Intro to Natural Language Processing
Assignment – 3

Name: Lakshmipathi Balaji
Roll No: 2021114007

2.1 Theory

1. Explain negative sampling. How do we approximate the word2vec training computation using this technique?

In word2vec training, negative sampling is a method for approximating the computation required to train the model. Word embeddings, which are vector representations of words that capture their meanings and interactions with other words, are what word2vec aims to teach users. The Continuous Bag of Words (CBOW) model trains on a huge corpus of text by either predicting a target word given its context or, in the case of other models, predicting the context given a target word (in the case of the Skip-gram model).

The model is trained using the softmax method, which determines the likelihood that each word in the vocabulary would be the appropriate output given the input context, in normal word2vec. An alternative to softmax that produces decent results is negative sampling. It allows us to approximate the computation required to train the model. Negative sampling only samples a limited number of "negative" words (words that are not in the context). For a given context it takes different a random word from the whole vocabulary. Note that going by emperical frequencies or uniform frequency for all the words doesn't give good results. A optimal way presented in Mikolov paper is to go by chance of emperical frequency raised to the power of 3/4th.

After getting the positive and negative samples the model is trained for the target word with positive sample and not target word with negative sample hence efficiently finding embeddings of words.

Source/ Reference: https://www.youtube.com/watch?v=4PXILCmVK4Q

2. Explain the concept of semantic similarity and how it is measured using word embeddings. Describe at least two techniques for measuring semantic similarity using word embeddings.

The degree to which two words or phrases have similar meanings is referred to as semantic similarity. In many natural language processing applications, including text categorization, information retrieval, and machine translation, measuring semantic similarity is crucial. Word embeddings, which are dense vector representations of words that capture their semantic and syntactic links, may be used to calculate semantic similarity.

Using word embeddings, there are numerous methods for calculating semantic similarity. Cosine similarity, which calculates the cosine of the angle between two vectors, is one widely used method. The dot product of the two vectors divided by the product of their magnitudes is the formula for calculating the cosine similarity between two word embeddings.The final value falls between -1 and 1, with -1 denoting full dissimilarity and 1 denoting complete similarity.

Word mover's distance (WMD), a distance metric that calculates the smallest distance needed to transfer the words from one phrase to another, is a different method for gauging semantic similarity. In this method, the WMD between each pair of word embeddings represents the semantic similarity between two phrases. The total semantic similarity of the two sentences may be captured by WMD, which takes into consideration the semantic links between words and their relative distances.

2.2 Implementation

The code submitted contains the implementation of both SVD word to vec model and the cbow model with negative sampling.

2.3Analysis

Hyperparameter tuning in Cbow -
Here are a few experiments made during tuning the hyperparameters -

SPARSEADAM OPTIMIZER, LR = 0.01, EPOCHS = 50, EMBEDDING_DIM = 300, BATCH_SIZE = 512

```
words_similar = model.similarity("titanic")
print(words_similar[0:10])
```

```
[['1.0000001' 'titanic']
 ['0.46512032' 'afficianado']
 ['0.46341032' 'polanski']
 ['0.46142277' 'nazipropaganda']
 ['0.46134686' 'aviator']
 ['0.46037847' 'indulged']
 ['0.44603226' 'tenderized']
 ['0.44593066' 'sirloin']
 ['0.43941846' 'calendar']
 ['0.4316482' 'excels']]
```

Adam OPTIMIZER, LR = 0.01, EPOCHS = 10, EMBEDDING_DIM = 300, BATCH_SIZE = 256 window_size= 4

```
[257] words_similar = model.similarity("titanic")
      print(words_similar[0:20])
```

```
[['0.99999994' 'titanic']
 ['0.48740113' 'camerons']
 ['0.32408056' 'lifewhere']
 ['0.3226966' 'instrument']
 ['0.30956832' 'lawnmower']
 ['0.30884796' 'sailboatmast']
 ['0.30057162' 'eliciting']
 ['0.2999454' 'layout']
 ['0.29642433' 'fullest']
 ['0.29570523' 'peculiarly']
 ['0.29553413' 'reselling']
 ['0.2935848' 'boundless']
 ['0.29352662' 'dazzling']
 ['0.29328153' 'passion<NUM>']
 ['0.29205582' 'twitchy']
 ['0.2903595' 'biographicaltype']
 ['0.2894059' 'forrest']
 ['0.28514293' 'zot']
 ['0.28490713' 'gutwrenching']
 ['0.28185594' '<NUM>aramaiclatin<NUM>']]
```

