NLP Assignment 2

Negative Sampling

Negative Sampling is used to boost the speed by reducing the computational time taken to compute the word embeddings. The way we achieve this is by considering negative samples. Conventionally we require to compute $V * d$ where d is the embedding size and V is the vocabulary size. When we consider k negative samples we only have to compute $k * d$. Generally V is in the order of 10^4 and k is in the order of 10. Thus achieving a 1000 fold reduction in computation.

Negative samples are nothing but wrong target words that do not fit the context. The idea is that in one run of the model, we update only the a subsection of the weight, ones corresponding to the negative samples. We update the positive vector for every training sample ensuring that the target vector is learned.

The negative points are randomly sampled by a probability distribution:

$$P(w_i) = \frac{f(w_i)^{\frac{3}{4}}}{\Sigma_j f(w_j)^{\frac{3}{4}}}$$

It has been observed that taking the Unigram Probability to the power $\frac{3}{4}$ works the best. Basically it significantly increases the occurrence of very less probable words and maintains relatively unchanged probabilities of those words that occur very frequently.

Analysis

Co-occurrence + SVD

I have considered the following words from the data:

1. Sad
2. Eat
3. Farm
4. Valentine
5. Camera

The closest words and their distances are as follows:

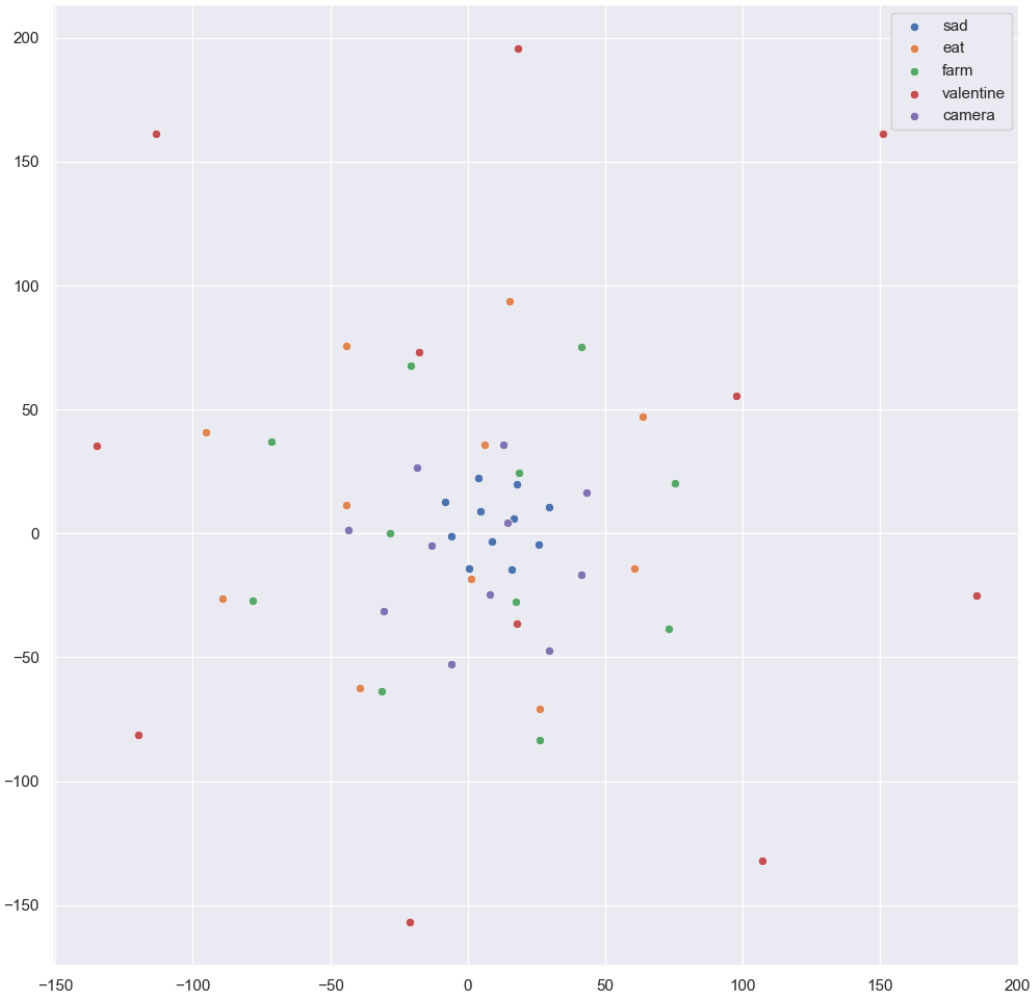
```
sad : 1
screwhole : 0.7076258659362793
usedduct : 0.7065192461013794
forced : 0.6989447474479675
rumored : 0.6825766563415527
difficult : 0.6799338459968567
comfortale : 0.6779948472976685
motivated : 0.6759713292121887
suppose : 0.6757966876029968
unacceptable : 0.674086332321167
suppossed : 0.6641539931297302
```

```
eat : 1
doesn : 0.6573582887649536
leapsters : 0.6243537664413452
autoscroll : 0.6213804483413696
mostlikely : 0.6160969734191895
hasn : 0.6149278283119202
fates : 0.6106479167938232
bubbling : 0.6087238788604736
appletalk : 0.608513593673706
rattiling : 0.6005882620811462
dosen : 0.5995253324508667
```

```
valentine : 1
pido : 0.9596784114837646
shures : 0.9595903754234314
tama : 0.9594146609306335
rie : 0.9593063592910767
earssemi : 0.9590460062026978
trae : 0.9588366150856018
debo : 0.9587633013725281
subesxpuestas : 0.958620011806488
sorporesa : 0.9585062265396118
potencia : 0.9583702683448792
```

```
farm : 1
replacment : 0.7014119029045105
rides : 0.6939384937286377
whale : 0.6936210989952087
teeth : 0.6807915568351746
salvaged : 0.6758299469947815
studying : 0.6677162647247314
eterex : 0.6635040640830994
rolodex : 0.656919538974762
soreness : 0.6415956616401672
dmc : 0.638459324836731
```

