Work Done

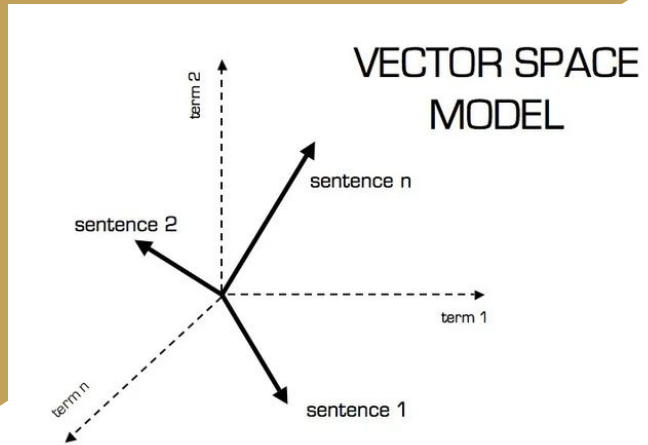
- Understanding the algorithms
- Writing the algorithms
- Performed Complexity Analysis
- Implemented code from scratch
- Implemented tf-idf to make a text based recommendation system.
- Implemented Convolutional Neural Network
- Made an image based recommendation system
- Apparel Recommendation System

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

TF-IDF

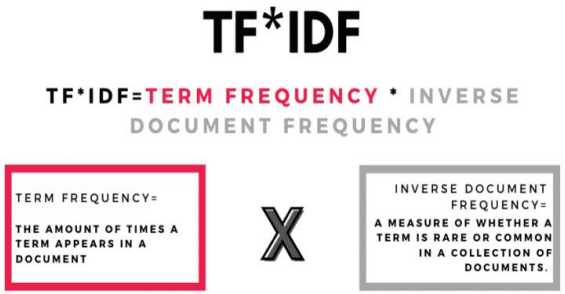
Term x within document y

$tf_{x,y}$ = frequency of x in y
df_x = number of documents containing x
N = total number of documents



Term Frequency-Inverse Document Frequency (TF-IDF)

- A technique to quantify words in a document



TERM FREQUENCY

- It refers to the frequency of a word in a document.

$$tf(t,d) = \frac{count(t)\ in\ 'd'}{count\ of\ word\ in\ 'd'}$$

DOCUMENT FREQUENCY

- Count of the occurrences of a term ‘t’ in the document (Corpus) ‘N’.