Adam OPTIMIZER, LR = 0.01, EPOCHS = 30, EMBEDDING_DIM = 300, BATCH_SIZE = 256 window_size= 4 neg_samples = 7(default = 5)

```
[264] words_similar = model.similarity("titanic")
      print(words_similar[0:20])
```

```
[['1.0' 'titanic']
 ['0.5660859' 'camerons']
 ['0.39677957' 'licorice']
 ['0.38538793' 'lawnmower']
 ['0.37798175' 'layout']
 ['0.3648934' 'rejoin']
 ['0.36370286' 'samson']
 ['0.36177763' 'alternates']
 ['0.3607466' 'recoiling']
 ['0.36067167' 'conscientious']
 ['0.35434982' 'corruptions']
 ['0.34977806' 'willfully']
 ['0.34574994' 'niggers']
 ['0.3402385' 'graham']
 ['0.33973143' 'xmen']
 ['0.33656037' 'leval']
 ['0.33454627' 'endedi']
 ['0.33424616' 'finalize']
 ['0.33297437' 'expounded']
 ['0.33252022' 'purim']]
```

Reason for unrelated outputs even after increasing epochs is because of higher learning rate. The descent had already crossed the minima 24th epoch.

```
print(words_similar[0:10])
```

```
14.85857367515564
Loss at epo 0: 343.8719177246094
29.04003620147705
Loss at epo 2: 105.64218139648438
29.463810205459595
Loss at epo 4: 61.11689376831055
29.71705913543701
Loss at epo 6: 51.904754638671875
28.861608028411865
Loss at epo 8: 47.29465103149414
29.704654693603516
Loss at epo 10: 43.788978576660156
28.618236303329468
Loss at epo 12: 43.07345962524414
28.403370141983032
Loss at epo 14: 41.28363037109375
28.92903423309326
Loss at epo 16: 40.74709701538086
29.33778429031372
Loss at epo 18: 40.47954177856445
28.560314893722534
Loss at epo 20: 39.16231918334961
29.818763732910156
Loss at epo 22: 39.55759048461914
29.52668285369873
Loss at epo 24: 39.20376968383789
29.683043003082275
Loss at epo 26: 39.40925598144531
29.27295207977295
Loss at epo 28: 39.11177444458008
Total_Training_Time:  445.14245319366455
[['1.0' 'love']
 ['0.3193727' 'ectasy']
 ['0.30190778' 'sprituality']
 ['0.27194932' 'blunted']
 ['0.2646799' 'forgave']
 ['0.26146376' 'analyse']
 ['0.2602146' 'mush']
 ['0.2574338' 'whack']
 ['0.25388128' 'inclusiveness']
 ['0.25275367' 'cruxcified']]
```

Adam OPTIMIZER, LR = 0.01, EPOCHS = 30, EMBEDDING_DIM = 300, BATCH_SIZE = 256 window_size= 4 neg_samples = 7(default = 5)

Adam OPTIMIZER, LR = 0.01, EPOCHS = 1, EMBEDDING_DIM = 300, BATCH_SIZE = 256 window_size= 4

```
[250] words_similar = model.similarity("titanic")
      print(words_similar[0:20])
```

```
[['1.0' 'titanic']
 ['0.57043487' 'camerons']
 ['0.44134235' 'worthwhile']
 ['0.42657244' 'disadvantaged']
 ['0.42148304' 'bucks']
 ['0.41501564' 'resources']
 ['0.41224125' 'girls']
 ['0.4098611' 'pbs']
 ['0.4069561' 'lest']
 ['0.40575147' 'produced']
 ['0.4047393' 'lifethese']
 ['0.4036835' 'cult']
 ['0.4031108' 'ne']
 ['0.40133983' 'agnostic']
 ['0.3969923' 'centered']
 ['0.39575848' 'knowledge']
 ['0.3957284' 'cons']
 ['0.3928674' 'semetism']
 ['0.39220935' 'gleaned']
 ['0.3911061' 'therapist']]
```

Adam OPTIMIZER, LR = 0.01, EPOCHS = 20, EMBEDDING_DIM = 300, BATCH_SIZE = 256 window_size= 4 neg_samples = 5(default = 5)