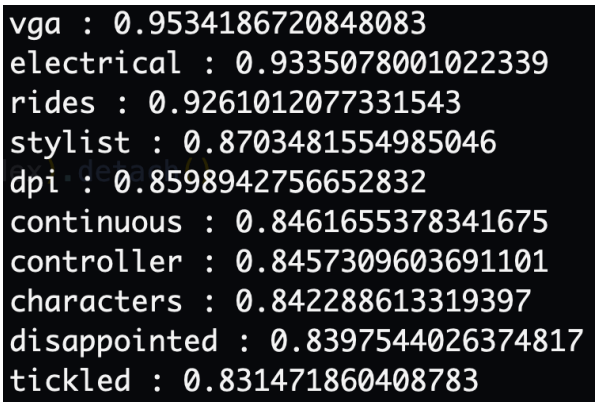
```
camera : 1
lense : 0.8197601437568665
lens : 0.8035140633583069
keyboard : 0.7954187393188477
headset : 0.7901676893234253
mouse : 0.7648659944534302
turntable : 0.7377771139144897
trackball : 0.7110946774482727
rack : 0.7065959572792053
filter : 0.7065799832344055
player : 0.6931032538414001
```

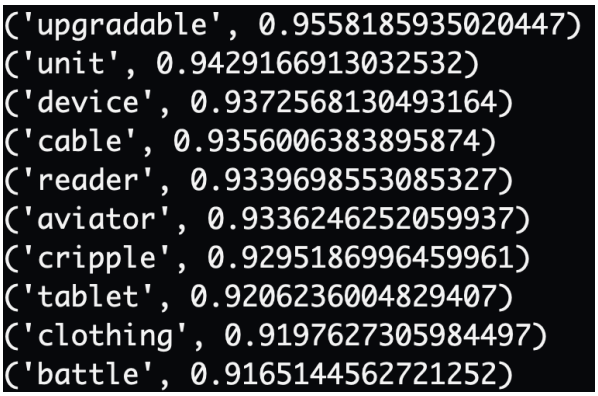


Word2Vec

Could train for only 5000 sentences due to time shortage hence tested in a small model. Will upload the large model.

Trained both on a window size of 5. The output is a word embedding of size 10. Found the top 10 hit for the word “camera”.





Left - My model, Right - Word2Vec (Gensim)