df(t) = occurrence of ‘t’ in ‘N’ documents

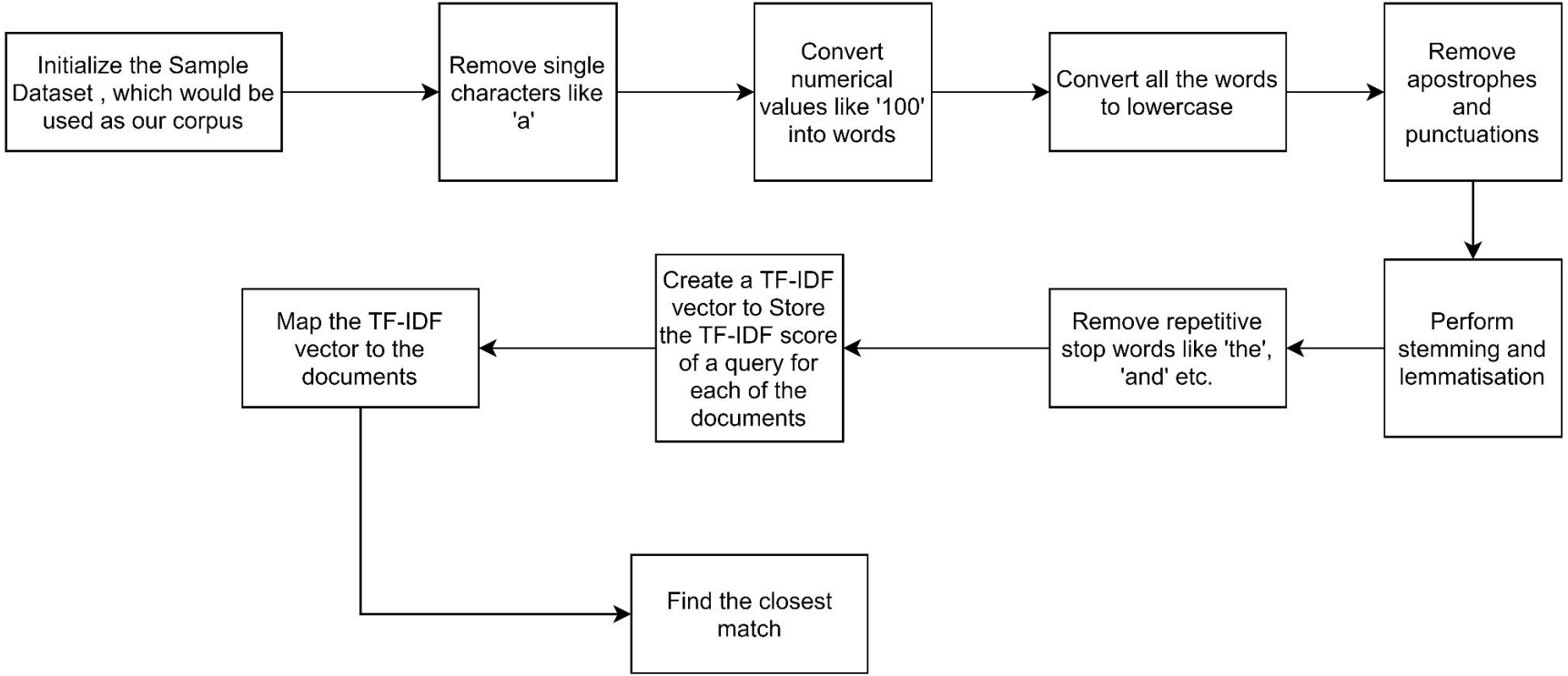
INVERSE DOCUMENT FREQUENCY

- Informativeness of a term ‘t’ in the corpus set ‘N’.
- Decrease relevance of stop words like ‘the’, ‘and’ etc.

$$idf(t) = \log_{10}(\frac{N}{df+1})$$

$$tf - idf(t,d) = tf(t,d) \times \log_{10}(\frac{N}{df+1})$$

Implementing TF-IDF on A Sample Dataset



Algorithm For TF-IDF

```
Algo tfidf
Let the total number of documents = M
Let the number of words in one document = N
Query term → 't'
doc[1, 2, 3, … , M][1, 2, 3, … , N]  //An array to traverse the words in the corpus

 count1 = 0 ;                          // Count for Term Frequency
 count2 = 0 ;                          // Count for Document Frequency
 for i = 1 to M
      for j = 1 to N
           if (doc[i][j] == 't')       // to check if word being traversed is 't'
             {
               count1++;              // counts no. of times 't' occurs in 'ith' document
                 }
      if (count1 > 0)          //executes when 't' occurs even once in 'ith' document
              count2++;
      tf [ i ] = (Count1/N);          // provides occurrences of 't' in 'ith' document
      count1 = 0;
df = count2;                          // provides the Document Frequency
idf = log (M/(df+1));
for k = 1 to M
      tfidf [ k ] = tf [ k ] * idf ;          // provides tf-idf score for each document
map (tfidf [1, 2, 3, … , M] → doc [1, 2, 3, … , M] ) ;
sort (tfidf [1, 2, 3, … , M ]) ;
// Now, we have obtained the documents in the order of increasing relevance
// The final document doc [ M ] is the most relevant one
Return ( doc [ M ] )
```

INTUITION

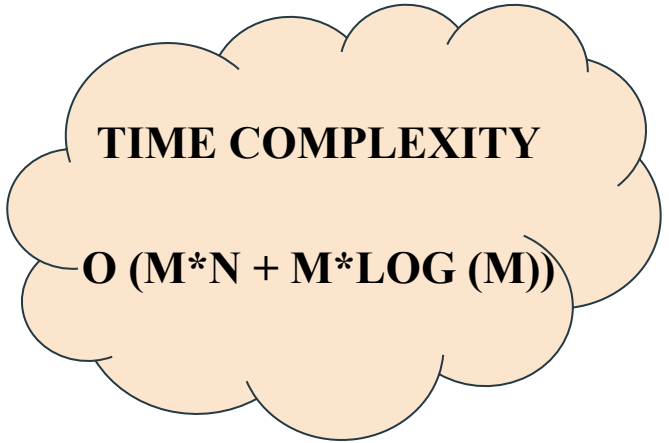
- A multi-dimensional array is created to store the documents and the information within them.
- We count the total occurrences of the query term 't' in each document to get the TF.
- We count the number of documents in which 't' occurs to get the DF.
- We calculate the tf-idf score for each document.
- We map the documents with the tf-idf scores and sort the tf-idf scores to obtain the documents in increasing order of relevance.

