each image within the dataset. The captions are in the same order as the URLs in the above file. For instance, the caption in line  $n$  in this file corresponds to the picture pointed by the URL in line  $n$  of the SBU\_captioned\_photo\_dataset\_urls.txt.

The descriptions are filtered so that they are likely to refer to visual content [9]. As one content of SBU Captioned Photo Dataset is text file with URLs, importing urllib and urllib2 to open resources in URLs from SBU\_captioned\_photo\_dataset\_urls.txt was necessary.

All the downloaded images are stored in SFrame. Any column of SFrame is an SArray, which is a series of elements stored on a disk, making the tabular data structure disk based. SGGraph is a scalable graph data structure and stores vertices and edges of graph in SFrames. Graphlab supports various data sources, and Local Machine and AWS S3 were used in combination for the experiments.

Step 2: The general practice to obtain a dataset required the use of a format that resembles a table, which is a straightforward format to use when cleaning data in preparation for more complicated data analysis. In this step, SFrame (tabular data structure to hold various types of data)

was used. The data types used in this SFrame, which was named as sfimage, were integers and images. All the first 100,000 images from the SBU Captioned Photo Dataset were downloaded and stored within. It is common knowledge that images are owned by users of Flickr, and they might be removed at any time by them. Knowing this, if an image is downloaded and used in sfimage after a user removed image of Flickr, the coherency with captions will be lost. Coherency loss occurs due to the fact that SBU\_captioned\_photo\_dataset\_captions.txt contains one million captions corresponding to each image in this dataset, and the captions are in the same order as the URLs. Captions are already downloaded to SBU\_captioned\_photo\_dataset\_captions.txt, and image removal by a user will not affect the sequence of captions. The caption in line  $n$  in this file corresponds to the image pointed by the URL in line  $m$  ( $m > n$ ). To solve this issue, every image is associated with its line number. For this reason, sfimage contains one column with integers and another column with images. Integers hold line number of the images' URLs in SBU\_captioned\_photo\_dataset\_urls.txt file. In this step, sfimage was initialized with '1' and an image to start the construction of SFrame for holding 100,000 images.

Step 3: URLs were taken from SBU\_captioned\_photo\_dataset\_urls.txt, and a “for loop” to read all the lines from the text file was created until reaching a point when there was no line left to read or any condition of break. Due to time constraints, it was not possible to analyze one million images, so a breaking condition was set, which broke loop when the 100,000 line was reached.

sfimage has one column to store only data of type image, and there can be situations in which a URL will not fetch any image due to user removal. This will cause an unexpected JPEG decode failure: a type of toolkit error. With regards to the experiment, when no image is found, it is expected that the remaining images would be the only ones to load in sfimage and URLs not fetching any image would be ignored. Since a static page was requested and did not send data to

the server, urllib2 was used to make a connection. urllib2.Request(line) returns an object suitable for use with urlopen(), in which a parameter line is a correctly formed URL for either a simple web page or with encoded data. urllib2.urlopen(urllib2.Request(line)) connects to a web server by using HTTP to retrieve data and return file objects.

If a file object contains an image, the image with the URL's line number was saved within sfimage. In other cases, no operation is performed. To copy a network object denoted by the URL to a local file, urllib.urlretrieve(line, 'Imagen.jpg') was used, which stores images as Imagen.jpg in the present working directory. To make image pre-processing easier for input into subsequent learning methods, Imagen.jpg was made compatible with Graphlab. In order to add the rows of an SFrame to the end of sfimage, both SFrames must have the same set of columns with the same column names and column types. For this, another SFrame, which was named frame2, with a Graphlab compatible image object and its occurrence number in SBU\_captioned\_photo\_dataset\_urls.txt. frame2, was then added at the end of sfimage. The whole process repeated 100,000 times.

Step 4: To hold all the captions, another SFrame was created and named as sfcaption. In this step sfcaption was initialized with an integer '1' and a caption.

Step 5: A process very similar to that of Step 3 was used to construct another SFrame, which was named sfcaption, consisting captions and corresponding line number in SBU\_captioned\_photo\_dataset\_captions.txt file. The major difference is that in Step 3, the resulting SFrame contained images, but in this case, the resulting SFrame contained captions.

Step 6: Billions of users share images every day for exchanging information, so finding any particular type of image requires exploration in the ever-growing collection of images. Image

retrieval falls under two types of categories: exact image retrieval and relevant image retrieval. Our work focuses on relevant images retrieval. For images retrieval, we first explored images using computer vision technologies, which is a wide field with sub-domains that deal with recognition of content within the image. Scale Invariant Feature Transform (SIFT) [10, 11] was pioneered as a new type of interest point detector, which ignores size change, view point change, and depth change during recognition. It uses primate visual system logic to identify an interest point, which makes it robust to scale change, rotation, distortion and occlusion. SIFT can be used for classification, as after running SIFT textures over the image at various locations, image key point detections were observed. A vector of detections, which is invariant to image translation, scaling, and rotation, describe the image based on key point location, can be used for Object Detection using a trainable classifier. Histogram of Oriented Gradients (or HOG) [12] is another feature detector which counts occurrences of gradient orientation in localized portions of an image and has been used a lot in Human Detection. It has also been extended for use in Object Detection [13]. The standard image classification approach uses feature detectors (SIFT, HOG, etc.) for finding interest points and feeding this to a classifier, but these ways of finding image similarity based on image quality are hand-built and have been found to be very difficult in creation.

ImageNet (<http://image-net.org/>) was started in year 2007 at Stanford Vision Lab and has over fifteen million labeled, high resolution images of approximately 22,000 categories. The images were collected from the web and was labelled by Amazon Mechanical Turk. There is an annual ImageNet Large Scale Visual Recognition Challenge (ILSVRC) (<http://www.image-net.org/challenges/LSVRC/>) and in the year 2012, team SuperVision's AlexNet [5] achieved top 1 and top 5 test set error rates of 37.5% and 17.0% respectively. The previous best score for top 1 and top 5 test set error rates were 45.7% and 25.7% respectively [14]. SuperVision's AlexNet were

inspired by the digit recognition task on MNIST[6], which has almost achieved human accuracy using Deep Convolutional Neural Networks. .

There is high computational error and extreme trouble of tuning in Deep Learning and to make this task easier, Transfer Learning, which utilizes knowledge gained from one solution to reuse it for a different problem, was used. It has been around since many years [15]. It has also been utilized for Unsupervised Learning [16] but had limited use in Convolutional Neural Networks.

In this step, we downloaded a pre-trained model of SuperVision's AlexNet, created by the team Turi, and stored in AWS S3 to our local machine. The pre-trained model was named as `deep_learning_model`.

Step 7: Using the `deep_learning_model`, the features were extracted and passed into a machine learning algorithm (a simple Classifier). A collection of some Deep Features for single images is shown in the text box below.

```
0.0, 3.4390671253204346, 0.0, 1.1566247940063477, 0.0, 0.0,  
0.0, 0.0, 0.0, 4.177254676818848, 3.1828880310058594e-05,  
0.0, 0.0, (Continued for 1000 times)
```

A lot of features are zeros, as ImageNet was created on 1.2 million images of 1000 object categories, so for each image, 1000 features were created, and there would be many features in those images that do not make sense for every image data, thus resulting in zero outcome.

Each column in an SFrame is a size-immutable, but SFrames are mutable in that columns can be added and subtracted with ease. So another column in sfimage was created and labeled as ‘deep features’ and extracted features were stored using deep\_learning\_model.

Step 8: The task was to process text and convert the content of string SArrays (holding captions) of sfcaption to a dictionary of (word, count) pairs. The strings were first tokenized into words, and then, word counts were accumulated. In each output dictionary, the keys were the words in the corresponding input data entry, and the values were the number of times the words appeared. By default, words were split on all whitespace and newline characters. The output is commonly known as the “bag-of-words” representation of text data; dictionary keys and list elements are strings.

