Plotting text and image vectors using t-SNE

How to plot multidimensional vectors created using Infersent and Resnet in two-dimensional space.

Ashutosh Singh Follow

Mar 11, 2019 · 5 min read ★

Power to your eye - Happy Visualizing

Doing wonders with Visualization

John Micternan rightly pointed out that ‘In presentation lies the true entertainment’, and if you can present your data in a way the other guy wished, he would be in a state of bliss. Imagine a world where communication of all data could be visualized, both my

Given the human brain's comfort when it comes to pictorial content and considering the fact the 90% of the information transmitted through the human brain is visual,we have tried to plot sentence and image vectors in a 2-D space ,where each spatial spot represents a sentence and similar sentence vectors would be placed in spatial proximity.

The Problem Statement:

To plot sentence and image vectors where sentence vectors are arrived at using Infersent or Google's Universal Sentence Encoder and images vectors are features extracted in the Resnet 50 model.

Our sentences are nothing but a combination of a Product's Name, description and its specifications whereas an Image is basically the product image. To get a hold of what a product means in the context of Indiamart, please read here.

Understanding Text and Image Vectors

Text vectors (either word vectors or sentence vectors)are created by converting textual data into the numerical form using embedding techniques like word2vec, fasttext, Infersent, Google universal sentence encoder etc. The dimension of your vector may depend upon your technique of preparing it. For example, Infesent creates a vector of 4096 dimensions for each sentence, whereas GUSE(Google Universal Sentence Encoder) creates a vector of 512 dimensions for each sentence.

Image vectors are the feature vector of an image created using some image feature extraction techniques. In our case, we have Resnet image vectors having 512 dimensions for each and every image.

Note: In this post, we are not dwelling into details of a sentence or word embedding (they will be covered separately in another blog), rather the focus is how we depicted them on a scatter plot.

Going Multidimensional and Coming Back

The difficulty with visualizing the distributed representation of text and image feature vectors is that they are multidimensional(for e.g some vectors are 512 dimensional

whereas some are of 4096 dimensions) which are quite higher than the usual 2 or 3-dimensional vector plots that human can see and interpret.

Since there are no direct ways to project these data points on the plot that is easy for humans to understand, we need some powerful method to map these high dimensional data into 2 or 3-dimensional space. Here t-SNE played the trick for us.

The below images shows how each vector was converted to a 2-D Array for ease of plotting.

	TSNE_x	TSNE_y	Label	Cluster
0	-20.043272	-10.387577	Earthmoving Bucket	3
1	0.843924	-61.846951	Excavator	16
2	4.514275	28.035549	Wheel Loaders	17
3	9.372581	-52.792614	Excavator	0
4	8.270648	3.150992	Excavator	3
5	-19.491745	38.546375	Wheel Loaders	13
6	17.083792	-17.043217	Excavator	16
7	-30.921326	-32.213989	Excavator	16
8	24.351297	24.055000	Motor Grader	17
9	14.048458	48.008224	Road Roller	4

Multidimensional Vectors converted to a two-dimensional vector using t-SNE.

What is t-SNE?

t-SNE stands for t-distributed stochastic neighbor embedding. It is a technique for dimensionality reduction that is best suited for the visualization of high dimensional data-set. t-SNE is a randomized algorithm, i;e every time we run the algorithm it returns slightly different results on the same data-set. To control this we set a random state with some arbitrary value. Random state is used here to seed the cost function of the algorithm.

Giving Colors to thoughts -The Scatter Plot

The below steps will guide you to depict your vectors on a scatter plot.