Time Complexity Analysis for the Algorithm

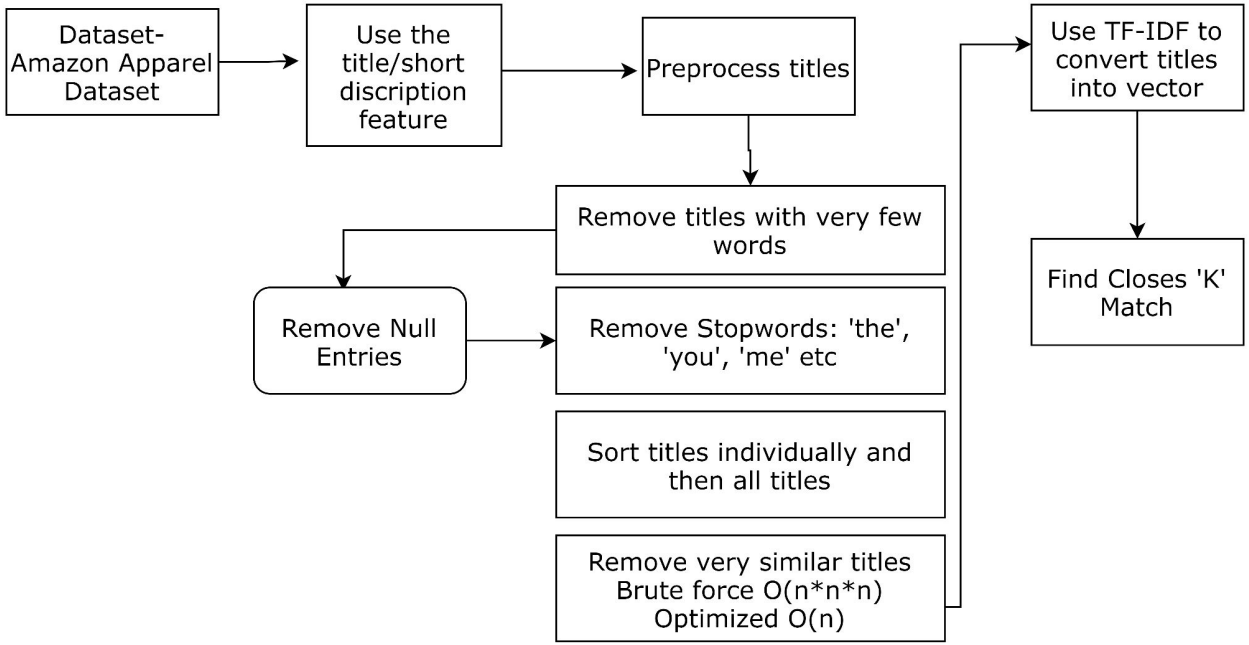
Algo tfidf

1. Let the total number of documents = M
2. Let the number of words in one document = N
3. Query term → 't'
4. doc[1, 2, 3, … , M][1, 2, 3, … , N]

5. count1 = 0 ; _____O (1)
6. count2 = 0 ; _____O (1)
7. for i = 1 to M
8. for j = 1 to N
9. if (doc[i][j] == 't')
10. {
11. count1++; _____O (MN)
12. }
13. if (count1 > 0)
14. count2++;
15. tf [i] = (Count1/N); _____O (M)
16. count1 = 0; _____O (M)
17. df = count2; _____O (1)
18. idf = log ($\frac{M}{df+1}$); _____O (1)
19. for k = 1 to M
20. tfidf [k] = tf [k] * idf ; _____O (M)
21. map (tfidf [1, 2, 3, … , M] → doc [1, 2, 3, … , M]) ; _____O (M)
22. sort (tfidf [1, 2, 3, … , M]) ; _____O (M*log(M))
23. // Now, we have obtained the documents in the order of increasing relevance
24. // The final document doc [M] is the most relevant one
25. Return (doc [M]) _____O (1)



Using Tf-Idf Technique to create an Apparel Recommendation System



CODE AND OUTPUT FOR TEXT BASED APPAREL RECOMMENDATION

IMPORTING THE NECESSARY LIBRARIES

Imports

```
[ ]  import json
     import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     import nltk
     from nltk.corpus import stopwords
     from tensorflow.keras.preprocessing.text import Tokenizer
     from tensorflow.keras.preprocessing.sequence import pad_sequences
     nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
True
```

```
[ ]  with open('/content/drive/My Drive/Dataset/tops_fashion.json') as f:
       data = json.load(f)
```

Exploring the data

EXPLORING OF DATA

Exploring the data

```
df = pd.DataFrame(data)
df = df.drop(['sku', 'author', 'publisher','availability', 'reviews', 'large_image_url','availability_type','s
df.head()
del data
```

```
[ ]  for i in range(10):
       print(df['title'][i]);
       plt.figure
```

```
Minions Como Superheroes Ironman Long Sleeve Round Neck T-Shirt For Women
FIG Clothing Womens Izo Tunic
FIG Clothing Womens Won Top
Focal18 Sailor Collar Bubble Sleeve Blouse Shirt Women Mori Girl Casual Top Harajuku
Featherlite Ladies' Long Sleeve Stain Resistant Tapered Twill Shirt, 2XL, Onyx Black/ Stone
[Fits Cloth] Grape Solid Modern Long Sleeve Plain T Shirt
Women's Unique 100% Cotton T - Special Olympics World Games 2015 White Size L
Floerns Women's Bell Sleeve Beading Casual Blouse Top
Standing on His Promises Rhinestones T-Shirt Ripped Cut Out Red Long
Fila Women's Tulip Durable Tennis Style Comfort Tank
```

RECOMMENDATION ON THE BASIS OF TITLES

Recommendation on basis of titles of clothes

1. Remove all products with very few words in there title.
2. Sort the whole data based on title (alphabetical order of title),m and then remove titles that are very similar
3. using TFIDF(Term Frequency inverse document frequency) on Product titles to get an array representation as it gives less weightage to the words that appear often in the documents and focuses on words that are descriptive of the image

```
[ ]  print('Total elements in the list: ', len(df))
     print('Attributes of an apparel: ', list(df.columns))
     print('Total null elements in formatter_price', sum(df['formatted_price'].isnull().values))
```

```
Total elements in the list:  183138
Attributes of an apparel:  ['asin', 'product_type_name', 'formatted_price', 'color', 'brand', 'title', 'medium_
Total null elements in formatter_price 154743
```

```
[ ]  df = df.sort_values('title')
     df.reset_index(drop = True, inplace = True)
```

```
[ ]  df.head(2)
```

CLEANING OF TEXTS (PRE-PROCESSING STEP)

