Documentation

Name: Jaykumar Maheshbhai Bhingaradia
Email: bhingaradia.j@northeastern.edu
Mobile No: +1 (857) 300 - 3551

(Click in this link if you want to go for question/ answer)

Objective:

1. Research work is generally scattered and focusses on generic problems in various domains. One natural way to begin your work is to formulate your task and see if there is existing research literature out there that could be used. You also need to formulate better so that you can have efficient correspondence with people (your professors, researchers, amazing folks on Stack exchange or Reddit etc.).

How would you formulate this feature/product/problem technically? Please provide references where necessary.

2. Neural networks are theoretically universal function approximators, so it only makes sense to wield their tremendous abilities for this product given their successful application in diverse challenging tasks with various inputs.

How would you go about structuring this task as a deep learning task?
Link 1
Link 2
Link 3

3. If you observed, the data we have is not labelled. One of the limitations machine learning practitioners face is the lack of labelled data they need for supervised learning.

As an engineer, what is your solution to this? Do you think we need to get labelled/annotated data? If so, what exact labels/annotations do you need?

4. As you know, not all problems need labelled/annotated data. Also, given the abundant supply of unlabeled data, it is tempting to build your solution with unsupervised learning.

Can you formulate this as an unsupervised learning problem? If yes, how? If not, why?

5. If you choose to make a bold claim that this cannot be solved with Machine Learning and that rule based approach is the only/efficient solution, please explain. Otherwise you can skip it unless you want to propose a good rule-based solution we can rather consider.

Please answer with as much clarity as you can summon. No penalty on being verbose, but it's a bonus to have smart responses. Please don't be vague in your responses We want to see your problem-solving approach, research aptitude, knowledge and communication.

Answers:

Demo:

First task is to **convert our skeleton slide to JSON files.**

Code:

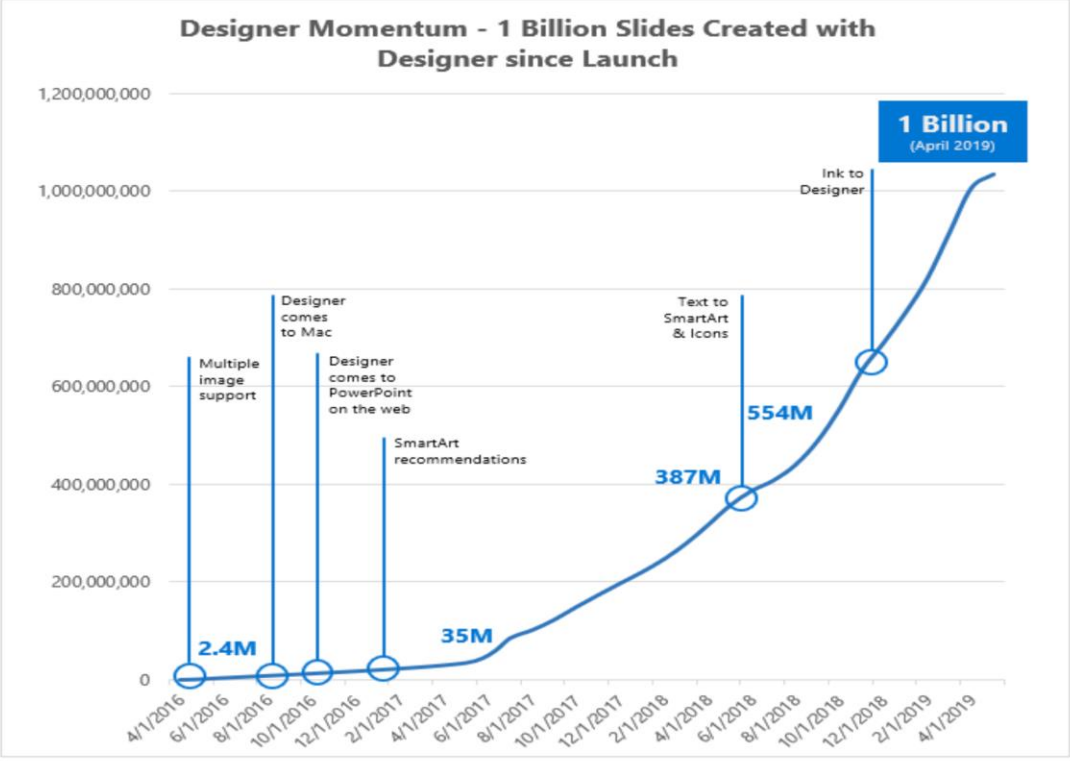
```
import json

local_pptxFileList = ["/content/drive/MyDrive/Slides/Backlog Management.pptx"]

all_texts = []
for i in local_pptxFileList:
    ppt = Presentation(i)
    this_pres_texts = []
    for slide in ppt.slides:
        for shape in slide.shapes:
            if shape.has_text_frame:
                this_pres_texts.append(shape.text)
    all_texts.append(this_pres_texts)

with open('data.txt', 'w') as outfile:
    json.dump(all_texts, outfile)
```

Store the extracted texts into a data structure such as a list (or list of lists, with one list for each presentation's texts).

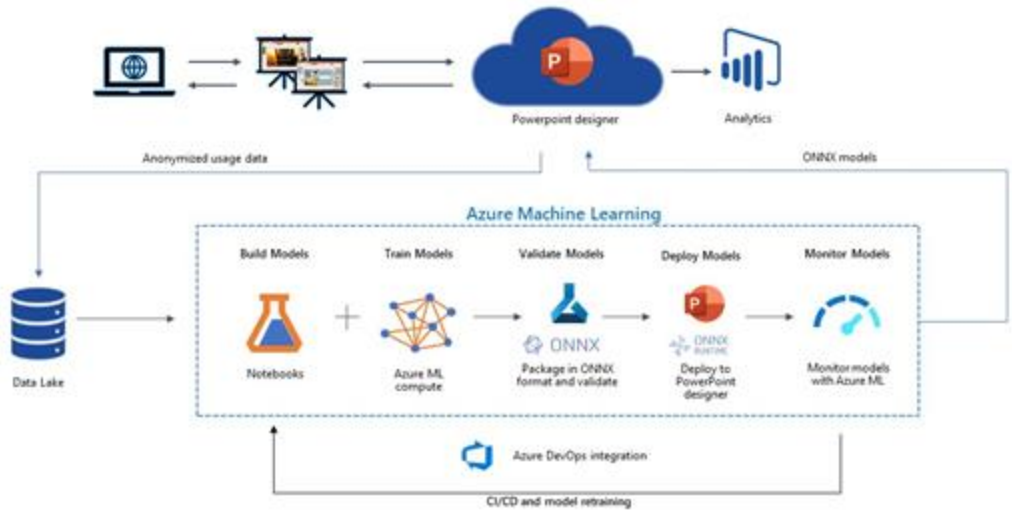
Use JSON module to create a JSON from your data structure and save to a file. I haven't dealt with encoding (e.g., as utf-8) to ensure that texts are correctly stored, but there's plenty of info about that you can find easily.