Step1: Import necessary libraries:

```
import numpy as np
from sklearn.manifold import TSNE
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
import os
from numpy import linalg
from numpy.linalg import norm
from scipy.spatial.distance import squareform, pdist

# Importing sklearn and TSNE.
import sklearn
from sklearn.manifold import TSNE
from sklearn.datasets import load_digits
from sklearn.preprocessing import scale

# We'll hack a bit with the t-SNE code in sklearn.
from sklearn.metrics.pairwise import pairwise_distances
from sklearn.manifold.t_sne import (_joint_probabilities,
                                    _kl_divergence)
#from sklearn.utils.extmath import _ravel
# Random state we define this random state to use this value in TSNE which is a randmized algo.
RS = 25111993

# Importing matplotlib for graphics.
import matplotlib.pyplot as plt
import matplotlib.patheffects as PathEffects
import matplotlib
%matplotlib inline

# Importing seaborn to make nice plots.
import seaborn as sns
sns.set_style('darkgrid')
sns.set_palette('muted')
sns.set_context("notebook", font_scale=1.5,
                rc={"lines.linewidth": 2.5})
```

Step 2: Loading vectors created using different embedding techniques: In this step, we have loaded the large dimensional vector created using GUSE, Infersent or Resnet (in case of image vectors) and fitted into 2 d space using t-SNE algorithm.

```
# Loading the vector
Data_1 = np.genfromtxt ('c:/Users/Imart/Desktop/text_vectors/excavator_text_vectors.csv', delimi
# Here we are importing KMeans for clustering Product Vectors
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=18, random_state=0).fit(Data_1)
# We can extract labels from k-cluster solution and store is to a list or a vector as per our re
Y=kmeans.labels_ # a vector

z = pd.DataFrame(Y.tolist()) # a list
# Fit the model using t-SNE randomized algorithm
digits_proj = TSNE(random_state=RS).fit_transform(Data_1)
```

load vector n-dim vector and fit the model using t-SNE hosted with ♡ by GitHub view raw

Step 3: Create a User defined function to plot the above vector in 2-d space: We have used a combination of t-SNE, matplotlib and seaborn to create the scatter plot.

```
# An user defined function to create scatter plot of vectors
def scatter(x, colors):
    # We choose a color palette with seaborn.
    palette = np.array(sns.color_palette("hls", 18))

    # We create a scatter plot.
    f = plt.figure(figsize=(32, 32))
    ax = plt.subplot(aspect='equal')
    sc = ax.scatter(x[:,0], x[:,1], lw=0, s=120,
                    c=palette[colors.astype(np.int)])
    #plt.xlim(-25, 25)
    #plt.ylim(-25, 25)
    ax.axis('off')
    ax.axis('tight')

    # We add the labels for each cluster.
    txts = []
    for i in range(18):
        # Position of each label.
```

```
        txt.set_path_effects([
            PathEffects.Stroke(linewidth=5, foreground="w"),
            PathEffects.Normal()])
        txts.append(txt)

    return f, ax, sc, txts
```