Cleaning the texts

Remove duplicates in data

```
[ ]  df = df.drop_duplicates(subset = 'medium_image_url').reset_index().drop(['index'],axis=1)
     df = df.drop_duplicates(subset = 'title').reset_index().drop(['index'],axis=1)
     print(len(df))
```

```
165698
```

Remove null entries

```
[ ]  df =df.loc[~df['color'].isnull()]
     df =df.loc[~df['title'].isnull()]
     df =df.loc[~df['formatted_price'].isnull()]
     len(df)
```

```
18785
```

REMOVING DUPLICATES IN DATA AND NULL ENTRIES (PRE-PROCESSING STEP)

Remove duplicates in data

```
[ ]  df = df.drop_duplicates(subset = 'medium_image_url').reset_index().drop(['index'],axis=1)
     df = df.drop_duplicates(subset = 'title').reset_index().drop(['index'],axis=1)
     print(len(df))
```

```
165698
```

Remove null entries

```
[ ]  df =df.loc[~df['color'].isnull()]
     df =df.loc[~df['title'].isnull()]
     df =df.loc[~df['formatted_price'].isnull()]
     len(df)
```

```
18785
```

Remove stop words

REMOVING STOP WORDS (PRE-PROCESSING STEP)

Remove stop words

```
[ ]  stopwrds = list(stopwords.words('english'))
     l = list(df['title'])
     for i in range(len(df['title'])):
       for j in range(len(stopwrds)):
         l[i] = l[i].replace(' '+stopwrds[j]+' ',' ')
     df['title']=l
```

```
[ ]  tokenizer = Tokenizer(oov_token="<UNK>") # if num words not provided it consider all
     tokenizer.fit_on_texts(list(df['title']))      #this is a must, to give the tokenizer an idea of the train dat
     word_index = tokenizer.word_index
     sequences = tokenizer.texts_to_sequences(list(df['title']))
     sequences[0]
     tok_sent = tokenizer.sequences_to_texts(sequences)
     df['title']=tok_sent
```

```
[ ]  for col in df.columns:
       print('Column: {}, Total values {}, unique values {}, nan values {}'.format(col, len(df[col]), len(set(list(
```

```
Column: asin, Total values 18785, unique values 18785, nan values 0
Column: product_type_name, Total values 18785, unique values 54, nan values 0
```

CREATING LISTS


```
[ ]  for col in df.columns:
       print('Column: {}, Total values {}, unique values {}, nan values {}'.format(col, len(df[col]), len(set(list(df[col])))
```

```
Column: asin, Total values 18785, unique values 18785, nan values 0
Column: product_type_name, Total values 18785, unique values 54, nan values 0
Column: formatted_price, Total values 18785, unique values 2928, nan values 0
Column: color, Total values 18785, unique values 4243, nan values 0
Column: brand, Total values 18785, unique values 3534, nan values 55
Column: title, Total values 18785, unique values 18781, nan values 0
Column: medium_image_url, Total values 18785, unique values 18785, nan values 0
Column: editorial_reivew, Total values 18785, unique values 12691, nan values 282
```

```
df.reset_index(drop=True,inplace=True)
indices = list(df.index)
df.head(1)
```

	asin	product_type_name	formatted_price	color	brand	title	medium_image_url	editoria
0	B008D30AGK	SHIRT	$7.51	Multicolored	Out+of+Print+Clothing	1984 retro book cover women's slim fit t shirt...	https://images-na.ssl-images-amazon.com/images...	

REMOVING VERY SIMILAR TITLES (PRE-PROCESSING STEP)

- Removing titles that have are very similar

```
import itertools
deduped_idx = []
i = 0
j = 0
while i < len(df) and j < len(df):
    previous_i = i
    a = df['title'].loc[i].split()
    j = i+1
    while j < len(df):
        b = df['title'].loc[indices[j]].split()
        length = max(len(a), len(b))
        count  = 0
        for k in itertools.zip_longest(a,b):
            if (k[0] == k[1]):
                count += 1
        if (length - count) > 2:
            deduped_idx.append(i)
            i = j
            break
        else:
            j += 1
    if previous_i == i:
        break
```

TRAVERSAL THROUGH CORPUS


```
[ ]  df = df.iloc[deduped_idx]
     df.reset_index(drop=True,inplace=True)
     df.head(2)
     list_titles = list(df['title'])
     print(len(df))
```

```
16231
```

```
[ ]  import sklearn
     from sklearn.feature_extraction.text import CountVectorizer
     from sklearn.feature_extraction.text import TfidfVectorizer
```

```
[ ]  vectorizer = CountVectorizer()
     vectorizer
     X = vectorizer.fit_transform(list_titles) #corpus is a list tof sentences
     analyze = vectorizer.build_analyzer()
```

```
[ ]  vectorizer = TfidfVectorizer()
     X = vectorizer.fit_transform(list_titles)
```

```
[ ]  def closestkrecom(idx,k):
       dist = sklearn.metrics.pairwise_distances( X, X[idx], metric='cosine')
       dist = np.squeeze(dist)
```