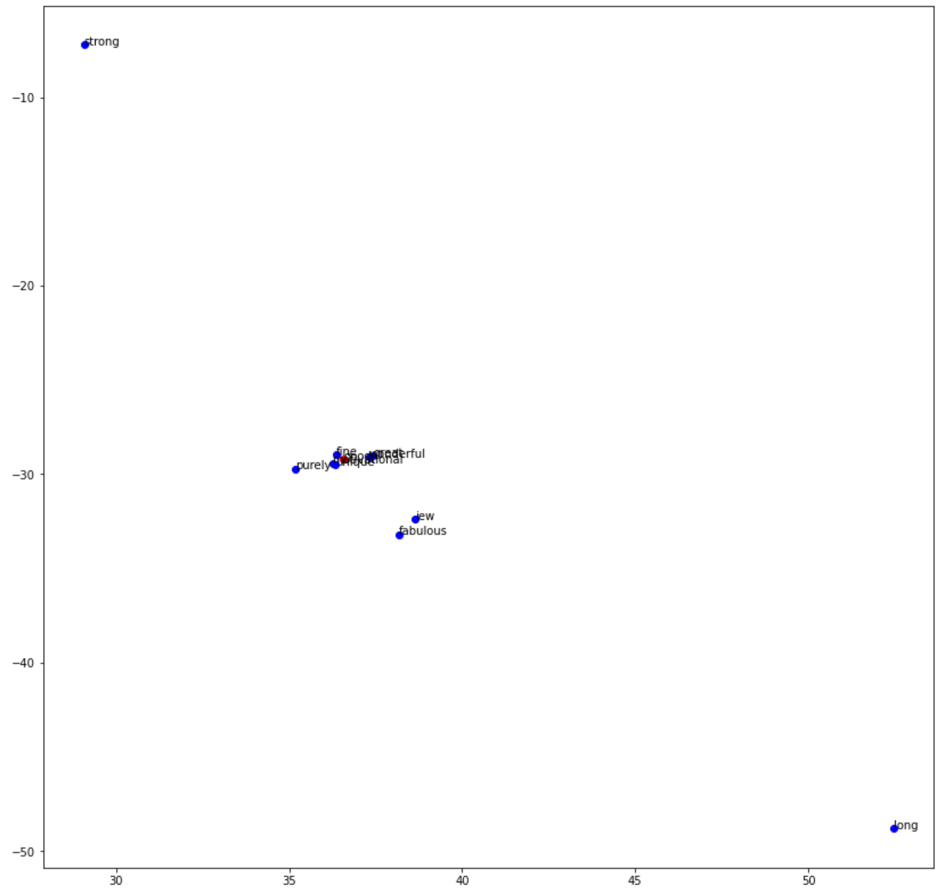
```
words_similar = model.similarity("titanic")
print(words_similar[0:20])
```

```
[['1.0000001' 'titanic']
 ['0.66696143' 'camerons']
 ['0.43803903' 'lawnmower']
 ['0.41648814' 'licorice']
 ['0.39927354' 'presbyterians']
 ['0.39144915' 'willfully']
 ['0.39014193' 'nazipropaganda']
 ['0.37903497' 'toothey']
 ['0.37613216' 'magedeline']
 ['0.37206873' 'chuck']
 ['0.36594698' 'mississippi']
 ['0.35873565' 'layout']
 ['0.35753435' 'spoliers']
 ['0.34681028' 'sailboatmast']
 ['0.34541878' 'waterman']
 ['0.3446798' 'mecurio']
 ['0.3335056' 'antijapanese']
 ['0.3332742' 'dvdbargainbuy']
 ['0.33309022' 'versa']
 ['0.33268923' 'reddyed']]
```