I take reference of PowerPoint Designer effort which is Artificial tool provided by Microsoft.

Let’s we talk about journey to make PPTx beautify Automatically.



Let’s we divide task into small part then try to implement each part to make whole ppt making process Automation base.



Get intelligent design recommendations and stay on brand with Designer for branded templates

One of our biggest asks from customers has been to make Designer work seamlessly with a company's branded templates. With the launch of Designer for branded templates, companies can now make sure the design recommendations people see meet their corporate branding and visual identity guidelines.

When working with branded templates, Design Ideas that people see are created directly from the layouts in the Slide Master, using the content on the user's slides. Designer's AI chooses the most suitable layouts for the content, intelligently crops images, and automatically recommends relevant icons and pictures. Organizations can design their templates for optimal Designer support, broadening the choices available to their users.

How it works:

1. Content recognition from JSON Dataset.

Best suitable image or template this by using natural language processing (NLP) and Data Science modeling (NMF, Naive Bayes) on JSON data to interpret slang meanings of Image and predict what image or slide templet would go best with your next client PPT.

Collecting the Data

- Collect Data from JSON file
- Use AWS to run the API using EC2 and automatically save the JSON file data to a bucket on S3

Cleaning & processing the data

- Identify JSON data that contain some key word which is most important for that slide
- Process the JSON dataset and use the image or slide templet which is available as labels
- Vectorize the JSON dataset using tfidf and count-vectorizer

Modeling

- Utilize NMF and topic modeling to cluster similar JSON dataset to identify image or slide templet that have similar meanings
- Create an image or slide templet predictor using Naive Bayes
- Identify words commonly associated with each emoji
    - Extracted from the NMF and the Naive Bayes models

Model Validation

- Test on a subset of JSON dataset that the model was not trained on

2. Image selection according contain token

Second things we need is image processing. Because once we recognize token and machine understand the meaning then after we need to decide which format, we should take to build our power point slide.

Once we find out token from the JSON file. We must map that token contain with images or pattern slide which is present in internet. So, first I recognize image and make classified using CNN algorithm.

For that we can used CNN a Machine Learning algorithm called Convolution Neural Network (CNN), it is a common Deep Learning technique used for image recognition and classification.

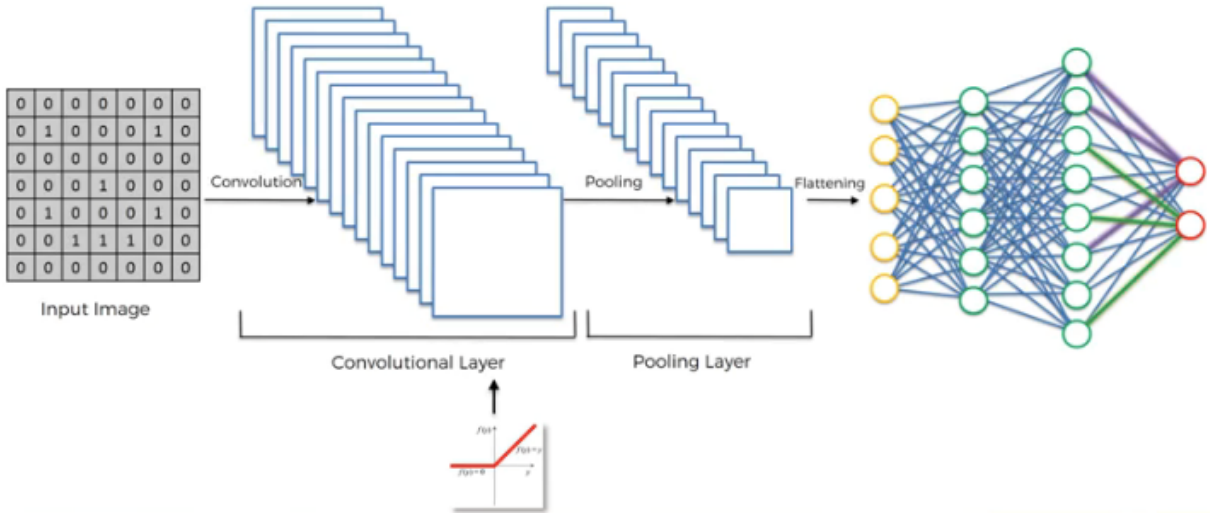
Let’s imagine that our dataset for images is internet, we should identified image by considering this dataset.

Convolution Neural Networks are good for pattern recognition and feature detection which is especially useful in image classification. Improve the performance of Convolution Neural Networks through hyper-parameter tuning, adding more convolution layers, adding more fully connected layers, or providing more correctly labeled data to the algorithm.

Create a Convolution Neural Network (CNN) with the following steps:

1. Convolution
2. Max Pooling
3. Flattening
4. Full Connection

For image recognition, we convolve the input image with Feature Detectors (also known as Kernel or Filter) to generate a Feature Map (also known as Convolved Map or Activation Map). This reveals and preserves patterns in the image and compresses the image for easier processing. Feature Maps are generated by element-wise multiplication and addition of corresponding images with Filters consisting of multiple Feature Detectors. This allows the creation of multiple Feature Maps.



This is when the output of a Convolution Neural Network is flattened and fed through a classic Artificial Neural Network. It's important to note that CNNs require fully connected hidden layers whereas regular ANNs don't necessarily need full connections.

Here is a summary of every step of a CNN, don't forget about the Rectifier Function that removes linearity in Feature Maps, also remember that the hidden layers are fully connected.

Alternative way,

3. DALL·E by Open AI

A neural network called DALL·E that creates images from text captions for a wide range of concepts expressible in natural language.

DALL·E is a 12-billion parameter version of GPT-3 trained to generate images from text descriptions, using a dataset of text–image pairs. We've found that it has a diverse set of capabilities, including creating anthropomorphized versions of animals and objects, combining unrelated concepts in plausible ways, rendering text, and applying transformations to existing images.