FINDING CLOSEST MATCH AND DISPLAYING OUTPUT


```
[ ]  def closestkrecom(idx,k):
       dist = sklearn.metrics.pairwise_distances( X, X[idx], metric='cosine')
       dist = np.squeeze(dist)
       #print(dist.shape)
       sort_idxs = np.random.choice(np.argsort(dist,axis=0)[1:],k)
       return sort_idxs
```

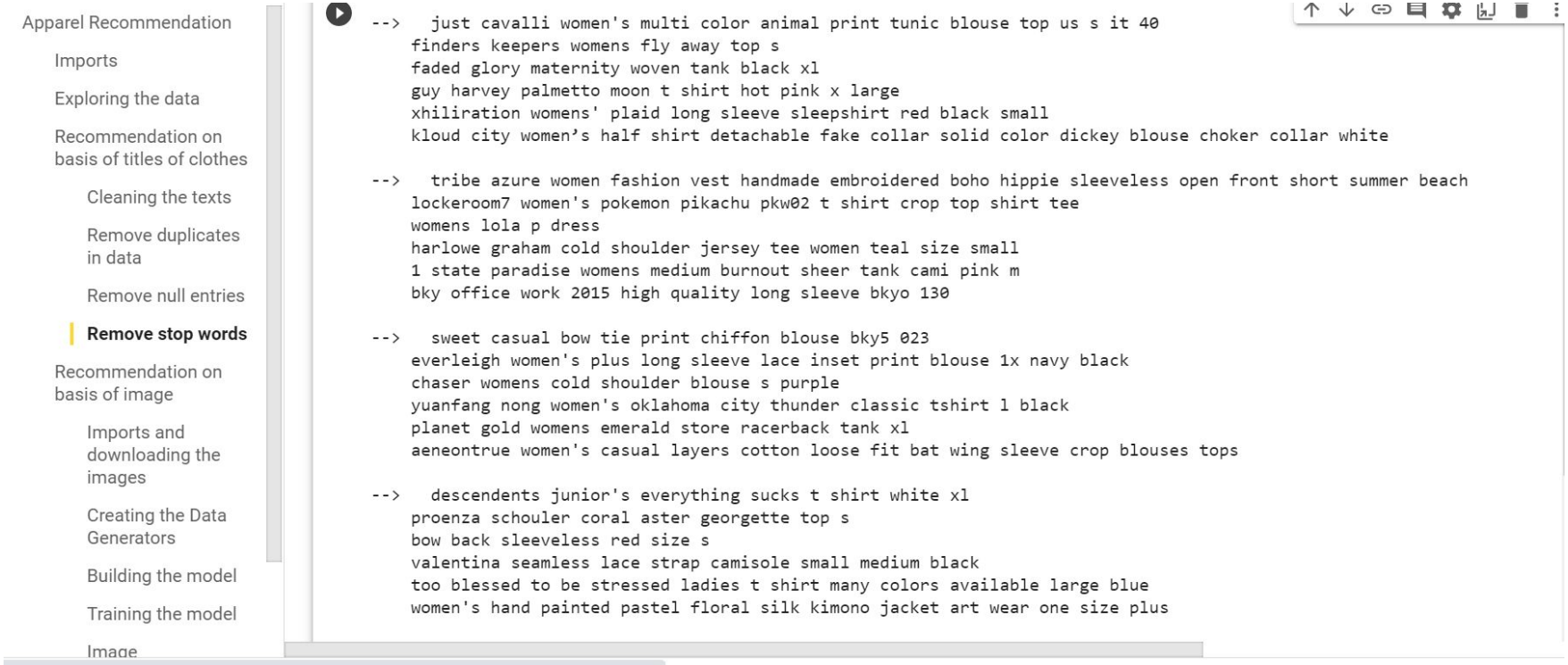
```
[ ]  import random
     import tensorflow as tf
```

```
[ ]  for idx in random.sample(list(np.arange(len(df))), 20):
       print('-->  ', df['title'][idx])
       for j in closestkrecom(idx,5):
         print('   ',df['title'][j])
       print()
```

```
-->   bridal party pastel floral robe pink glitter style danirb bm fugl 1x 2x
    sexy summer sport halter tops digital printed women casual t shirt seawater small fresh
    bar iii sleeveless mock neck swing top black combo xxl
    bobeau women's small petite striped tank cami top brown ps
    kingyuan adjustable nipple clamps women men bondage sets couples flirt toy
    ms read women's contrast textured top 22 pale blue navy

-->   halston heritage sash halter top bone size medium
    official tummeow hang 2 art l large white t shirt women
```