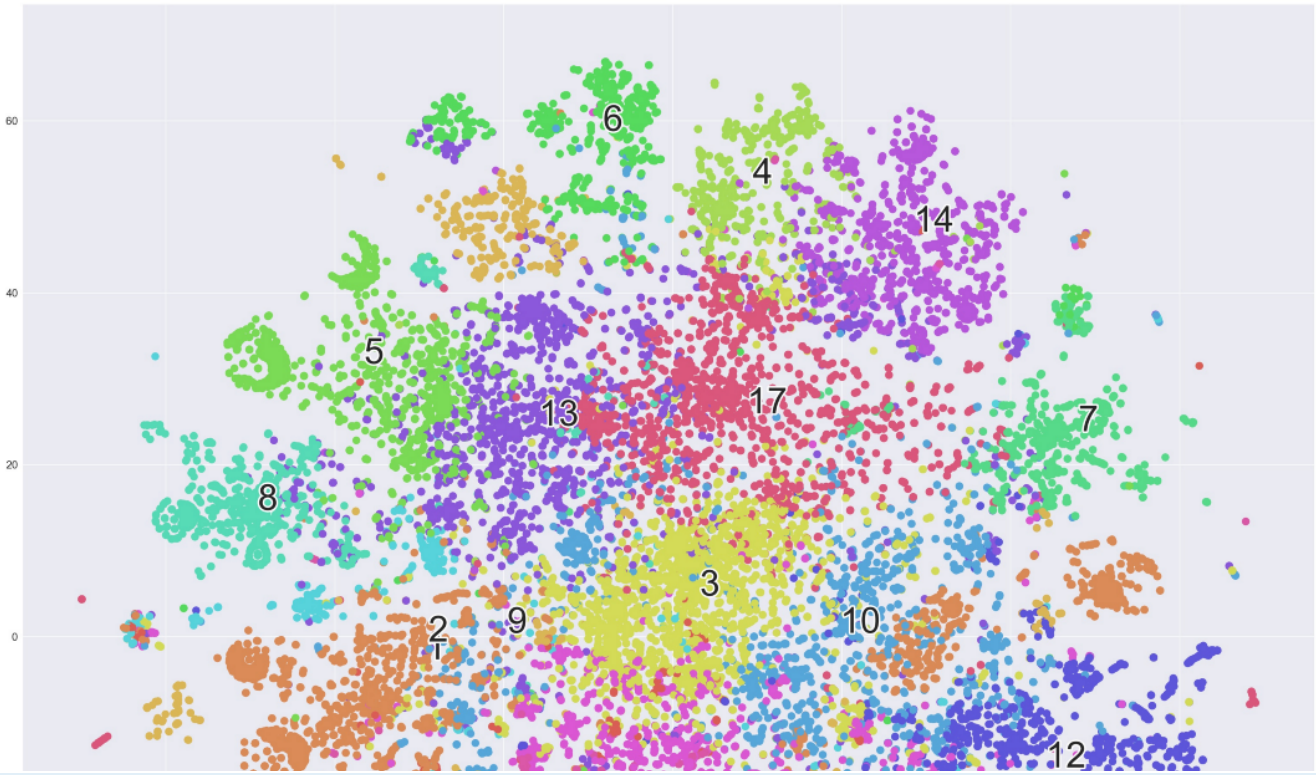
Step 4: Visualizing and saving the plot

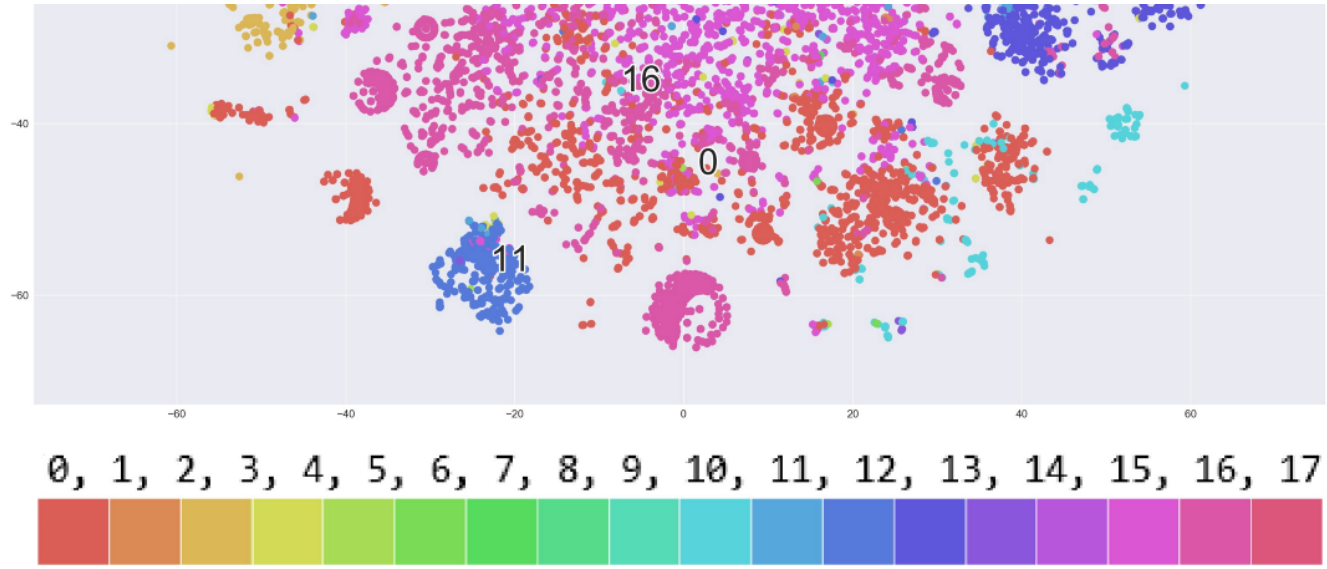
```
print(list(range(0,18)))
sns.palplot(np.array(sns.color_palette("hls", 18)))
scatter(digits_proj, Y)
plt.savefig('digits_tsne-generated_18_cluster.png', dpi=120)
```