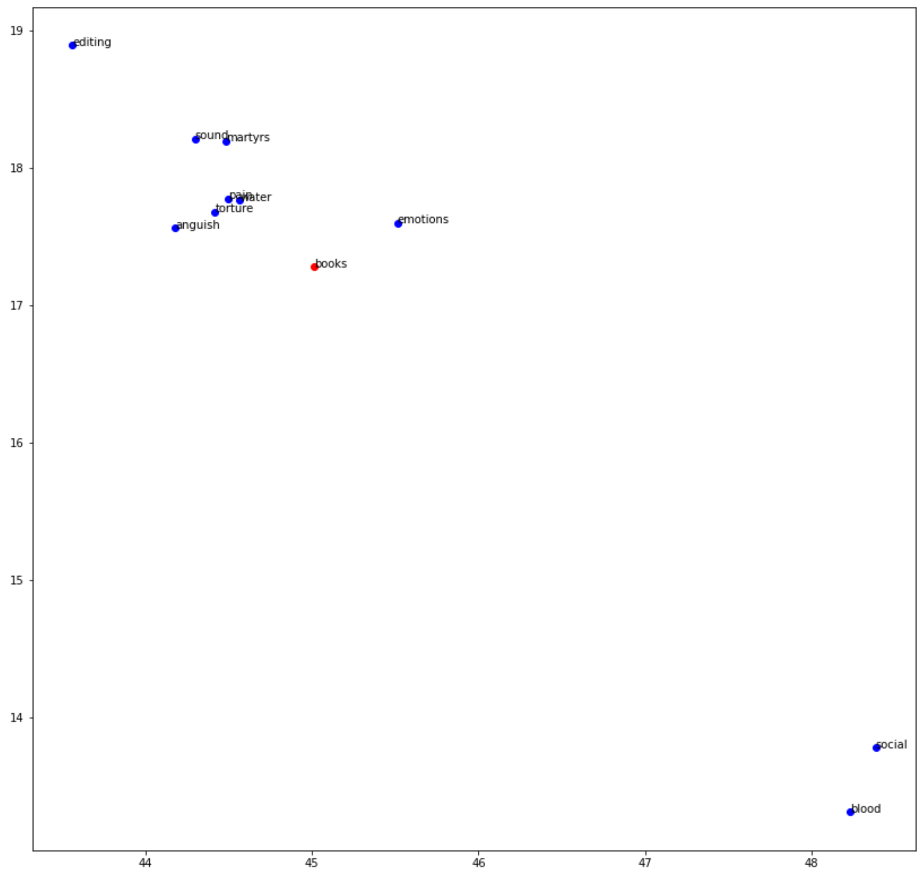
So finally a model with 20 epochs is found to perform better and continued for further functionalities.

PLOTS:

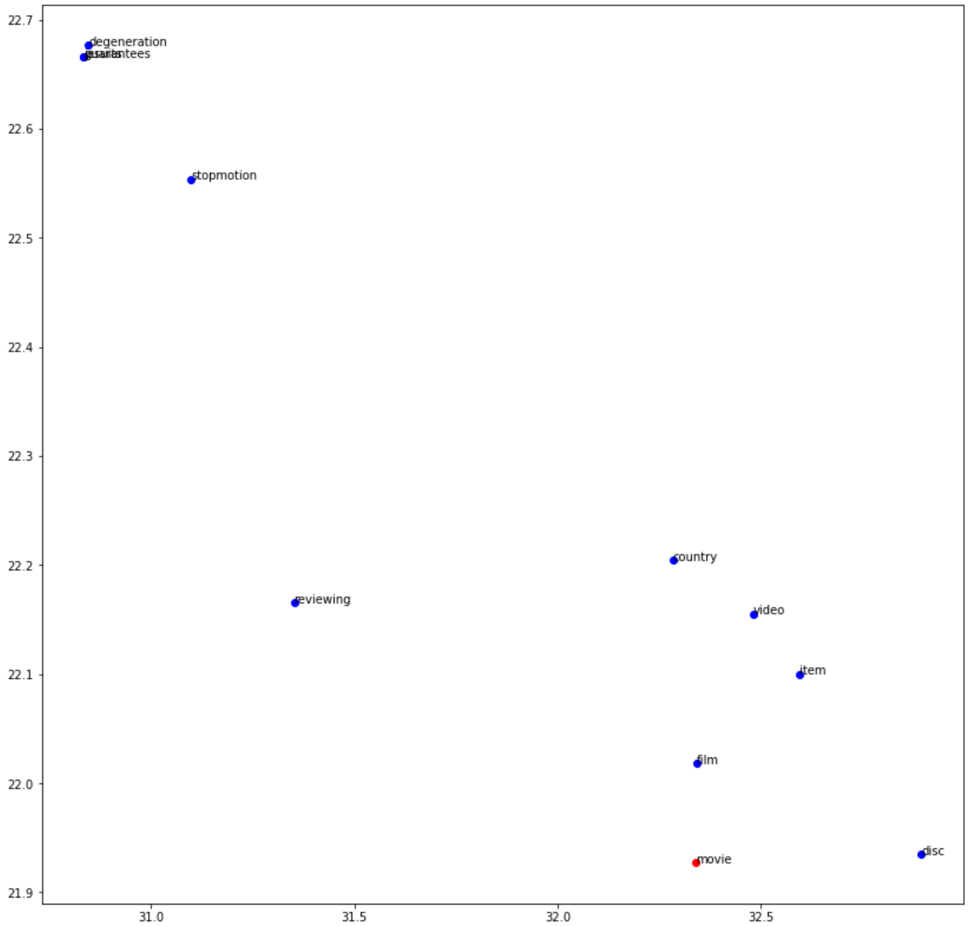
SVD



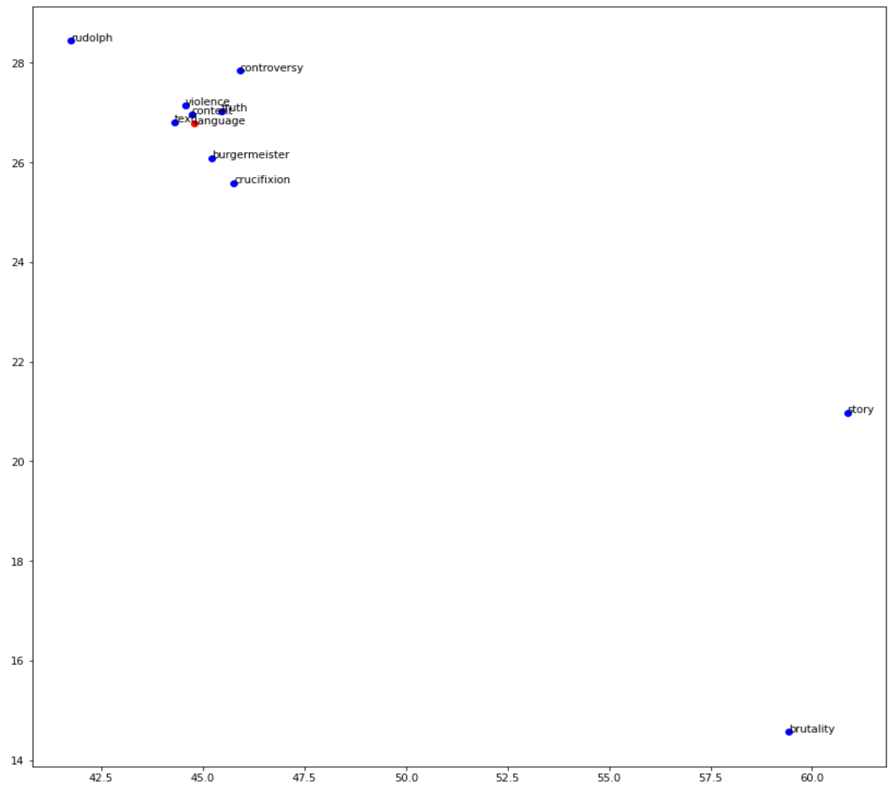
strong
purely
fine
jew
fabulous
long
−10
−20
−30
−40
−50
30
35
40
45
50