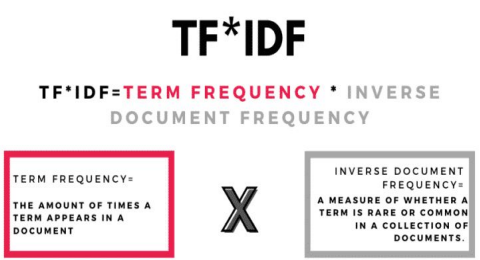
OUTPUT

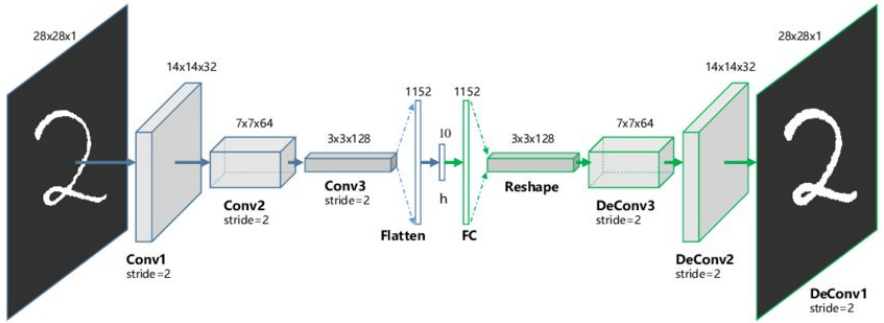

Apparel Recommendation
Imports
Exploring the data
Recommendation on basis of titles of clothes
Cleaning the texts
Remove duplicates in data
Remove null entries
Remove stop words
Recommendation on basis of image
Imports and downloading the images
Creating the Data Generators
Building the model
Training the model
Image
--> just cavalli women's multi color animal print tunic blouse top us s it 40
finders keepers womens fly away top s
faded glory maternity woven tank black xl
guy harvey palmetto moon t shirt hot pink x large
xhiliration womens' plaid long sleeve sleepshirt red black small
kloud city women’s half shirt detachable fake collar solid color dickey blouse choker collar white
--> tribe azure women fashion vest handmade embroidered boho hippie sleeveless open front short summer beach
lockeroom7 women's pokemon pikachu pkw02 t shirt crop top shirt tee
womens lola p dress
harlowe graham cold shoulder jersey tee women teal size small
1 state paradise womens medium burnout sheer tank cami pink m
bky office work 2015 high quality long sleeve bkyo 130
--> sweet casual bow tie print chiffon blouse bky5 023
everleigh women's plus long sleeve lace inset print blouse 1x navy black
chaser womens cold shoulder blouse s purple
yuanfang nong women's oklahoma city thunder classic tshirt l black
planet gold womens emerald store racerback tank xl
aeneontrue women's casual layers cotton loose fit bat wing sleeve crop blouses tops
--> descendents junior's everything sucks t shirt white xl
proenza schouler coral aster georgette top s
bow back sleeveless red size s
valentina seamless lace strap camisole small medium black
too blessed to be stressed ladies t shirt many colors available large blue
women's hand painted pastel floral silk kimono jacket art wear one size plus

Creating an Advanced Recommendation System

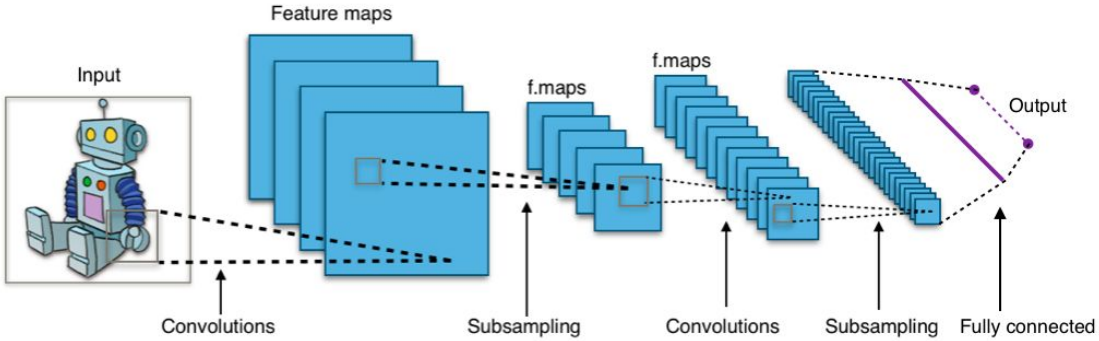
Text Based Recommendation System:

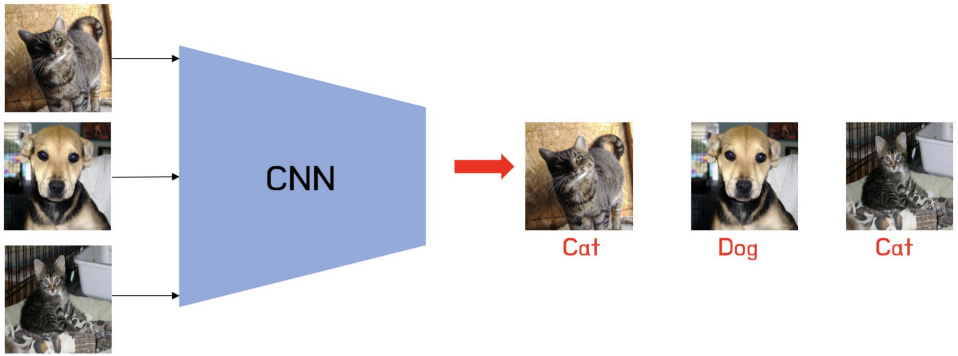
Image based Recommendation System

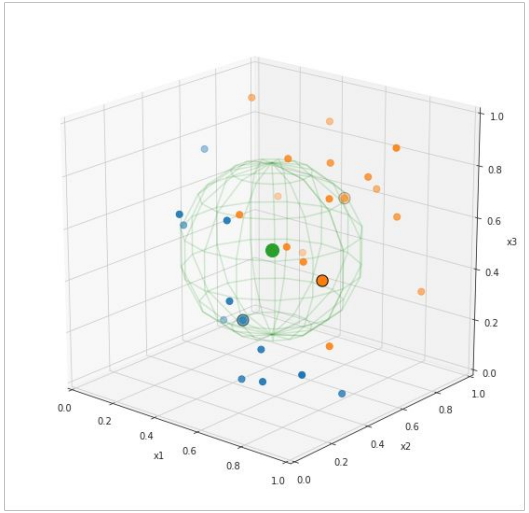




CNN- Convolutional Neural Network

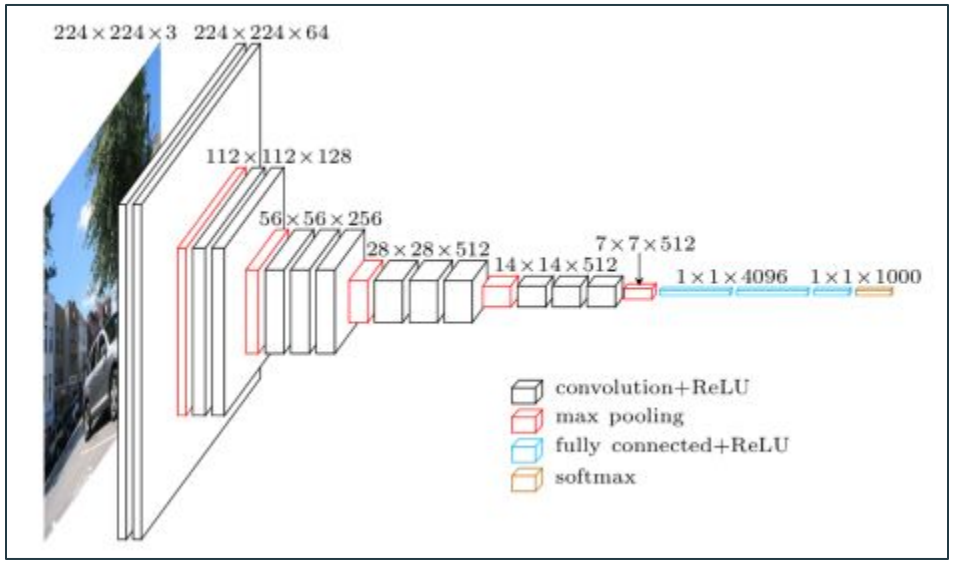






Plot vector in space
Find nearest match accordingly

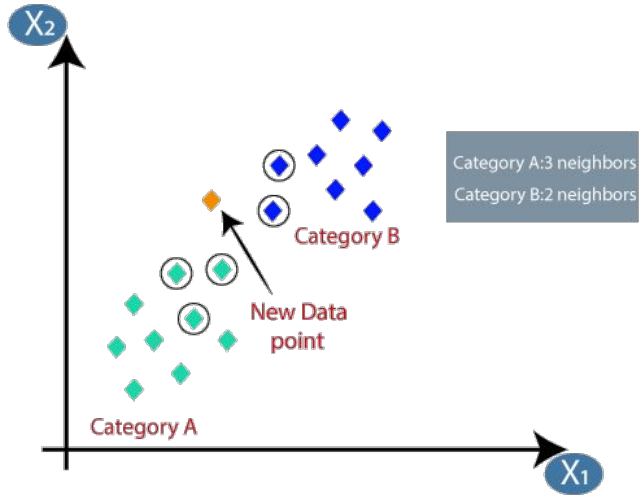
VGG-16- A Convolutional Neural Network Model

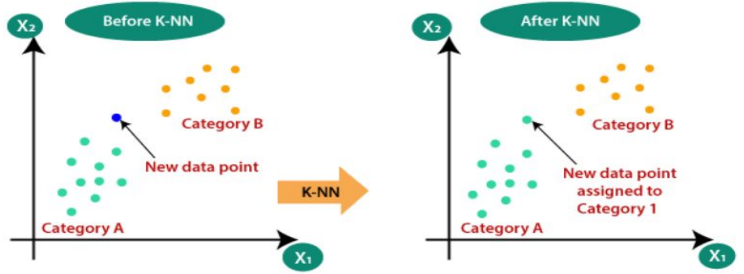


he **VGG16** is a Convolutional Neural Network which is a widely used, practical method of visual object recognition software.

KNN- K nearest Neighbors

- For each image, find its encoding and the closest k ones.
- Precompute other image encoding using VGG-16
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm





CODE AND OUTPUT FOR IMAGE BASED APPAREL RECOMMENDATION

FEATURE EXTRACTION

Apparel Recommendations using Convolutional Neural Network

Get the feature vectors of all apparel images

```
Running this cell will take time, you can skip running this cell. you can download the feature vectors from give
16k_data_cnn_features.npy: https://drive.google.com/open?id=0BwNkduBnePt2c1BkNzRDQ1dOVFk
bottleneck_features_cnn.npy : https://drive.google.com/open?id=0BwNkduBnePt2ODRxWHhUVzIyWDA
```

```
[ ]  import numpy as np
     from keras.preprocessing.image import ImageDataGenerator
     from keras.models import Sequential
     from keras.layers import Dropout, Flatten, Dense
     from keras import applications
     from sklearn.metrics import pairwise_distances
     import matplotlib.pyplot as plt
     import requests
```