visualize and save the plot hosted with ♡ by GitHub view raw

Visualization based on K-Means cluster of the above sentence vectors

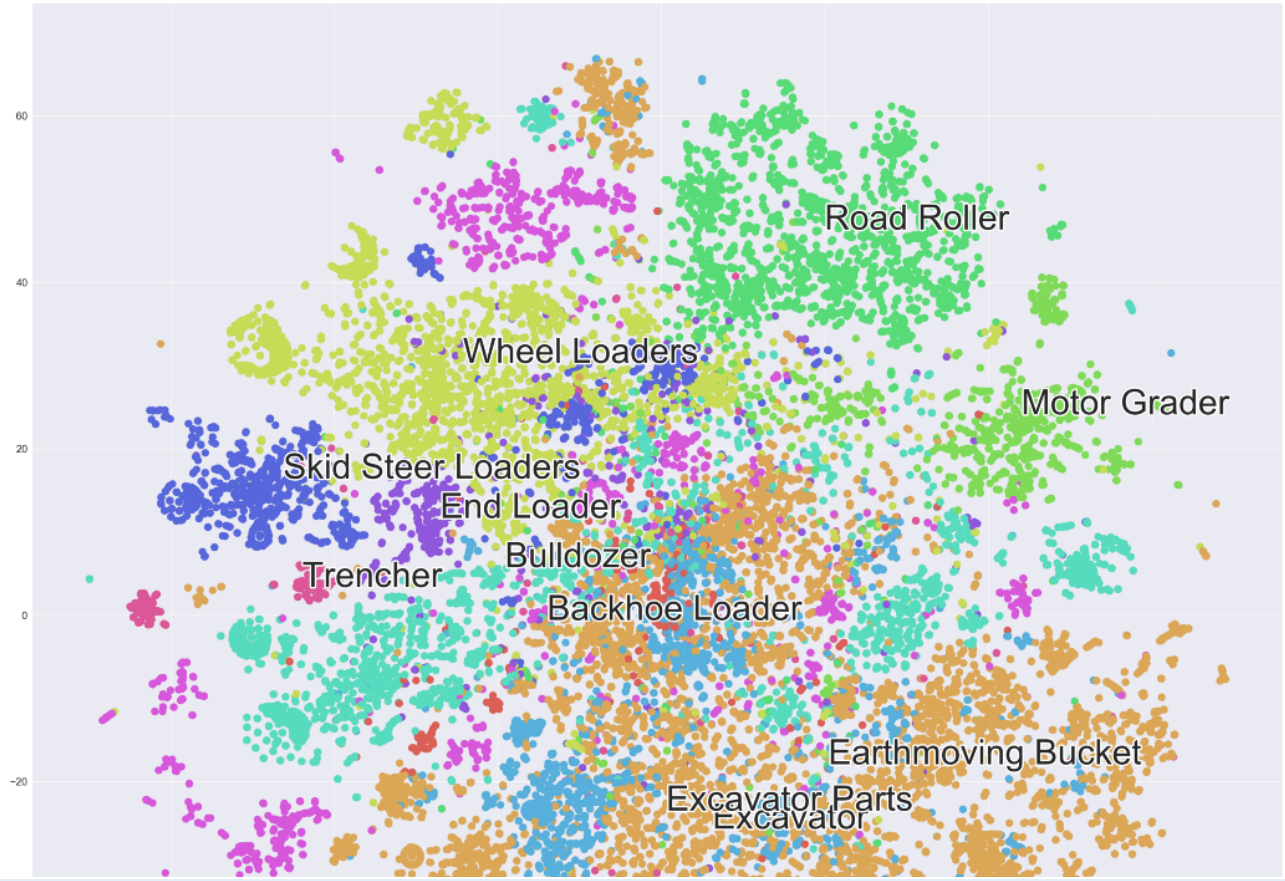
We used K-Means clustering to group similar sentence vectors. 18 such clusters were formed. The below scatter diagram shows the distribution of products within each cluster.