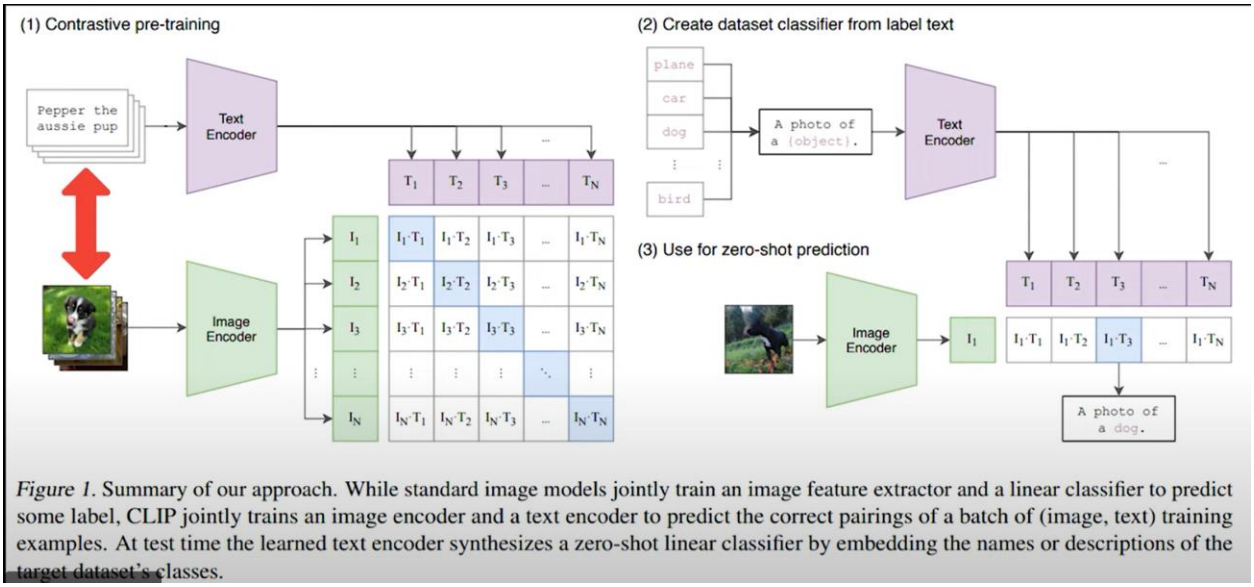


Figure 1. Summary of our approach. While standard image models jointly train an image feature extractor and a linear classifier to predict some label, CLIP jointly trains an image encoder and a text encoder to predict the correct pairings of a batch of (image, text) training examples. At test time the learned text encoder synthesizes a zero-shot linear classifier by embedding the names or descriptions of the target dataset's classes.

We're releasing an API for accessing new AI models developed by Open AI. Unlike most AI systems which are designed for one use-case, the API today provides a general-purpose "text in, text out" interface, allowing users to try it on virtually any English language task.

You can now request access in order to integrate the API into your product, develop an entirely new application, or help us explore the strengths and limits of this technology.

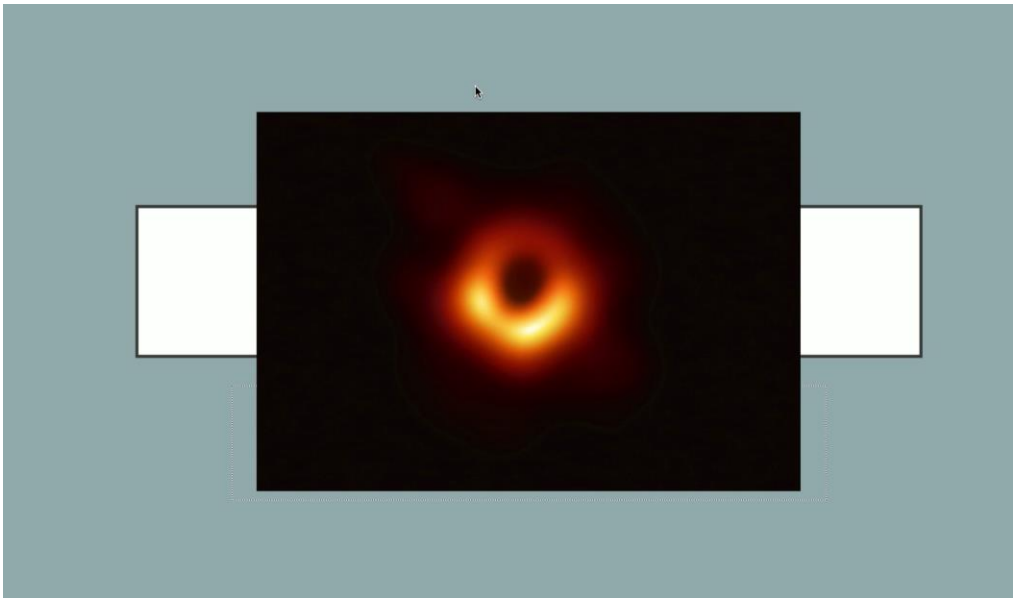
We've designed the API to be both simple for anyone to use but also flexible enough to make machine learning teams more productive. In fact, many of our teams are now using the API so that they can focus on machine learning research rather than distributed systems problems. Today the API runs models with weights from the GPT-3 family with many speed and throughput improvements. Machine learning is moving very fast, and we're constantly upgrading our technology so that our users stay up to date.

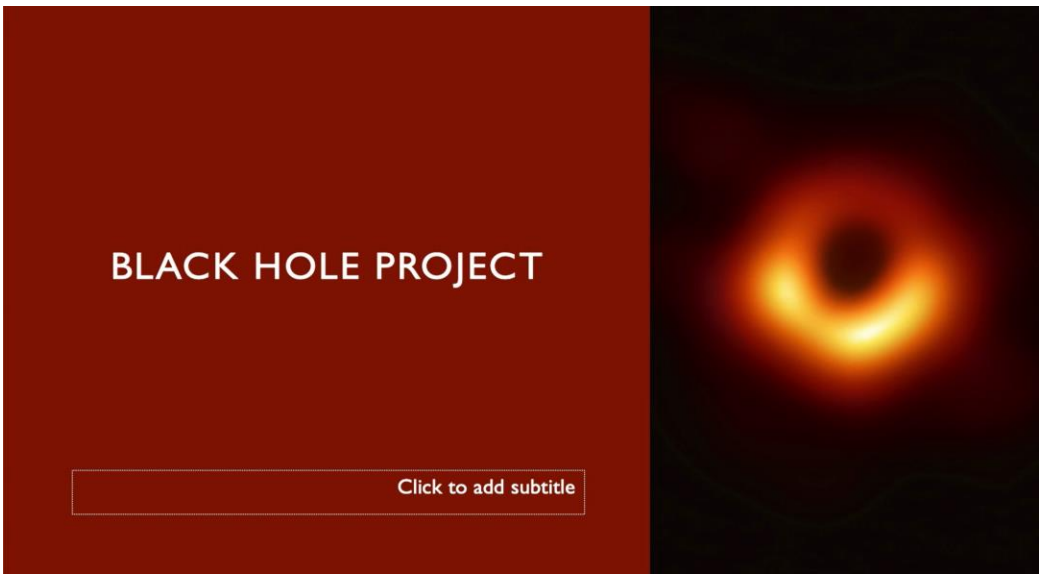
Input: Text (Given by user) – [A teddy bear on a skateboard in Times Square]

Output: generated image base on input contains using API

Once we developed image using this API. Now our next step will be to place this picture and slide background into PPTx. This make our PPTx beauty this slides.