Step 9: Term Frequency (TF) [17] is the frequency of a word in a document and is the most obvious technique to find relevancy. The more frequent a word is; the more relevance the word holds in the context. Inverse Document Frequency (IDF) [18] is the inverse of the document frequency among the whole corpus of documents. IDF is the parameter which helps us determine the relevance of words and is based on the principle that less frequent words are generally more informative. The relevance of words (obtained from IDF) and the occurrence of words in the documents (obtained from TF), can be multiplied to obtain TF-IDF [19] which has been mainly used because if a user searches for “the increase in number of images” on Google, it is certain that “the” will occur more frequently than “images,” but the relative importance of images is higher from the search query point of view. In such cases, TF-IDF weighting negates the effect of high frequency words in determining the importance of an item (description). TF-IDF was the underlying method used to retrieve similar descriptions. The task in this step was to compute the TF-IDF scores for each word in each caption. It required the collection of captions to be in a bag-

of-words format, which was previously established in Step 8. Another column in sfcaption was created and labeled as ‘tf-idf’ for storing the TF-IDF computed prior.

Step 10: The Graphlab’s Nearest Neighbors’ toolkit is used to find the rows in a data table that are most similar to a query row. This is a two-stage process, where a Nearest Neighbors Model was created, using a reference dataset contained in an SFrame. A query was then performed on the model. Specific feature (“deep features”) were listed to use in our distance computations, which is the most common choice in computing Nearest Neighbors; distance Function measures the dissimilarity between any pair of observations. The model was named as image\_nn\_model.

Another Nearest Neighbor Model, which was named as caption\_nn\_model, was trained on specific features TF-IDF of captions

Step 11: This step fetched first 1000 Nearest Neighbors from the image retrieval system that located the similar images for any image from Deep Features.

Step 12: This step fetched first 1000 Nearest Neighbors from the Description-based retrieval system that located the similar descriptions for any description using TF-IDF.

Step 13: Using inner join function Hybrid Model is created from image\_nn\_model and caption\_nn\_model.

Step 14: In this step, a function was created to fetch images using each model.

Step 15-17: These steps displayed results using various methods.

## CHAPTER 4: AUTOMATIC TEXTUAL DESCRIPTION

During analysis on various images, one situation was encountered when the image below



had a description “Pallino and Jasmine on my window while staring at some pigeons down in the street”. As SBU Captioned Photo Dataset has human generated caption, there is the possibility of the caption wrongly describing an image. In this situation, a method to fix improper descriptions for natural images [4] such as this, will be discussed in the this chapter.

For humans, preparing concise descriptions has always been an effortless task, but it is a challenging problem when completed by automatic methods. Recently, there has been proliferation in attempts to generate concise descriptions for images, due to an upsurge in images on the Internet, which requires content interpretation. There have been a vast number of solutions to automating the descriptions of images, and many of them have come from Natural Language Processing and Computer Vision experts working in collaboration. Both Natural Language Processing and Computer Vision have originated from Artificial Intelligence, but they have developed separately. Algorithms generating precise and detailed description of images are present, but this field has a lot of challenges and is still developing. The advances in this technology can identify objects such as  $n$  number of people, or a ticket counter, but will not be able to interpret if people are waiting in queue for their turn to come in to purchase tickets.

The major parts of this work are:

- Analyzing the visual content: Computer Vision has helped tremendously in providing solution to this but sometimes solution remains out of its scope as Computer Vision detects Objects, Actions and Scenes, also determine Attributes[20] and gives unstructured lists of detection, which makes the task of creating descriptions extremely difficult.
- Generating a textual description: Natural Language Processing has taken care of this part. It has modified unstructured list of detections to make it syntactically, lexically and grammatically correct. Natural Language Processing makes descriptions linguistically richer.

Generation of description depend on recognizing the image content and composing descriptions based on detections by Yang, Teo [21], [22, 23]. Early work on generating description

done in [23] depends on combining computer vision predictions of image and smoothing noisy detections with text from large document corpora. It uses computer vision to detect and classify image's objects/stuffs and associating detections with attributes and prepositions. It then combines these vision based potentials with text potentials using a Conditional Random Field (CRF) to predict final labelling for image. A diagrammatic representation of image description system proposed by [23] is shown in Figure 3.

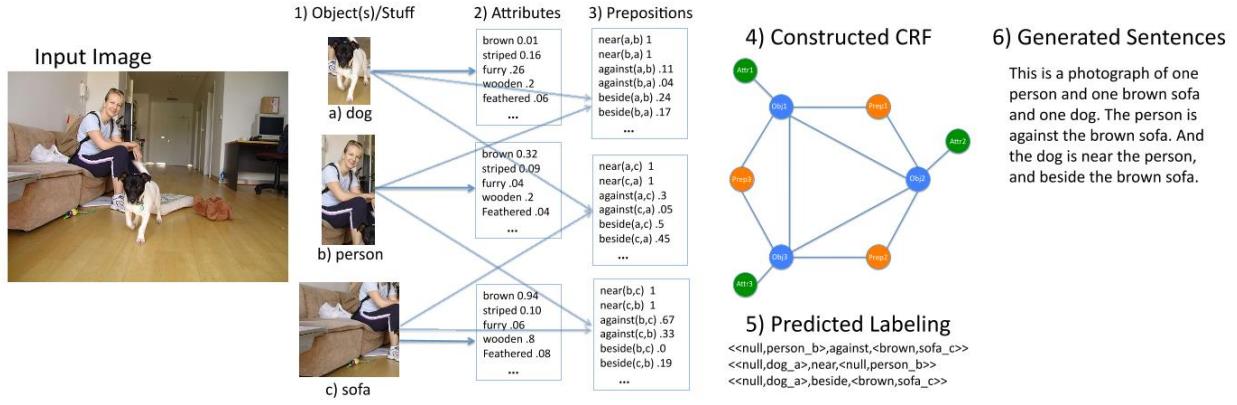


Figure 3. Diagrammatic representation of image description system

The task of automatic description is heavily motivated from work by [4] where interactions of salient objects has been used for descriptions of images. The method first decides what the story of image is, or what needs to be expressed about a picture. It proposed an approach for automatically generating textual descriptions of images, by detecting essence of story for an image using Saliency Map to find objects to talk about. It then identifies various objects. If salient objects are not determined, then a long list of worthless detected objects will create difficulty for image description, so good image understanding is essential but not sufficient. To detect candidate region of interest, a method discussed in [24] have used sliding window mechanism. The method by [4] uses saliency maps where the application needs a saliency detector to locate the parts of the image

that preserve the essence of the original picture. It used context aware saliency detector [25] which is contrast to the classic algorithm of for detecting the salient parts of image whose goals is to either identify fixation points or detect the dominant object. It extracts the essence from the image to find the salient regions that are distinctive both locally and globally. The authors of context aware saliency method followed below four principles of human visual attention which are supported by psychological evidence[26-29].

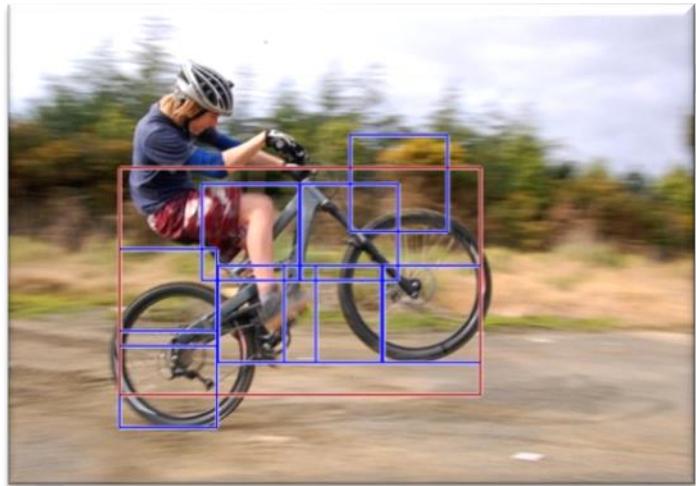
1. Local low-level considerations, including factors such as contrast and color.
2. Global considerations, which suppress frequently occurring features, while maintaining features that deviate from the norm.
3. Visual organization rules, which state that visual forms may possess one or several centers of gravity about which the form is organized.
4. High level factors, such as human faces.