USING CNN TO SQUEEZE THE IMAGES INTO A 16 X 16 MATRIX

Apparel Recommendations using Convolutional Neural Network

Get the feature vectors of all apparel images

load the extracted features

get the most similar apparels using euclidean distance measure

Section

```
[ ] import pandas as pd
    import pickle
```

```
Using Theano backend.
Using cuDNN version 5110 on context None
Mapped name None to device cuda: GeForce GTX 1050 (0000:01:00.0)
```

```
[ ] # https://gist.github.com/fchollet/f35fbc80e066a49d65f1688a7e99f069
    # https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html


    # dimensions of our images.
    img_width, img_height = 224, 224

    top_model_weights_path = 'bottleneck_fc_model.h5'
    train_data_dir = 'images2/'
    nb_train_samples = 16042
    epochs = 50
    batch_size = 1


    def save_bottlebeck_features():
        asins = []
        datagen = ImageDataGenerator(rescale=1. / 255)

        # build the VGG16 network
```

SAVING BOTTLENECK FEATURES INTO THE MATRICES

Apparel Recommendations using Convolutional Neural Network

Get the feature vectors of all apparel images

load the extracted features

get the most similar apparels using euclidean distance measure

Section

```
def save_bottlebeck_features():
    asins = []
    datagen = ImageDataGenerator(rescale=1. / 255)

    # build the VGG16 network
    model = applications.VGG16(include_top=False, weights='imagenet')
    generator = datagen.flow_from_directory(
        train_data_dir,
        target_size=(img_width, img_height),
        batch_size=batch_size,
        class_mode=None,
        shuffle=False)

    for i in generator.filenames:
        asins.append(i[2:-5])

    bottleneck_features_train = model.predict_generator(generator, nb_train_samples // batch_size)
    bottleneck_features_train = bottleneck_features_train.reshape((16042,25088))

    np.save(open('workshop/models/16k_data_cnn_features.npy', 'wb'), bottleneck_features_train)
    np.save(open('workshop/models/16k_data_cnn_feature_asins.npy', 'wb'), np.array(asins))


save_bottlebeck_features()
```

```
Found 16042 images belonging to 1 classes.
```

CALCULATING EUCLIDEAN DISTANCE


```
Found 16042 images belonging to 1 classes.
```

load the extracted features

```
[ ] bottleneck_features_train = np.load('workshop/models/16k_data_cnn_features.npy')
    asins = np.load('workshop/models/16k_data_cnn_feature_asins.npy')
```

get the most similar apparels using euclidean distance measure

```
[ ] data = pd.read_pickle('workshop/pickels/16k_apperal_data_preprocessed')
    df_asins = list(data['asin'])
    asins = list(asins)
```

```
[ ] from IPython.display import display, Image, SVG, Math, YouTubeVideo

    def get_similar_products_cnn(doc_id, num_results):
        doc_id = asins.index(df_asins[doc_id])
        pairwise_dist = pairwise_distances(bottleneck_features_train, bottleneck_features_train[doc_id].reshape(1,
```

SORTING CLOTHES IN DECREASING ORDER OF EUCLIDEAN DISTANCE TO GET CLOSEST RECOMMENDATIONS

Apparel Recommendations using Convolutional Neural Network

Get the feature vectors of all apparel images

load the extracted features

get the most similar apparels using euclidean distance measure

Section

```
from IPython.display import display, Image, SVG, Math, YouTubeVideo

def get_similar_products_cnn(doc_id, num_results):
    doc_id = asins.index(df_asins[doc_id])
    pairwise_dist = pairwise_distances(bottleneck_features_train, bottleneck_features_train[doc_id].reshape(1

    indices = np.argsort(pairwise_dist.flatten())[0:num_results]
    pdists  = np.sort(pairwise_dist.flatten())[0:num_results]

    for i in range(len(indices)):
        rows = data[['medium_image_url','title']].loc[data['asin']==asins[indices[i]]]
        for indx, row in rows.iterrows():
            display(Image(url=row['medium_image_url'], embed=True))
            print('Product Title: ', row['title'])
            print('Euclidean Distance from input image:', pdists[i])
            print('Amazon Url: www.amzon.com/dp/'+ asins[indices[i]])

get_similar_products_cnn(12566, 10)
```

OUTPUT

Apparel Recommendations using Convolutional Neural Network

Get the feature vectors of all apparel images

load the extracted features

get the most similar apparels using euclidean distance measure

Section

```
[ ]
```

```
Product Title:  burnt umber tiger tshirt zebra stripes xl  xxl
Euclidean Distance from input image: 0.0625
Amazon Url: www.amzon.com/dp/B00JXQB5FQ
```

```
Product Title:  pink tiger tshirt zebra stripes xl  xxl
Euclidean Distance from input image: 30.0501
Amazon Url: www.amzon.com/dp/B00JXQASS6
```

```
Product Title:  yellow tiger tshirt tiger stripes  l
Euclidean Distance from input image: 41.2611
Amazon Url: www.amzon.com/dp/B00JXQCUIC
```