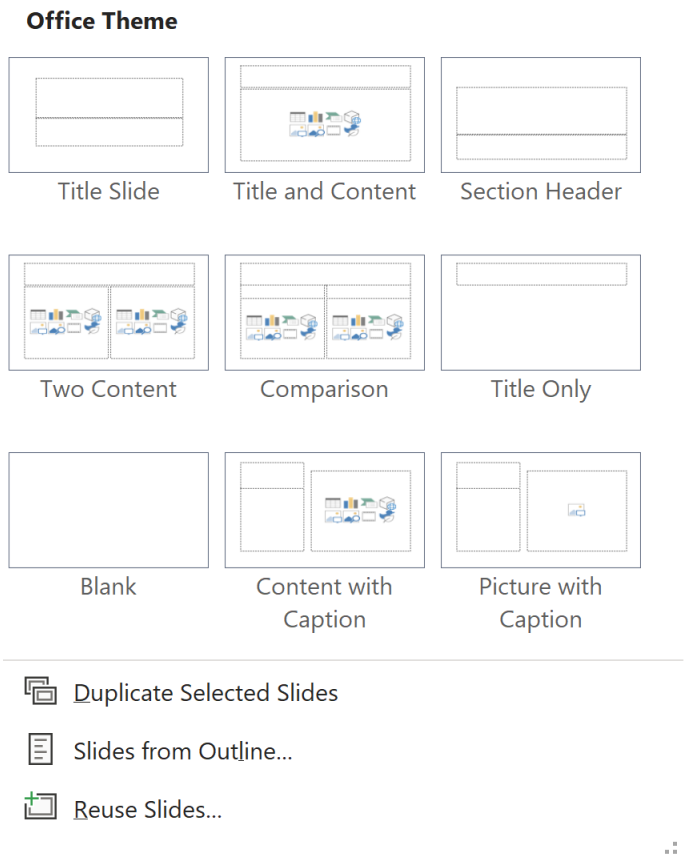
4. Arrangement of contains on the PowerPoint:





How we can create best match layout according with our JSON contains. So, we can map this component with right place in the PPTx to developed professional version.

So, what we can do? We take help from existing format which is present in Microsoft power point software.



In our JSON file we divided contain mainly 2 parts.

1. shape (Image, Chart, Table)

2. text (Header, Bullet points)

We can identified contain attribute by there JSON component then we can relate each component with best place where we can place in the diagram.

I want to convert JSON data to string

```
import java.io.BufferedReader;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL.
import org.json.JSONArray;
import org.json.JSONObject;
public static void main(String[] args) throws Exception
```

```
{

   URL url = new URL("http://192.168.1.13/test/ProductWb.php?productId=9");
   HttpURLConnection conn ;
   conn = (HttpURLConnection) url.openConnection();
   conn.setRequestMethod("GET");
   conn.setReadTimeout(60);
   conn.setRequestProperty("Accept", "application/json");
   String json="";

   json = readUrl(conn);
         System.out.println(json);
         JSONObject jsonObject=new JSONObject(json);
         JSONArray jarray=jsonObject.getJSONArray("modeles");
         JSONObject  modele= jarray.getJSONObject("modele");
         for (int i=0;i<modele.length();i++) {
            System.out.println(modele(i).getString("id_product"));
             System.out.println(modele(i).getString("meta_title"));
             System.out.println("*********");
           }}
```

5. Theme selection:

What am I thinking? so when we got any image or any slide layout. How should we know this color is best suitable for this model so we can do this by identified most spread color present in image.

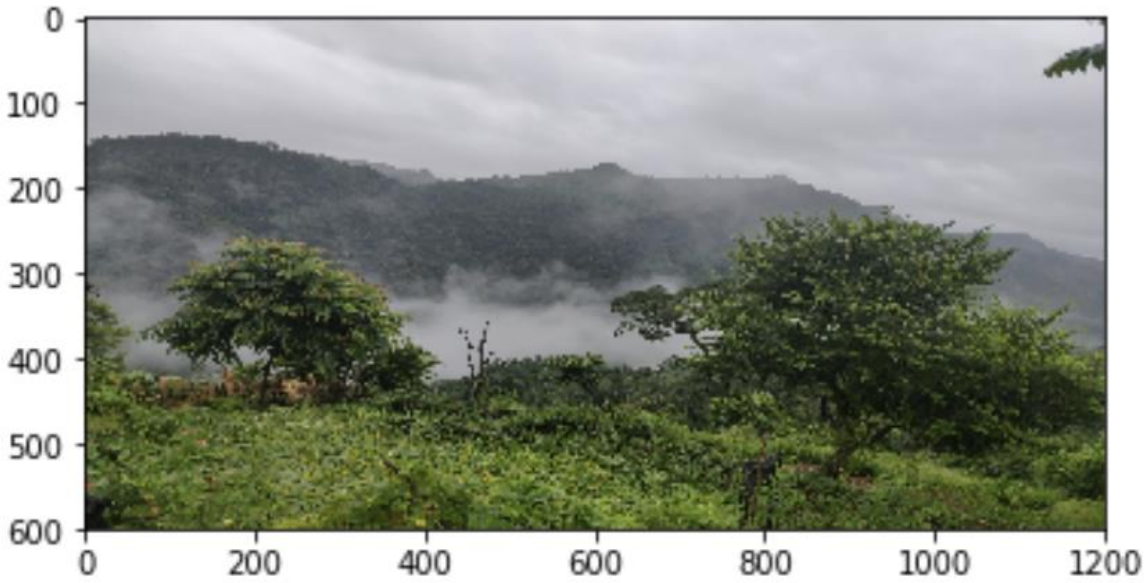
I recently started reading about how I could work with Images in Python. When I came across OpenCV which allows import and manipulation of images in Python, I started to wonder if information could be extracted out of those images using Machine Learning and used in some way.

We've all seen that we can search online on the basis of certain filters one of which is *color*. I got inspired to actually write the code that can extract colors out of images and filter the images based on those colors.

Import Library:

```
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import numpy as np
import cv2
from collections import Counter
from skimage.color import rgb2lab, deltaE_cie76
import os

%matplotlib inline
```

Input image:



Get colors from an image

```
  counts
  Counter(labels)
```

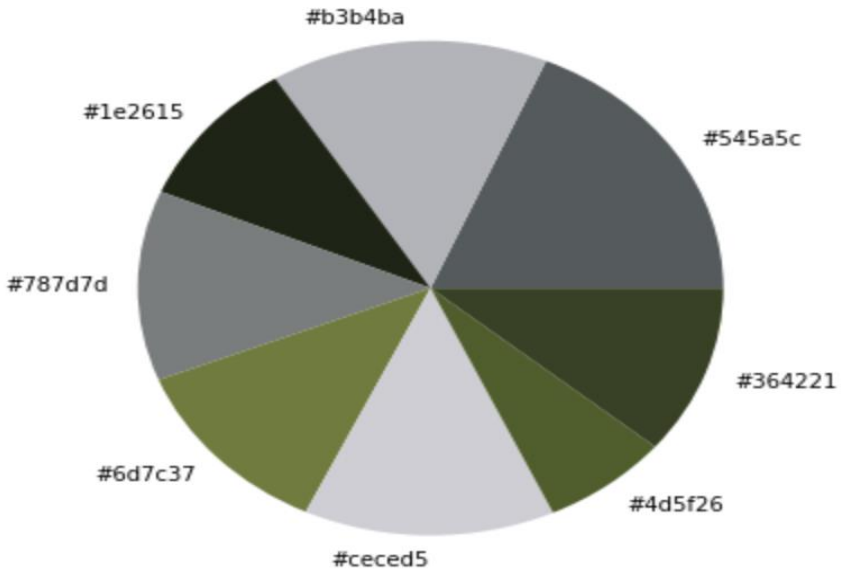
```
                center_colors = clf.cluster_centers_
                # We get ordered colors by iterating through the keys
                ordered_colors = [center_colors[i] for i in counts.keys()]
                hex_colors = [RGB2HEX(ordered_colors[i]) for i in counts.key
                rgb_colors = [ordered_colors[i] for i in counts.keys()]

                if (show_chart):
                    plt.figure(figsize = (8, 6))
                    plt.pie(counts.values(), labels = hex_colors, colors = hex_colo

                return rgb_colors
```

image: The image whose colors we wish to extract.

number_of_colors: Total colors we want to extract.

show_chart: A boolean that decides whether we show the pie chart or not.