After finding Salient Regions in Image it applies Object Detectors. In the study Deformable Part Model [13, 30] is used to detect the objects from 20 PASCAL 2008 categories.

20 PASCAL 2008 classes:

- *Person*: person
- *Animal*: bird, cat, cow, dog, horse, sheep
- *Vehicle*: aeroplane, bicycle, boat, bus, car, motorbike, train
- *Indoor*: bottle, chair, dining table, potted plant, sofa, tv/monitor

The Deformable Part Model decomposes objects in its constituent parts that are probabilistically linked to each other. Figure 4 is demonstration of it.



Man On a Bike

If an object from any of the 20 classes is detected: Then a big bounding box in Red is generated which shows the location of the object and smaller boxes in Blue are generated to show the locations of the constituent parts of that object.

*Figure 4. Deformable Part Model of man riding bike*

Often objects in image are not singular. We can have more than one objects. More object leads to interaction between objects which requires to be included in generated descriptions. Model in [31] represented these interactions using the most likely <object, action, scene> triplet for every image. But in [4] the interaction between two objects is determined directly using the information in object poses returned by object detectors.

Work done in [4] of generating descriptions is heavily motivated by the work [32], where using captions of images in SBU Captioned Photo Dataset the authors have built an N-gram model of descriptions and have considered following three different types of relationships extracted from treebank parsing guidelines in between two objects:

- Prepositional
- Verbal
- Verbal with Preposition

The process proposed by [32] is as follows:

First objects are split into different noun groups of size three and nouns are ordered in each group based on common ordering of nouns in the sample descriptions. Then sub-trees are built around every object, where the left object grows only to the right, the right one grows to the left and middle one, if any, grows in both directions. Then, they combined these sub-trees to create full syntactic trees.

The flaw in this method is that the actions in sub-trees and interactions between sub-trees are mostly, as they pointed out, ‘hallucinated,’ implying that they are from the N-grams rather than actual image content. As a result, the guessed verb often appeared to be incorrect. In the method of [4] interactions are predicted from the pose and positions of objects in the image. A table of actions and interactions between any two objects in the set of 20 PASCAL Objects is generated from the SBU Captioned Photo Dataset. It uses the same three different object-object relationships that are used in [32] and Table 2 shows some of the actions of a Person’s action and the possible interactions of a person with bike. (i), (ii) and (iii) represents Prepositional, Verbal and Verbal with Prepositional respectively.

*Table 2. Actions and possible interactions of a person with bike*

Objects	Actions	Interactions (with the bike)
Person	Sitting Running Standing	Riding(ii) Next to (i) Behind(i) Holding(ii), Parking (ii), To the right of(i)

	Posing	Riding with a Helmet (iii)
--	--------	----------------------------

Actions have been used to expand sub-trees. So the example descriptions that describe interactions between a person and a bike can be person riding with a helmet on a bike. In order to find all these actions and interactions, the captions in SBU Captioned Photo Dataset (one million images/captions) was searched with every object pairs in 20 Pascal Categories (Say Man and Dog) by using AWK command. We parsed all Object Pair captions generated previously, with the Stanford Parser[33] to determine the verb for interactions.

A list of all verbs is attached as Appendix C for Man-Dog combination interaction.

After identifying Objects and Interactions, we identify background as sometime background of image plays crucial role[31]. So for this subset of SUN Database [34] having 15 categories namely <Room, Suburb, Industrial, Kitchen, Living Room, Coast, Forest, Highway, Inside City, Mountain, Open Country, Street, Tall Building, Office and Store>, was used. Scene is then added with proper preposition [35] at the end.

Figure 5 depicts methodology of [4]

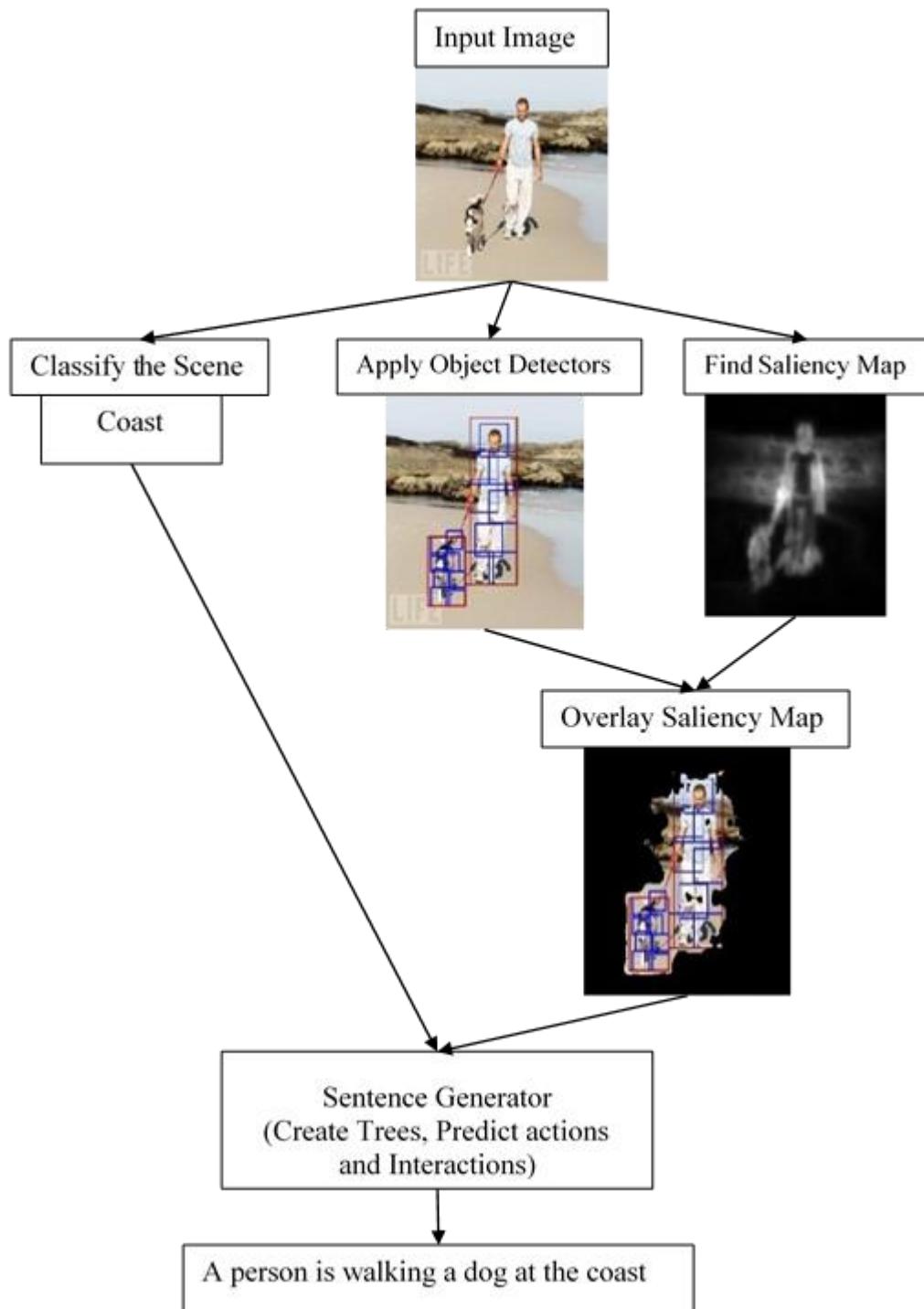


Figure 5. Steps of method for image description

Table 3 shows Result generated by[4] and to evaluate it, the consensus-based approach [36] is proposed. As the consensus-based approach requires 50 descriptions to avoid discrepancy, so using PASCAL-50 sentences the 50 descriptions for each image have been attached in Appendix D.