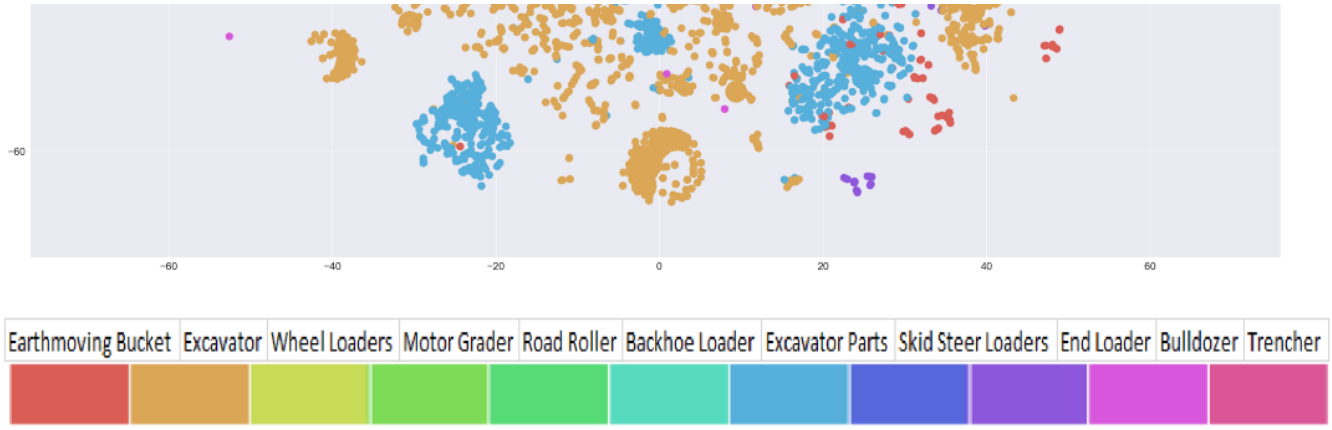




Visualization based on the existing cluster of the sentence vectors.

Here the sentence vectors were plotted on the basis of it's existing cluster. Each product (sentence vector) belongs to some cluster or category and each category has a unique color.





We can see how the products(sentences) from Earthmoving Bucket are spread across the plot suggesting their nearness to other related clusters.

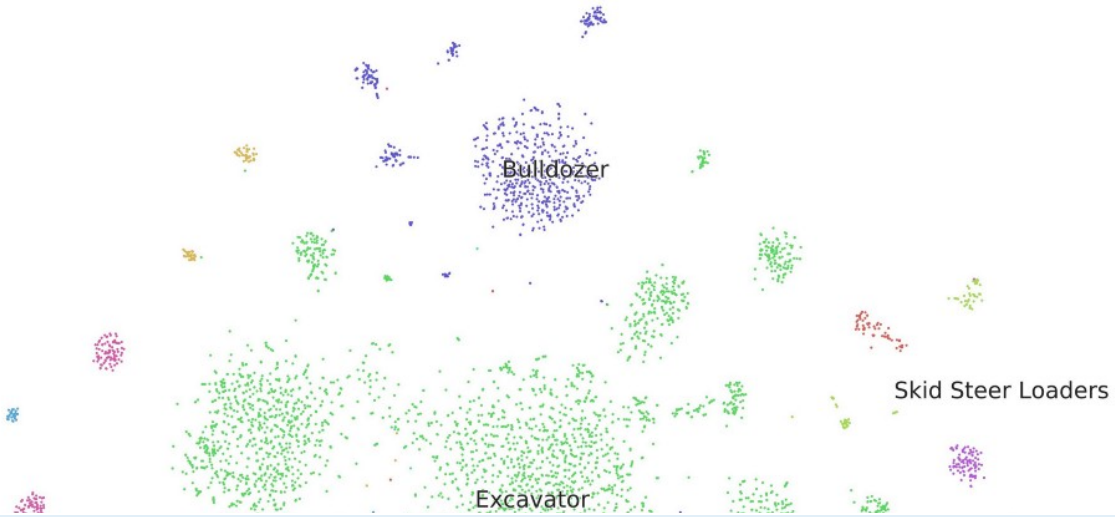
Adding an axis to your 2-d space: We can do the same by simply setting x-limit(going from -60 to + 60 in our case) and y-limit(going from -60 to +60) on the axis