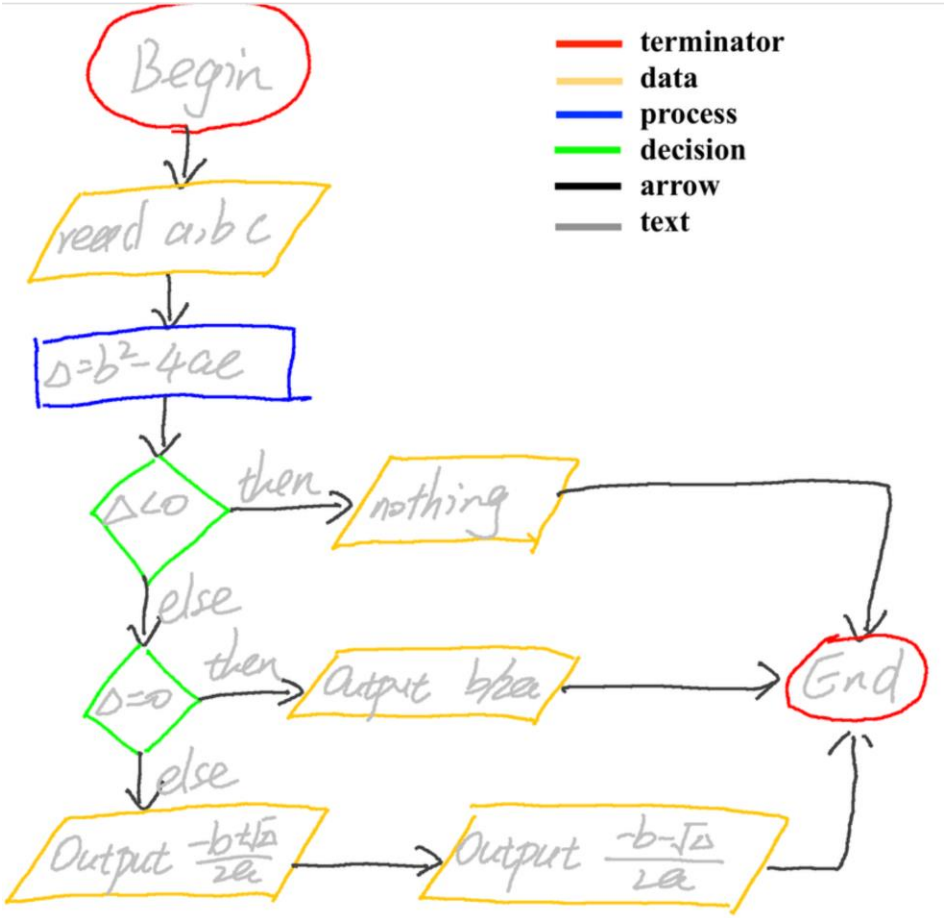
Output:



Output of this image we get this type of color combination was used in this image.

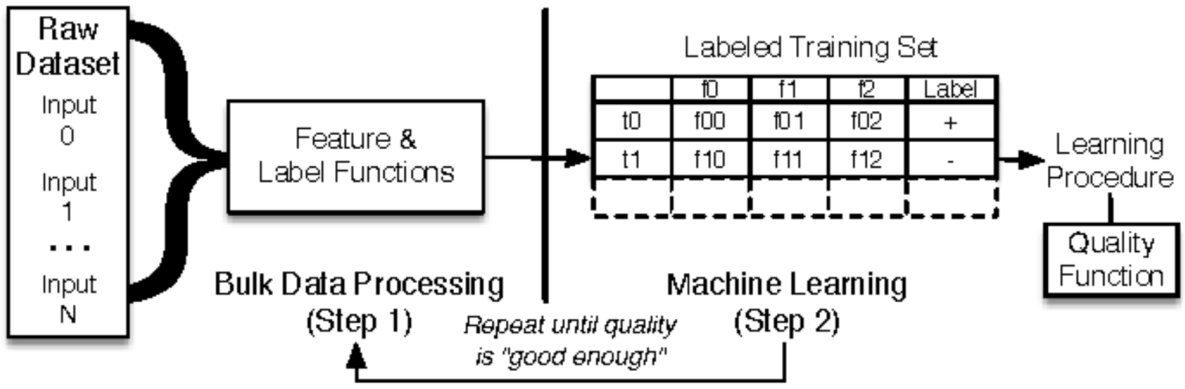
Now, we make (#545a5c), this color as theme of our power point PPtx for our slide. So, we find our theme color.

6. Diagram generator by using handwritten



Machine Learning Based Approach

This methodology involves the Image Processing, Feature Extraction and ML Model building. Model generated will be more optimal in terms of Memory footprints and Time Complexity when compared to CNN based model.