*Table 3. The results of Automatic Textual Description [4].*

Image	Salient Objects	Descriptions
	Person Bike Bike	A person standing next to a bike next to a bike in coast. A person Parking a bike next to a bike in coast.
	Person Dog Chair	A person walking a dog next to a chair in living room. A person on the left of a dog in front of a chair in living room. A person playing with a dog in front of a chair in living room



Person

A person riding a bike in front of a car in street.

Bike

A person riding on a bike in a car in street.

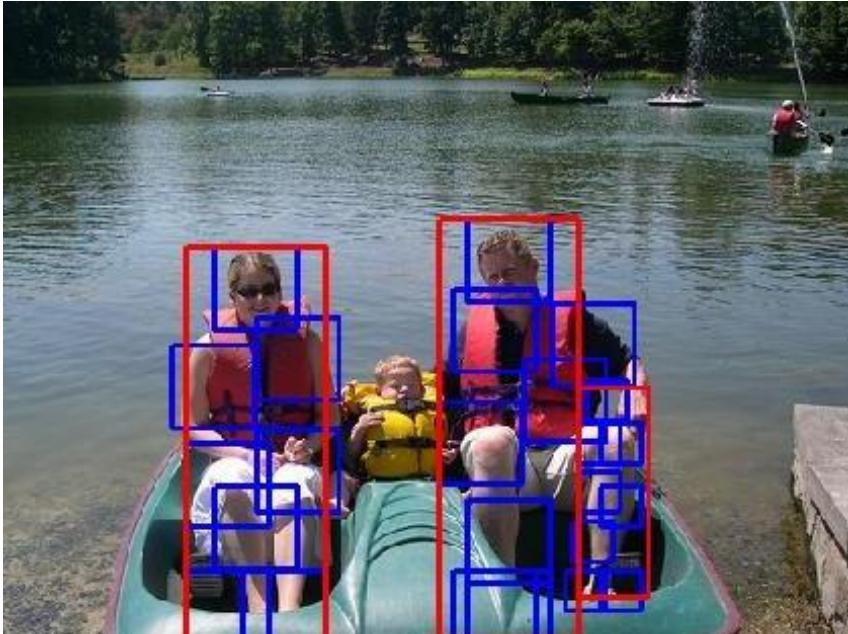
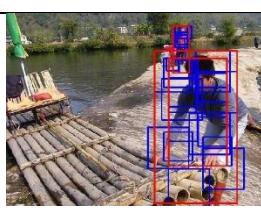
Car

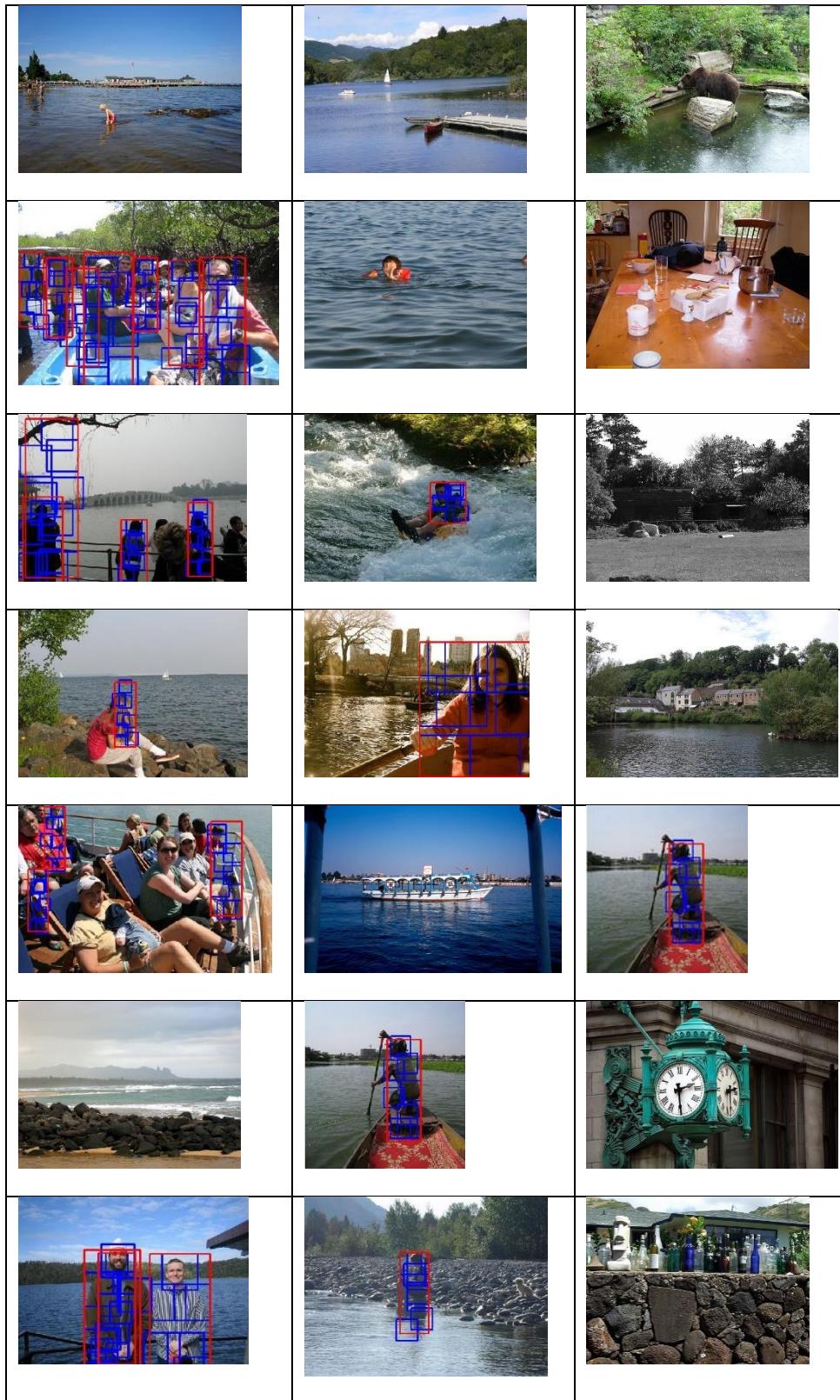
A person on a bike in a car in street.

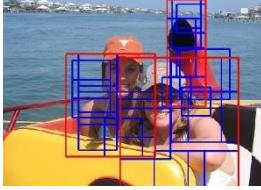
## CHAPTER 5: RESULTS

The method for evaluating results was to see how same or similar the object detected in the original image to images generated by various models occurred. It is done by detecting an object using discriminately trained Deformable Part Model [37]. As original image had persons, Deformable Part Model was trained to detect persons. It detected content of every image generated using each type of image retrieval model, and the if images had persons then big-bounding boxes was created. Table 4 shows collection images after passing through Deformable Part Model.