De-coding the plot

Each dot in the scatter plot represents the text inside a single product. Products with semantically similar text are placed close to each other. The color of each dot represents the cluster to which it belongs i.e all sentences in cluster 0 are represented in red, in cluster 1 are represented in orange and so on.

Visualization of Image vectors

Each spot in the plot represents the features of a single image. Similar images fall in a closely knit space. Each image is converted to feature vectors using Resnet.



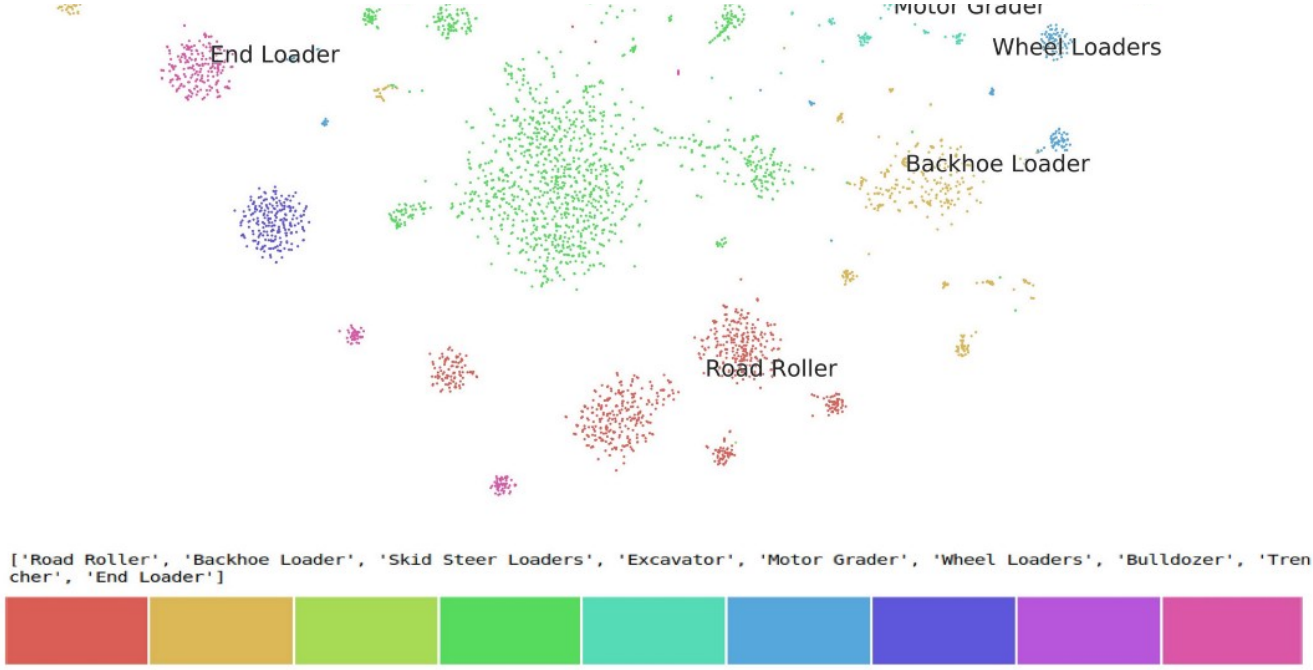


Image Vectors plotted in a 2-D Space. On close observation, it is seen that similar images are occupying close spaces while different image vectors are scattered in the plot.

Putting things to use

The scatter plot has multiple applications for us-it told us how similar our existing images or sentences were and whether the current categorization is appropriate with respect to the products mapped in it. It also helped us categorize a new set of products or images.

Check out the full python code on my Github repository.

Machine Learning | Data Visualization | Data Science | Deep Learning | Artificial Intelligence