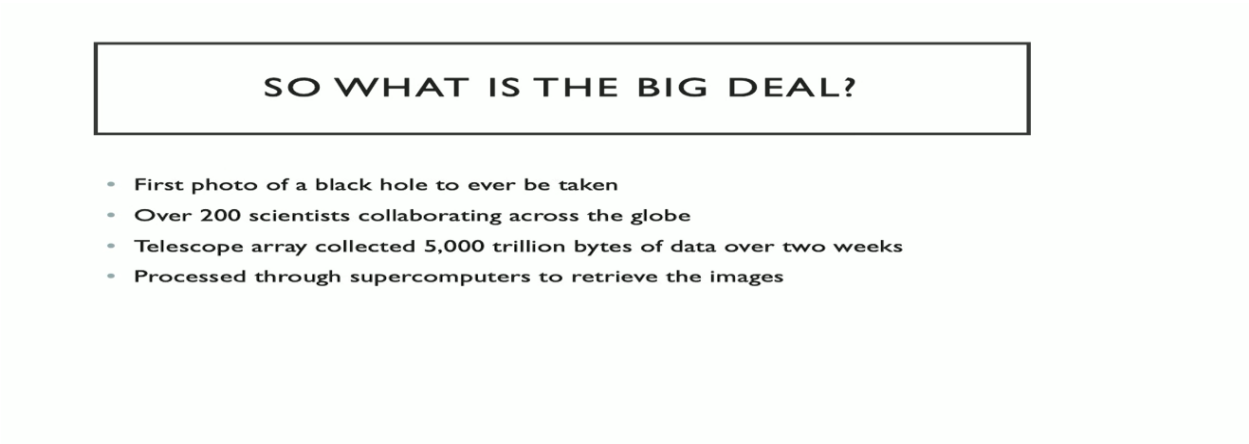



```
import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread('noisy2.png',0)

# global thresholding
ret1,th1 = cv2.threshold(img,127,255,cv2.THRESH_BINARY)

# Otsu's thresholding
ret2,th2 = cv2.threshold(img,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)

# Otsu's thresholding after Gaussian filtering
blur = cv2.GaussianBlur(img,(5,5),0)
ret3,th3 = cv2.threshold(blur,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)

# plot all the images and their histograms
images = [img, 0, th1,
          img, 0, th2,
          blur, 0, th3]
titles = ['Original Noisy Image','Histogram','Global Thresholding (v=127)',
          'Original Noisy Image','Histogram',"Otsu's Thresholding",
          'Gaussian filtered Image','Histogram',"Otsu's Thresholding"]

for i in xrange(3):
    plt.subplot(3,3,i*3+1),plt.imshow(images[i*3],'gray')
    plt.title(titles[i*3]), plt.xticks([]), plt.yticks([])
    plt.subplot(3,3,i*3+2),plt.hist(images[i*3].ravel(),256)
    plt.title(titles[i*3+1]), plt.xticks([]), plt.yticks([])
    plt.subplot(3,3,i*3+3),plt.imshow(images[i*3+2],'gray')
    plt.title(titles[i*3+2]), plt.xticks([]), plt.yticks([])
plt.show()
```

Image Processing steps will improve and optimize the image quality in terms of Resolutions, Pixel Densities, Black level Improvements, White Level Improvements etc. The Segmented patched now can be used for the feature engineering & feature extractions.

1. **Height**: This feature generates the height for each of the image segments or patch.
2. **Width**: This feature generates the height for each of the image segments or patch.
3. **Aspect Ratio**: It is a number which represents the ration height with the width for each of the image segments or patch. (AR = W/H)
4. **Black Pixel Density:** This represent the density of the black pixels available in a patch on the foreground white plane.
5. **White Pixel Density:** This represent the density of the white pixels available in a patch on the foreground Black plane.

7. Bullet point conversion:

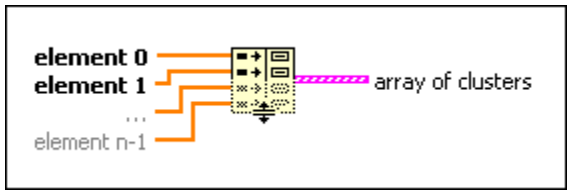


Labeling Data:

Why labeling is so important for JSON dataset. If we classified data by best parameter base on JSON data.
After this label data, we can make cluster then label name like chart data, table data, bullet point data, Title data, etc.

Once we identified our cluster and label name according to our output

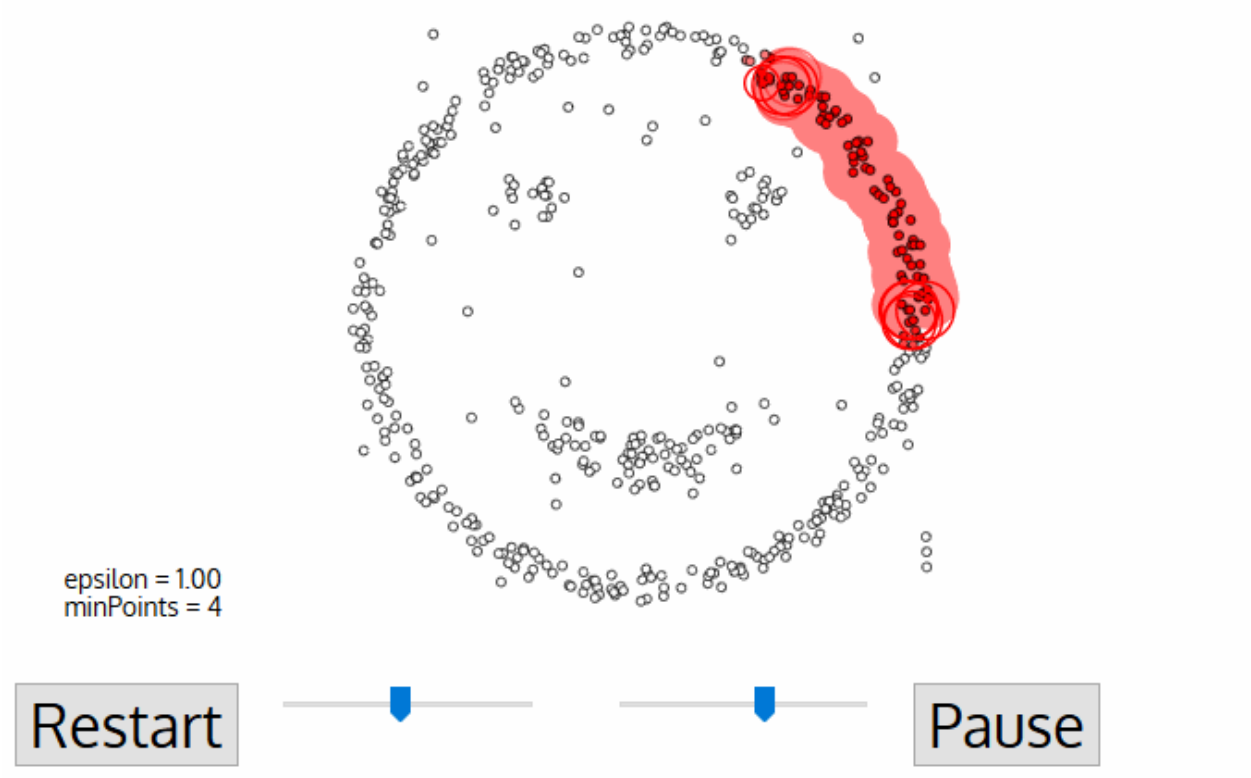
Input: JSON data (Object Type, Amount of Text,Position)
Output: Cluster



For that we can used DBSCAN model

DBSCAN Model

This algorithm is preferred for this task because it selects the number of clusters automatically, whereas for KNN we would need to specify the number of clusters.



DBSCAN is a density-based clustered algorithm similar to mean-shift, but with a couple of notable advantages. Check out another fancy graphic below and let's get started

1. DBSCAN begins with an arbitrary starting data point that has not been visited. The neighborhood of this point is extracted using a distance epsilon ε (All points which are within the ε distance are neighborhood points).

2. If there are a sufficient number of points (according to minPoints) within this neighborhood, then the clustering process starts and the current data point becomes the first point in the new cluster. Otherwise, the point will be labeled as noise (later this noisy point might become the part of the cluster). In both cases that point is marked as "visited".