*Table 4. Object Detection using Deformable Part Model*

Original Image # 2			
Hybrid Model	Deep Features Model	Description-based Model	
			



		
Correct Count = 7/9	Correct Count = 5/9	Correct Count=1/9

As images have more objects and attributes than just one or two object types from VOC Pascal 20 object categories, so API from [www.clarifai.com](http://www.clarifai.com) was used, which offers image recognition as a service. Clarifai API uses artificial intelligence to say what's inside the image content and has a tag endpoint which is used to tag the contents of the image. Clarifai API generates tags, associated with a probability, for every image. Images are input into Clarifai system, which processes using Deep Learning platform, and a list of tags is returned. Using Clarifai evaluation was done by generating all tags with a probability greater than 95% for the original image, as well as images generated from all image retrieval models. For all images retrieved from each model, tags which are the same as the original image, were collected to create a table. The original Image # 5 Analysis is in Table 5.

Table 5. Analysis of image #5

	Hybrid Model	Deep Features Model	Description-based Model
Tree	9	9	4
Nature	9	8	6
Outdoors	3	6	3
No Person	7	6	6
Bird	4 + 4 Woodpecker 1 Parrot 1 Animal 1 Avian 1	3 + 1 Animal 1	2 + 2 Woodpecker 1 Animal 1

Wood	3	5	2
Wildlife	3	1	1
Park	1	0	0
Environment	2	1	0
Total	45	40	26

As noticed, the Hybrid Model performed better than the primitive way of image retrieval. The result for method of image description discussed in Chapter 4 will rely heavily on its score on CIDEr [36], which is a novel paradigm for evaluating image descriptions using human consensus.

## CHAPTER 6: CONCLUSION AND FUTURE WORKS

We presented a novel approach of recommending images based on similarity by retrieval using Deep Features extracted from specific Convolutional Neural Network Model and TF-IDF of Description-based Model. Our method showed an increase in efficiency over Deep Features Model and Description-based Model on evaluation when we experimented on 100,000 images of SBU Captioned Photo Dataset. Two methods for evaluation of result were discussed, and evaluation techniques can be modified to a more automatic approach to find correct, accurate efficiency and conduct it easily on a larger number of images.

A new method for textually describing images was explored, which finds interesting objects and entities, and determines interactions between them. The method was lacking in result evaluation, and a consensus based technique was explored to compute its performance.

Our dataset consisted of natural images, but to make it more practicable for Recommendation System, it can be done on datasets consisting of image products.

Another modification possible is, during intersection of sets, we gave weight to both Deep Features Model and Description-based Model as the same, but by using Neural Network's back propagation method, we can accurately determine correct contribution of each method and can assign weights accordingly.

## REFERENCES

- [1] Sarwar, B.M., et al., *Item-based collaborative filtering recommendation algorithms*. 2001. p. 285-295.
- [2] Pazzani, M.J. and D. Billsus, *Content-Based Recommendation Systems*. 2007. p. 325-341.
- [3] Schein, A.I., et al., *Methods and metrics for cold-start recommendations*. 2002. p. 253-260.
- [4] Adeli, H., *Modeling Salient Object-Object Interactions to Generate Textual Descriptions of Natural Images*, in *Computer Science*. 2012, East Carolina University: Greenville.
- [5] Krizhevsky, A., I. Sutskever, and G.E. Hinton, *ImageNet Classification with Deep Convolutional Neural Networks*. 2012. p. 1106-1114.
- [6] Cun, Y.L., et al., *Handwritten digit recognition with a back-propagation network*, in *Advances in neural information processing systems 2*, S.T. David, Editor. 1990, Morgan Kaufmann Publishers Inc. p. 396-404.
- [7] Donahue, J., et al., *DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition*. 2014. p. 647-655.
- [8] Low, Y., et al., *GraphLab: A New Framework for Parallel Machine Learning*. CoRR, 2010. **abs/1006.4990**.
- [9] Vicente, O., K. Girish, and L.B. Tamara, *Im2Text: Describing Images Using 1 Million Captioned Photographs*. 2011: p. 1143--1151.
- [10] Lowe, D.G. *Object recognition from local scale-invariant features*. in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*. 1999.
- [11] Lowe, D.G., *Distinctive Image Features from Scale-Invariant Keypoints*. International Journal of Computer Vision, 2004. **60**(2): p. 91-110.
- [12] Dalal, N. and B. Triggs. *Histograms of oriented gradients for human detection*. in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. 2005.
- [13] Felzenszwalb, P.F., et al., *Object Detection with Discriminatively Trained Part-Based Models*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2010. **32**(9): p. 1627-1645.
- [14] Sanchez, J. and F. Perronnin. *High-dimensional signature compression for large-scale image classification*. in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. 2011.
- [15] Caruana, R., *Multitask Learning*. Machine Learning, 1997. **28**(1): p. 41-75.

- [16] Raina, R., A.Y. Ng, and D. Koller, *Constructing informative priors using transfer learning*, in *Proceedings of the 23rd international conference on Machine learning*. 2006, ACM: Pittsburgh, Pennsylvania, USA. p. 713-720.
- [17] Luhn, H.P., *A Statistical Approach to Mechanized Encoding and Searching of Literary Information*. IBM Journal of Research and Development, 1957. **1**(4): p. 309-317.
- [18] Jones, K.S., *A statistical interpretation of term specificity and its application in retrieval*. Journal of Documentation, 2004. **60**(5): p. 493-502.
- [19] Wu, H.C., et al., *Interpreting TF-IDF term weights as making relevance decisions*. ACM Trans. Inf. Syst., 2008. **26**(3).
- [20] Lampert, C.H., H. Nickisch, and S. Harmeling. *Learning to detect unseen object classes by between-class attribute transfer*. in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. 2009.
- [21] Yang, Y., et al., *Corpus-guided sentence generation of natural images*, in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 2011, Association for Computational Linguistics: Edinburgh, United Kingdom. p. 444-454.
- [22] Li, S., et al., *Composing simple image descriptions using web-scale n-grams*, in *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*. 2011, Association for Computational Linguistics: Portland, Oregon. p. 220-228.
- [23] Kulkarni, G., et al., *BabyTalk: Understanding and Generating Simple Image Descriptions*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2011. **35**(12): p. 2891-2903.
- [24] Piji, L. and M. Jun. *What is happening in a still picture?* in *The First Asian Conference on Pattern Recognition*. 2011.
- [25] Goferman, S., L. Zelnik-Manor, and A. Tal, *Context-Aware Saliency Detection*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2012. **34**(10): p. 1915-1926.
- [26] Koch, C. and T. Poggio, *Predicting the visual world: silence is golden*. Nat Neurosci, 1999. **2**(1): p. 9-10.
- [27] Paul, R.K., K. Koffka. *Principles of Gestalt Psychology*. 1955.
- [28] Treisman, A.M. and G. Gelade, *A feature-integration theory of attention*. Cognitive Psychology, 1980. **12**(1): p. 97-136.
- [29] Wolfe, J.M., *Guided Search 2.0 A revised model of visual search*. Psychonomic Bulletin & Review, 1994. **1**(2): p. 202-238.
- [30] Felzenszwalb, P., D. McAllester, and D. Ramanan. *A discriminatively trained, multiscale, deformable part model*. in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. 2008.

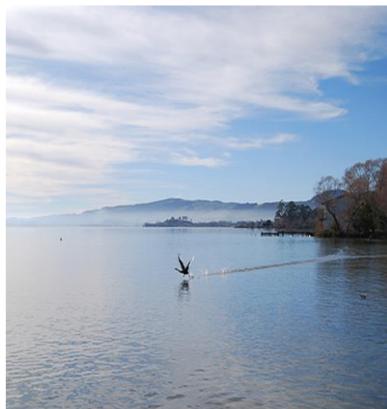
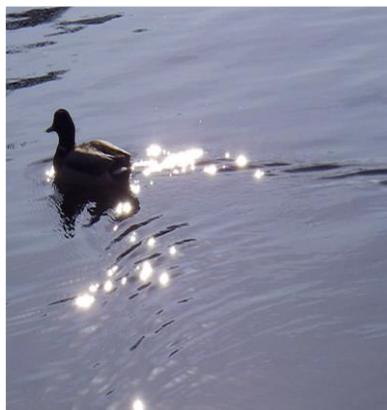
- [31] Farhadi, A., et al., *Every picture tells a story: generating sentences from images*, in *Proceedings of the 11th European conference on Computer vision: Part IV*. 2010, Springer-Verlag: Heraklion, Crete, Greece. p. 15-29.
- [32] Mitchell, M., et al., *Midge: generating image descriptions from computer vision detections*, in *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*. 2012, Association for Computational Linguistics: Avignon, France. p. 747-756.
- [33] Manning, M.M.a.B.M.a.C. *Generating Typed Dependency Parses from Phrase Structure Parses*. in *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC-2006)*. 2006. Genoa, Italy: European Language Resources Association (ELRA).
- [34] Xiao, J., et al., *SUN Database: Exploring a Large Collection of Scene Categories*. *International Journal of Computer Vision*, 2016. **119**(1): p. 3-22.
- [35] Gupta, A. and L.S. Davis, *Beyond Nouns: Exploiting Prepositions and Comparative Adjectives for Learning Visual Classifiers*, in *Computer Vision – ECCV 2008: 10th European Conference on Computer Vision, Marseille, France, October 12-18, 2008, Proceedings, Part I*, D. Forsyth, P. Torr, and A. Zisserman, Editors. 2008, Springer Berlin Heidelberg: Berlin, Heidelberg. p. 16-29.
- [36] Vedantam, R., C.L. Zitnick, and D. Parikh, *CIDEr: Consensus-based image description evaluation*. 2015. p. 4566-4575.
- [37] Felzenszwalb, P.F., et al., *Object Detection with Discriminatively Trained Part-Based Models*. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2010. **32**(9): p. 1627-1645.

## Appendix A: Output Images from each models

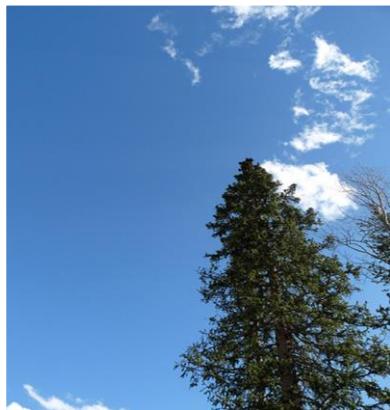


Original image # 1

## Hybrid Model



## Deep Features Model



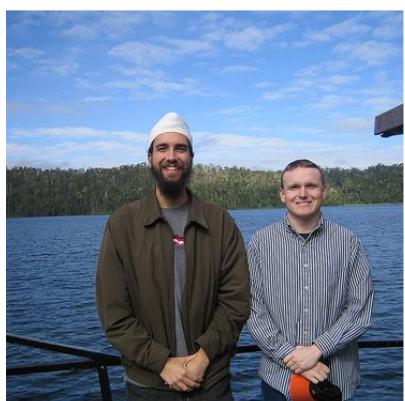
## Description-based Model



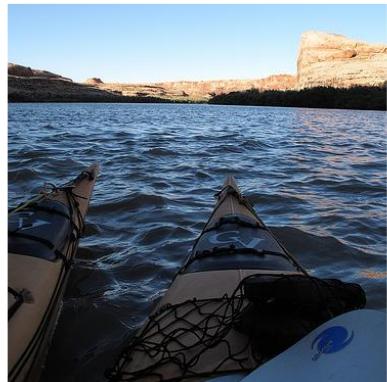


Original image # 2

## Hybrid Model



## Deep Features Model



## Description-based Model





Original image # 3

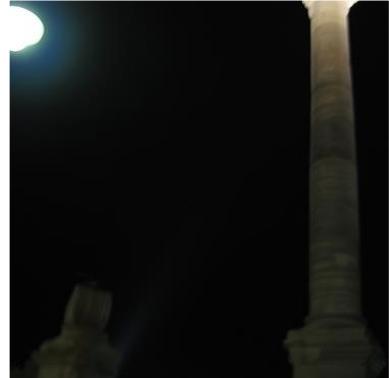
## Hybrid Model



A n s a d a v u t      P h o t o g r a p h y



## Deep Features Model



## Description-based Model





Original image # 4

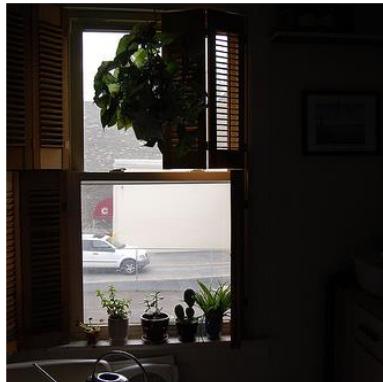
## Hybrid Model



## Deep Features Model



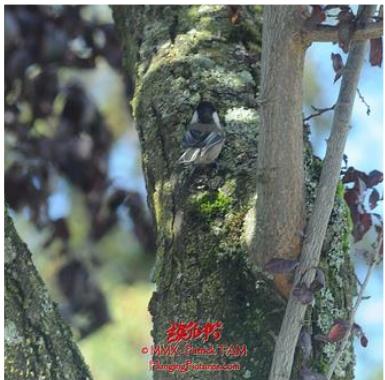
## Description-based Model



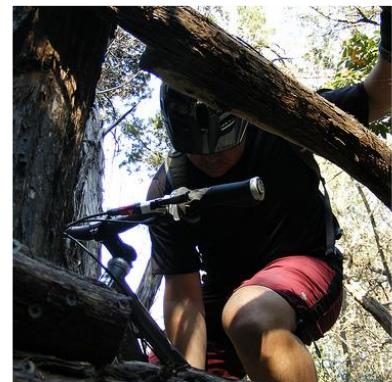


Original image # 5

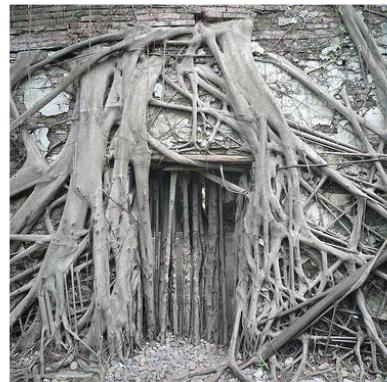
## Hybrid Model



## Deep Features Model



## Description-based Model



## Appendix B: Code for Experiment



Steps for building Intelligent Model for Image-Based Recommendation System

```
# Step 1: Fireup Graphlab Create and import urllib & urllib2.
```

```
import graphlab  
import urllib  
import urllib2
```

```
# Step 2: Initialize Count and a SFrame to hold 100K images.
```

```
count = 1  
img = graphlab.Image('image0.jpg')  
sfimage = graphlab.SFrame({'image' : [img], 'count' :[count]})
```

```
# Step 3: Loop to retrieve first 100K images of SBU 1M images dataset from text file consisting  
the URLs.
```

```
#Few images have been removed by user, so some part of the code handles those situations when  
image is unavailable.
```

```
for line in open("SBU_captioned_photo_dataset_urls.txt"):  
    count = count + 1  
    if count < 100000:  
        try:  
            f = urllib2.urlopen(urllib2.Request(line))  
            imageFound = True  
            urllib.urlretrieve(line, 'Imagen.jpg')  
            img = graphlab.Image('Imagen.jpg')  
            frame2= graphlab.SFrame({'image' : [img], 'count' :[count]})  
            sfimage = sfimage.append(frame2)  
        except:  
            imageFound = False
```

A newer version of GraphLab Create (v2.1) is available! Your current version is v1.10.1.

You can use pip to upgrade the graphlab-create package. For more information, see  
<https://dato.com/products/create/upgrade>.