3. For this first point in the new cluster, the points within its ε distance neighborhood also become part of the same cluster. This procedure of making all points in the ε neighborhood belong to the same cluster is then repeated for all of the new points that have been just added to the cluster group.

4. This process of steps 2 and 3 is repeated until all points in the cluster are determined i.e all points within the ε neighborhood of the cluster have been visited and labeled.

To make bullet point beautified, what we can do. Firstly, we should recognize the contain by NLP concept to understand text. After this text, we can find best suitable token from our available contain of token. This whole process is same process which we discuss on point number 1 and point 2. Here only one difference is icon prediction, and over there we predict image for slide.

Now,

second thing is to develop rectangular box surround the bullet point, for that we don't require any kind of machine learning approach. What we can do when we find any bullet point in out JSON file. We trigger one event. And generate checkbox surround the text box. Then for color selection we can select any auto-color selection API, or We can add color according to the theme bases.

Now, here I show How we can convert JSON data to PPTx chart, plot, bullet point beautified.

Bar Charts

```
const PPTX = require('nodejs-pptx');
let pptx = new PPTX.Composer();

let barChartData1 = [
  {
    name: 'Series 1',
    labels: ['Category 1', 'Category 2', 'Category 3', 'Category 4'],
    values: [4.3, 2.5, 3.5, 4.5],
  },
  {
    name: 'Series 2',
    labels: ['Category 1', 'Category 2', 'Category 3', 'Category 4'],
    values: [2.4, 4.4, 1.8, 2.8],
  },
  {
    name: 'Series 3',
    labels: ['Category 1', 'Category 2', 'Category 3', 'Category 4'],
    values: [2.0, 2.0, 3.0, 5.0],
  },
];

await pptx
  .compose(async pres => {
```

```
    await pres.layout('LAYOUT_4x3').addSlide(async slide => {
      await slide.addChart(chart => {
        chart
          .type('bar')
          .data(barChartData1)
          .x(100)
          .y(100)
          .cx(400)
          .cy(300);
      });
    });
  })
  .save('./chart.pptx');
```

Images

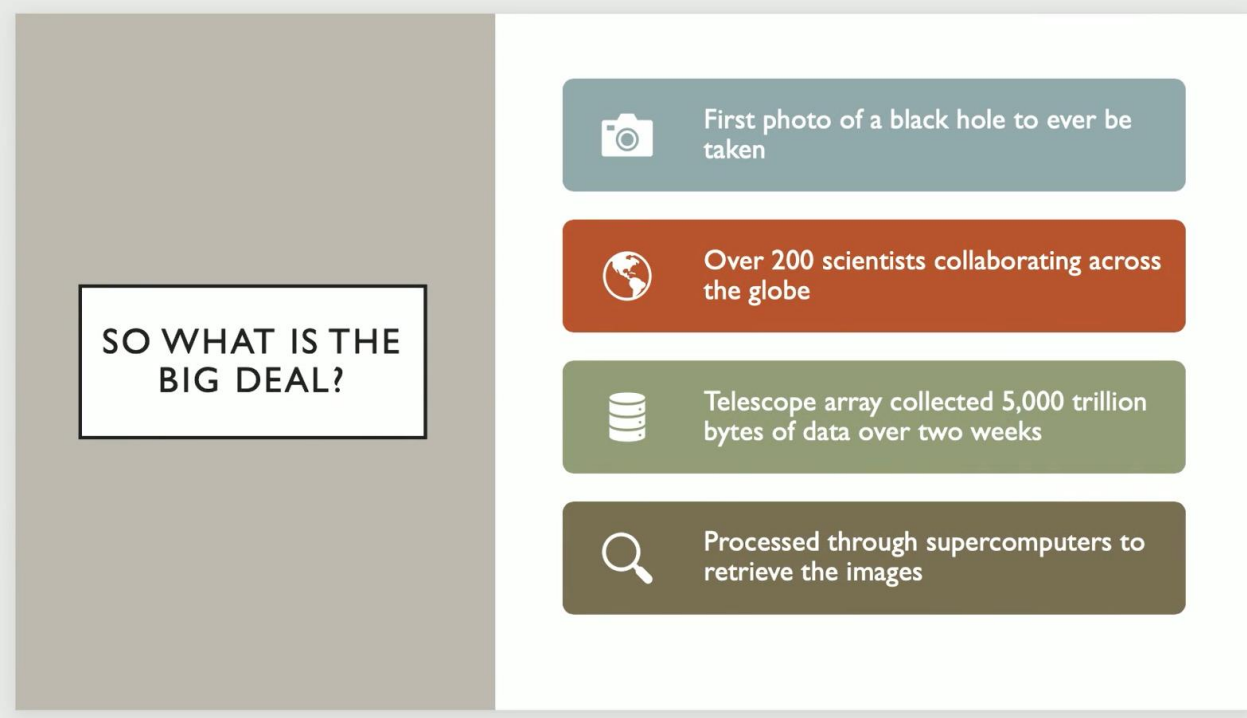
```
const PPTX = require('nodejs-pptx');
let pptx = new PPTX.Composer();

await pptx.compose(async pres => {
  await pres.addSlide(async slide => {
    // Images can be added locally
    slide.addImage(image => {
      image
        .file('./images/pizza.jpg')
        .x(100)
        .cx(200);
    });
    // Images can be downloaded from the internet.
    await slide.addImage({
      src:
'https://www.google.com/images/branding/googlelogo/2x/googlelogo_color_120x44dp.png',
      href: 'https://www.google.com',
      x: 10,
      y: 400,
      cx: 50,
    });

    // Images can be added inline as a base64 encoded string.
```

```
    slide.addImage(image => {
      image
        .data('iVBORw0KGgoA[...]Jggg')
        .x(350)
        .y(200);
    });
  });
});
```

Text Boxes

As the name suggests text can be added to the slide using addText. The text box element also supports the creation of external links (which open a web browser) and internal linking (which link to another slide in the same presentation).