This non-commercial license of GraphLab Create for academic use is assigned to  
prateekp14@students.ecu.edu and will expire on February 16, 2017.

```
[INFO] graphlab.cython.cy_server: GraphLab Create v1.10.1 started. Logging:  
/tmp/graphlab_server_1470451888.log
```

```
# Step 4: Initialize another SFrame to hold 100K captions.
```

```
count = 1
```

```
sfcaption = graphlab.SFrame({'caption' : ['A person playing with a dog in front of a chair in living room'],
                             'count' :[count]})
```

# Step 5: Loop to extract captions from text file.

```
for line in open("SBU_captioned_photo_datasetCaptions.txt"):
```

```
    count = count + 1
```

```
    if count < 100000:
```

```
        caption = line
```

```
        frame3= graphlab.SFrame({'caption' : [caption], 'count' :[count]})
```

```
        sfcaption = sfcaption.append(frame3)
```

# Step 6: Load ImageNet model of extracting Deep Features.

```
deep_learning_model = graphlab.load_model('http://s3.amazonaws.com/GraphLab-Datasets/deeplearning/imagenet_model_iter45')
```

# Step 7: Computing Deep Features for our set of 100K images.

```
sfimage['deep_features'] = deep_learning_model.extract_features(sfimage)
```

Downloading http://s3.amazonaws.com/GraphLab-

```
Datasets/deeplearning/imagenet_model_iter45/dir_archive.ini to /var/tmp/graphlab-anant/2148/38392426-73db-4c65-9509-b0e62a1f7299.ini
```

Downloading http://s3.amazonaws.com/GraphLab-

```
Datasets/deeplearning/imagenet_model_iter45/objects.bin to /var/tmp/graphlab-anant/2148/c692bf4c-0c71-47e6-a821-7e0286e69959.bin
```

Images being resized.

# Step 8: Get the word counts for all captions in sfcaption SFrame.

```
sfcaption['word_count'] = graphlab.text_analytics.count_words(sfcaption['caption'])
```

# Step 9: Compute Term Frequency and Inverse Document Frequency for all entries of the sfcaption Dataset.

```
tfidf = graphlab.text_analytics.tf_idf(sfcaption['word_count'])
```

```
sfcaption['tfidf'] = tfidf
```

# Step 10: Train the image retrieval system that finds nearest neighbor for any image from deep features.

```
image_nn_model = graphlab.nearest_neighbors.create(sfimage,features=['deep_features'],
                                                    label='count')
```

```
caption_nn_model = graphlab.nearest_neighbors.create(sfcaption,features=['tfidf'],
                                                    label='count')
```

Starting brute force nearest neighbors model training.

Starting brute force nearest neighbors model training.

```
# Step 11: Create a function to fetch m nearest neighbors for nth image (m=1000 & n=200 in this case).
```

```
queryimage = sfimage[sfimage['count']==200]
```

```
def get_images_from_ids(query_result):
    return sfimage.filter_by(query_result['reference_label'],'count')
```

```
qi_neighbors1 = get_images_from_ids(image_nn_model.query(queryimage, k = 1000))
Starting pairwise querying.
```

+-----+	+-----+	+-----+	+-----+
Query points	# Pairs	% Complete.	Elapsed Time
+-----+	+-----+	+-----+	+-----+
0   1   0.00109599   143.121ms			
0   3398   3.72416   1.10s			
0   6780   7.43079   2.10s			
0   10116   11.087   3.10s			
0   13507   14.8035   4.11s			
0   16835   18.4509   5.11s			
0   20227   22.1685   6.10s			
0   23555   25.816   7.11s			
0   26923   29.5072   8.10s			
0   30275   33.181   9.11s			
0   33603   36.8284   10.12s			
0   36931   40.4759   11.11s			
0   40323   44.1935   12.11s			
0   43715   47.911   13.11s			
0   47043   51.5585   14.11s			

0	50310	55.1391	15.10s	
0	53671	58.8227	16.10s	
0	57018	62.491	17.11s	
0	60355	66.1483	18.11s	
0	63619	69.7256	19.11s	
0	66923	73.3467	20.11s	
0	70211	76.9503	21.11s	
0	73539	80.5978	22.11s	
0	76940	84.3252	23.11s	
0	80295	88.0022	24.11s	
0	83661	91.6913	25.11s	
0	86991	95.341	26.11s	
0	90396	99.0728	27.11s	
Done		100	27.44s	

# Step 12: Create a function to fetch m nearest neighbors for nth description (m=1000 & n= 200 in this case)

```
querycaption = sfcaption[sfcaption['count']==200]
```

```
def get_captions_from_ids(query_result):
    return sfcaption.filter_by(query_result['reference_label'],'count')
```

```
qi_neighbors2 = get_captions_from_ids(caption_nn_model.query(querycaption, k = 1000))
```

# Step 13: Join fetched descriptions with sfimage SFrame to see the images for them.

```
qi_neighbors2 = qi_neighbors2.join(sfimage, on= 'count', how = 'inner')
```

Starting pairwise querying.

+-----+	-----+	-----+	-----+
---------	--------	--------	--------

Query points   # Pairs   % Complete.   Elapsed Time
---

+-----+	+-----+	+-----+
0	1	0.00100001   235.872ms
Done		100   338.38ms
+-----+	+-----+	+-----+

```
# Step 14: Join results from both methods to make a Hybrid Model
neighbors_common = qi_neighbors1.join(qi_neighbors2, on= 'count', how = 'inner')
# Step 15: Display Results of Image Model Method
qi_neighbors1['image'].show()
Canvas is accessible via web browser at the URL: http://localhost:37825/index.html
Opening Canvas in default web browser.

# Step 16: Display Results of Sentence Model
qi_neighbors2['image'].show()
Canvas is accessible via web browser at the URL: http://localhost:37825/index.html
Opening Canvas in default web browser.

# Step 17: Display Results of Hybrid Model
neighbors_common['image'].show()
```

## Appendix C: Collection of verbs for Man-Dog combination

<walks>, <washes-rose>, <running>, <passed-made>, <walking>, <carries>, <leads>, <crashed>, <fetch>, <left-rode>, <shirt-walking>, <hat>, <running>, <reflected>, <walking-foliage>, <sleeps-sleeps>, <walking-2010>, <hole>, <relaxing>, <is-walking-was-going-chasing>, <walks-starts>, <was-thinking>, <putting-aka>, <walking>, <aving>, <helping>, <sunset-walking>, <ended-did>, <walks-draped-wear>, <rides>, <walking-855-am>, <was-walking-streetwalking>, <carrying>, <is-reflected-walking>, <crossing-covered>, <is-proves>, <sitting>, <sitting>, <england>, <finds>, <catching-walking>, <looks-looks-sees>, <sleeping>, <makes-feel>, <see-driving>, <was-set-were-shooting>, <kung-having-swim>, <walking>, <blooming-walking-cross-blanketed-\_1182>, <belongs>, <running>, <o>, <stops>, <having>, <shapes>, <hid-attacked>, <take>, <walking-855-am>, <takes>, <playing>, <walking>, <was-walking>, <walking>, <looking>, <was>, <w>, <walking>, <harbor>, <was>, <plays>, <egging-sleeping>, <were-sitting-drawn>, <smiling-forest>, <harbor>, <plays-waits>, <walking-@>, <were-riding>, <is-walking-is>, <resting-waiting-vaccinated>, <are-silhouetted-cross>, <owns-keeps>, <was-throwing>, <carrying>, <caught-waving>, <share>, <falling-walks>, <walking>, <walking>, <waterford>, <sitting>, <suppose-being-do-'t>, <sitting>, <loved>, <is>, <turned-dogtaring-got-upnd-followed>, <plays-15-2007>, <sitting-looking>, <swimming>, <shows-walking>, <playing>, <takes>, <had>, <looking-catches-walking>, <walking>, <walking>, <drinks-loves-dogs>, <walking-paquier>, <takes-drawing-sitting-2007>, <walking>, <walking>, <walking-do>, <walks>, <take-has-been-cleared>, <walking-is>, <is-made>, <harbor>, <harbor>, <walking>, <looks>, <make>, <walking-pictured>, <oming-saw-walking>, <do-'t-sit>, <playing>, <is-&>, <is-is-/k>, <walking-told-lived-had-died>, <were-living-welcomed>, <was-sleeping-played>, <walking-riding>, <cross>, <is-caught>, <walking>, <found>, <made>, <walking>, <seated>, <walking>, <walking>, <is-is>, <walking-is>, <looks-look-looks-looks>, <mastered-pitty>, <walking>, <taking>, <is-lived>, <going>, <walking>, <eating>, <running-dragging>, <walking>, <beach>, <made-bought-gave>, <walking>, <having>, <walking>, <walking-blizzard>, <walking>, <coated-led-aged>