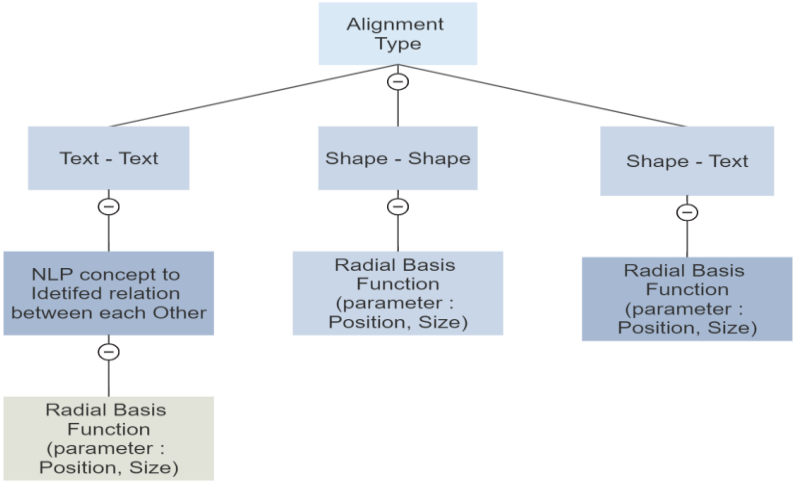
```
const PPTX = require('nodejs-pptx');
let pptx = new PPTX.Composer();

await pptx.compose(async pres => {
  await pres.addSlide(slide => {
    // declarative way of adding an object
    slide.addText(text => {
      text
        .value('Hello World!')
        .x(100)
        .y(50)
        .fontFace('Alien Encounters')
        .fontSize(20)
        .textColor('CC0000')
        .textWrap('none')
        .textAlign('left')
        .textVerticalAlign('center')
        .line({ color: '0000FF', dashType: 'dash', width: 1.0 })
        .margin(0);
    });
    // plain "config" method of adding an object
    slide.addText({ value: 'Link to google.com', x: 200, y: 300, href: 'http://www.google.com' });
  });
});
await pptx.save(`./text-box-new-simple.pptx`);
```



8. Aligning two components:

Object alignment can split into these three categories.

1. Text – Text
2. Shape – Shape
3. Shape – Text

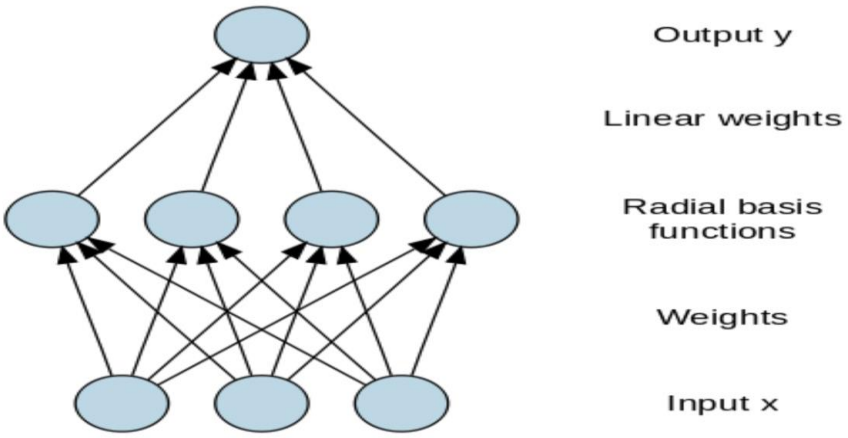


The Aim of this mapping is how to proceed with align components. This is classification problem. But each case handle differently.

In our first scenario, Text – Text -> we must determine text contained in each box is related to each other. This can be done through NLP techniques like Topic modeling. So, text belonging to same topics will be aligned together.

Once we determined positive relation between text, we can use Radial Basis Function (RBS) neural Network to determined axis along which alignment need to be made.

For rest of two category, it's not depend on any kind of contents. Only parameter significant are related to proximity, so Radial Basis Function is sufficient to determine the axis for alignment



References:

1. Barak, Erez. “How Azure Machine Learning Enables PowerPoint Designer | Azure Blog and Updates | Microsoft Azure.” *How Azure Machine Learning Enables PowerPoint Designer*, azure.microsoft.com, https://azure.microsoft.com/en-us/blog/how-azure-machine-learning-enables-powerpoint-designer/. Accessed 18 May 2022.

2. “DALL·E APIs | API Tracker.” *DALL·E APIs | API Tracker*, apitracker.io, https://apitracker.io/a/openai-dall-e/apis. Accessed 18 May 2022.

3. Chandradevan, Ramraj. “Radial Basis Functions Neural Networks — All We Need to Know | by Ramraj Chandradevan | Towards Data Science.” *Medium*, towardsdatascience.com, 18 Aug. 2017, https://towardsdatascience.com/radial-basis-functions-neural-networks-all-we-need-to-know-9a88cc053448.

4. “OpenAI’s DALL-E Creates Plausible Images of Literally Anything You Ask It to – TechCrunch.” *TechCrunch*, techcrunch.com, https://techcrunch.com/2021/01/05/openais-dall-e-creates-plausible-images-of-literally-anything-you-ask-it-to/?guccounter=1&guce_referrer=aHR0cHM6Ly93d3cuYmluZy5jb20v&guce_referrer_sig=AQAAAKxqqci-JysRfl9clQoVI6eBhAKsuaB__yFJ2m1-iuNogHasfbH43NRNt92z2Z0H2MZaZnDg69ncG3sJE_R2LOMk91SK6NAX52MG9EL-MQNEjYxaHiU7xWBq8sfE6W8lchAmi0SHHM6IVdFmYvshtq2gv0fXm2HyROlfnhZmCMKa. Accessed 18 May 2022.

5. “How to Convert a Json Data to String in Java - Stack Overflow.” *Stack Overflow*, stackoverflow.com, 7 Apr. 2014, https://stackoverflow.com/questions/22914808/how-to-convert-a-json-data-to-string-in-java.

6. “Intelligently Extract Text & Data with OCR - Amazon Textract - Amazon Web Services.” *Amazon Web Services, Inc.*, aws.amazon.com, https://aws.amazon.com/textract/. Accessed 18 May 2022.

7.Bhanot, Karan. “Color Identification in Images—Machine Learning Application.” *Medium*, towardsdatascience.com, 24 May 2019, https://towardsdatascience.com/color-identification-in-images-machine-learning-application-b26e770c4c71.

8. “KNN Algorithm - Finding Nearest Neighbors.” *KNN Algorithm - Finding Nearest Neighbors*, www.tutorialspoint.com, https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_knn_algorithm_finding_nearest_neighbors.htm. Accessed 18 May 2022.

9. Seif, George. “The 5 Clustering Algorithms Data Scientists Need to Know | by George Seif | Towards Data Science.” *Medium*, towardsdatascience.com, 11 Feb. 2022, https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68.

10. Gao, Peter. “How To Set Up An ML Data Labeling System | by Peter Gao | Towards Data Science.” *Medium*, towardsdatascience.com, 5 May 2021, https://towardsdatascience.com/how-to-set-up-an-ml-data-labeling-system-4eea9b15181f#:~:text=To%20get%20labeled%20training%20data%2C%20ML%20teams%20often,boxes%20on%20images%20for%20hours%20at%20a%20time.

11. SmashingQuasar. “GitHub - SmashingQuasar/Beautified-Select: A Simple Way to Embellish Your Tag without Ruining Your DOM!” *GitHub*, github.com, 31 Dec. 2020, https://github.com/SmashingQuasar/beautified-select.

12. “PowerPoint AI Gets an Upgrade and Designer Surpasses a Major Milestone of 1 Billion Slides.” *Microsoft 365 Blog*, www.microsoft.com, 18 June 2019, https://www.microsoft.com/en-us/microsoft-365/blog/2019/06/18/powerpoint-ai-upgrade-designer-major-milestone-1-billion-slides/.