## Appendix D: PASCAL 50 Sentences



[1] A guy holding two bicycles so they won't fall down. [2] A man holds the handles of two bikes. [3] A man holds two cruiser bikes on the beach. [4] A man is holding two bicycles upright. [5] A man looking at two bicycles. [6] A man is holding two bikes at once. [7] a man holds two bicycles by the handlebars [8] A man is holding two bicycles [9] a man holding up two bikes taking a look at the one to the left of him [10] A man is holding two bikes. [11] a man that has two bikes [12] A man is holding two bicycles. [13] A young man holds up two bicycles. [14] A man is evaluating two bicycles. [15] A man holding two bicycles. [16] A man is holding on to two bicycles. [17] A man is standing and holding two bikes. [18] A man is holding up two bicycles. [19] man holding a couple bikes up [20] A man standing by a beach and holding two bikes [21] A man examines 2 bicycles. [22] A man testing out the grips on two different bikes. [23] A man is holding two bicycles on pavement next to sand. [24] A man holds two beach cruisers by their handlebars. [25] A man holds two bikes parked on the beach [26] A man holds two bicycles on the sidewalk. [27] A man holding up two bicycles near a park. [28] The guy does not know what to do with the two bikes [29] A man is trying to steady to bicycles. [30] A man holding two bicycle on each hands. [31] A man holding onto two bicycles near sand. [32] The man has two bicycles. [33] A man holding some bikes. [34] A man is holding two bikes by their handlebars [35] A man touching two bikes outside. [36] A man holds two bikes by the handlebars. [37] A man is holding the handle of two bicycles. [38] A young man holding two

bikes with jeans on [39] A man is trying to hold up two bicycles. [40] A man holding the handles of two separate bikes. [41] A man is examining two bicycles. [42] A man is holding onto two bicycles. [43] guy holding two bikes on the beach [44] A man holds the handlebars of two bikes on a beach. [45] A man is holding onto two bikes. [46] A man holds up two bikes. [47] There is a man holding on to two bicycles. [48] The man holds two bicycles near the beach. [49] A man is holding onto two bicycles. [50] A man is holding two bikes.



[1] A woman with a small dog on a leash. [2] A standing woman holding a leash to a puppy. [3] A woman is holding a small dog on a lead. [4] A woman walks a small dog [5] A woman stands with a leashed dog attached to her belt. [6] A woman has her dog on the leash. [7] A woman holding the leash of a toy dog. [8] A woman holding a leash with her puppy dog on the end of it. [9] A woman stands on a patio with a puppy on a leash. [10] A woman is walking her dog. [11] A women walking a tiny dog. [12] A woman with a pink belt and red hair holding a blue leach with a small puppy attached. [13] A woman has a small dog on a leash [14] A girl walks a black and white puppy. [15] A girl stands holding a small puppy on a leash [16] a women walking her dog [17] A girl walking a dog with a leash. [18] A woman holding a little dog by a leash. [19] A young woman holds her small dog at the end of a leash. [20] A woman holding her dog attached

to a leash. [21] A girl is standing with a pointer puppy on a leash. [22] A woman walks a small black and white dog on a leash. [23] A woman is holding a small dog on a leash. [24] a women standing out side with her dog [25] A young women walks her small puppy. [26] A woman, wearing a gray jacket, walking her small dog [27] Woman standing with a small dog. [28] girl holding a dog on a leash [29] A woman with red hair is walking a small black/white puppy. [30] Inside, a girl has a small dog's leash attached to her belt. [31] A redhead girl in jeans is standing with her black and white dog on a leash. [32] A lady walks her small dog outside. [33] A girl has her puppy on a leash. [34] A woman has a puppy on a leash/ [35] A woman holding a black and white dog on a leash. [36] A girl has a small dog on a leash tied to her belt loop. [37] The girl has a dog on a leash. [38] The woman is walking her dog [39] A girl walking her dog. [40] A woman with a dog on a leash. [41] A girl holds back a curious puppy on a leash. [42] The woman has a small dog on a leash. [43] A lady and a dog. [44] A woman poses with her little dog on a leash. [45] A woman is walking a tiny dog on a leash. [46] A woman walking a small dog on a blue leash. [47] A woman is walking her dog on the patio. [48] A red headed woman with a small dog on a leash. [49] A girl holds a small dog on a blue leash [50] a girl walking a dog with the leash attached to her belt



[1] A car is driving behind a bicyclist. [2] A bicyclist being followed by a car with two bicycles on its roof [3] A cyclist in a race being trailed by a car. [4] A cyclist rides ahead of a car. [5] A bicycler climbing a steep hill. [6] A biker is competing in a race. [7] A bicycleist is riding in front of a car [8] Cyclist racing with his team car following him. [9] Biker riding down street being followed by a car. [10] Man rides his bike in front of a silver car with two bikes on top of it. [11] A man is photographing a cyclist from a car behind him. [12] A man is riding a bicycle on the street. [13] A man on a bicycle being following by a car with two bikes on its bike rack. [14] A man riding a bike in front of a car. [15] A cyclist is in front of a car [16] a bike race on a road [17] A man is riding his bicycle in front of a car that has two more bicycles on top of it. [18] A man riding a bike is followed by a car carrying two bikes on its roof [19] A cyclist biking in front of a car. [20] A man rides a bike in front of a car with two bikes on the roof. [21] Man riding a bike in front of a car [22] A red clad cyclist is riding in front of a silver car. [23] A man is riding in a bike race. [24] a man riding a bike down the street with a car driving behind him [25] A bicyclist is being followed by a car. [26] A man in a red bike outfit followed by a car with several bikes. [27] A bicyclist rides in front of the car. [28] A cyclist races on the road with a car behind him. [29] A car behind a man on a bike. [30] A biker is riding down the street. [31] A bicyclist moves down the street at high speeds. [32] A man rides his bike along the road. [33] A man is riding a bike while being followed by a car. [34] This is a man on a bike. [35] A man is riding a bike dressed like a racer and is followed by a car carrying 2 extra bikes. [36] A man rides his bike in front of a car moving behind him really fast [37] A pro bike racer that is competing. [38] A cyclist rides ahead of a car. [39] The car is following the cyclist. [40] A man riding a bicycle in front of a car. [41] The cyclist is in the middle of a marathon [42] The bicycle rider is in front of the maintenance car. [43] A man on a bike with a car with bikes on it A man

on a bike with a car with bikes on it's roof behind him. [44] A man is riding a bicycle down the road. [45] The bike rider is followed by car. [46] A bicyclist is filmed by a team in a moving car. [47] A man rides a bicycle with a car following behind. [48] A man on a bike with a car behind him. [49] A man wearing racing gear and pedaling a bike in front of a car. [50] A person riding a bicycle in front of a car with two bicycles on top of it.