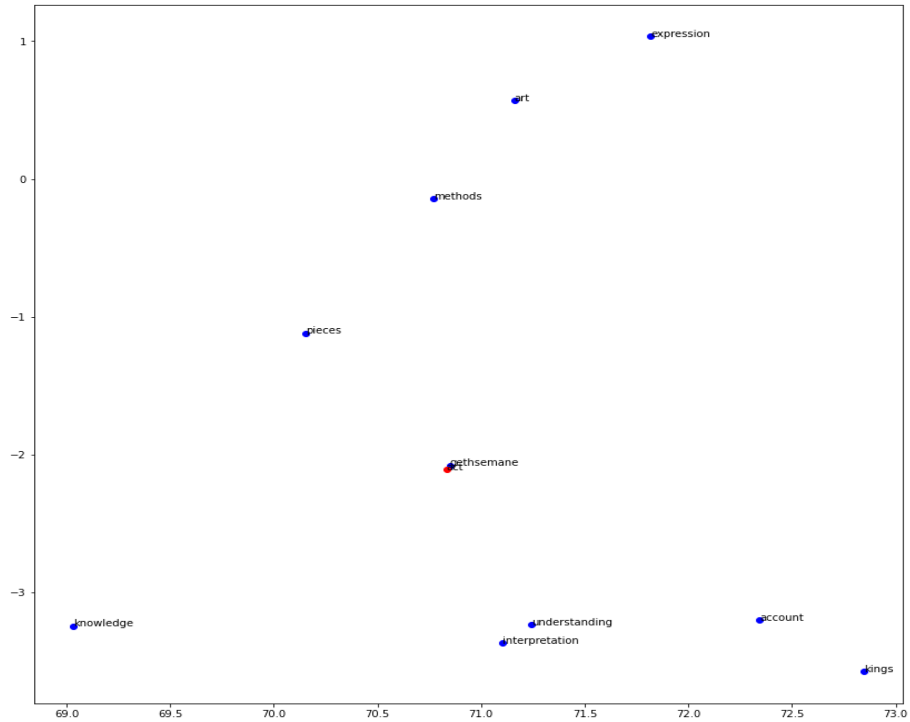
editing
sound
martyrs
pain
water
torture
anguish
emotions
books
social
blood
19
18
17
16
15
14
44
45
46
47
48

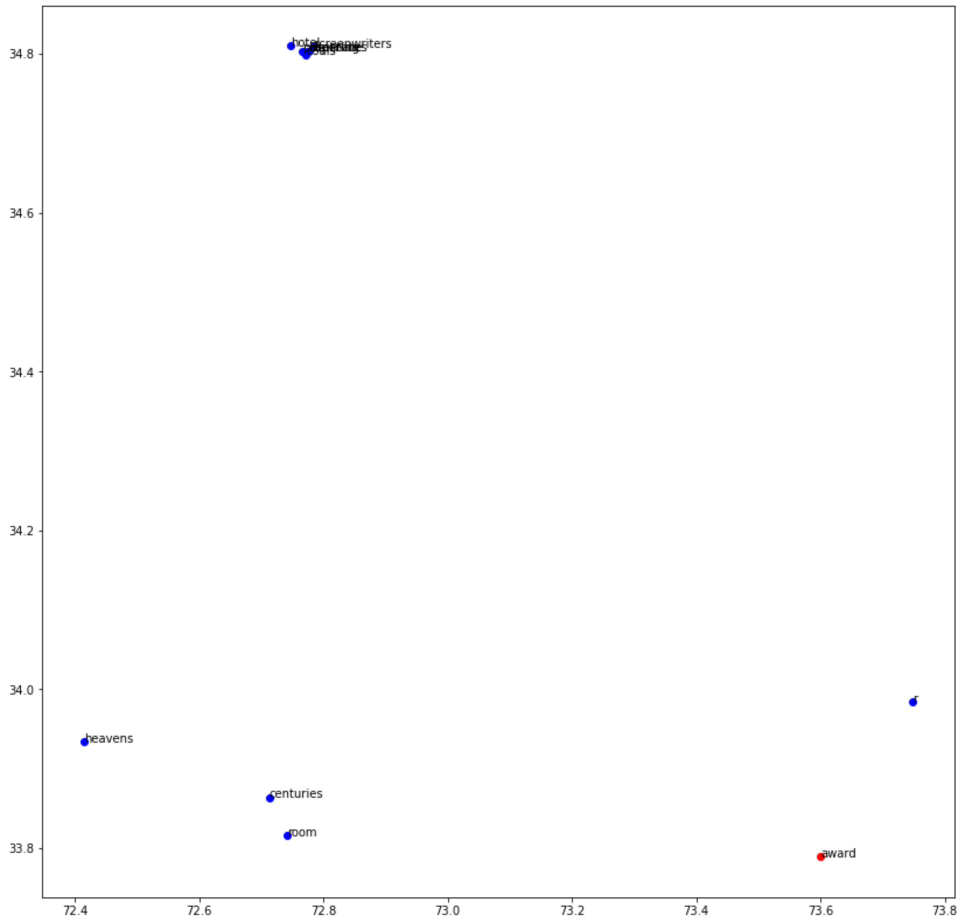


degeneration
guarantees
stopmotion
country
reviewing
video
item
film
movie
disc
22.7
22.6
22.5
22.4
22.3
22.2
22.1
22.0
21.9
31.0
31.5
32.0
32.5

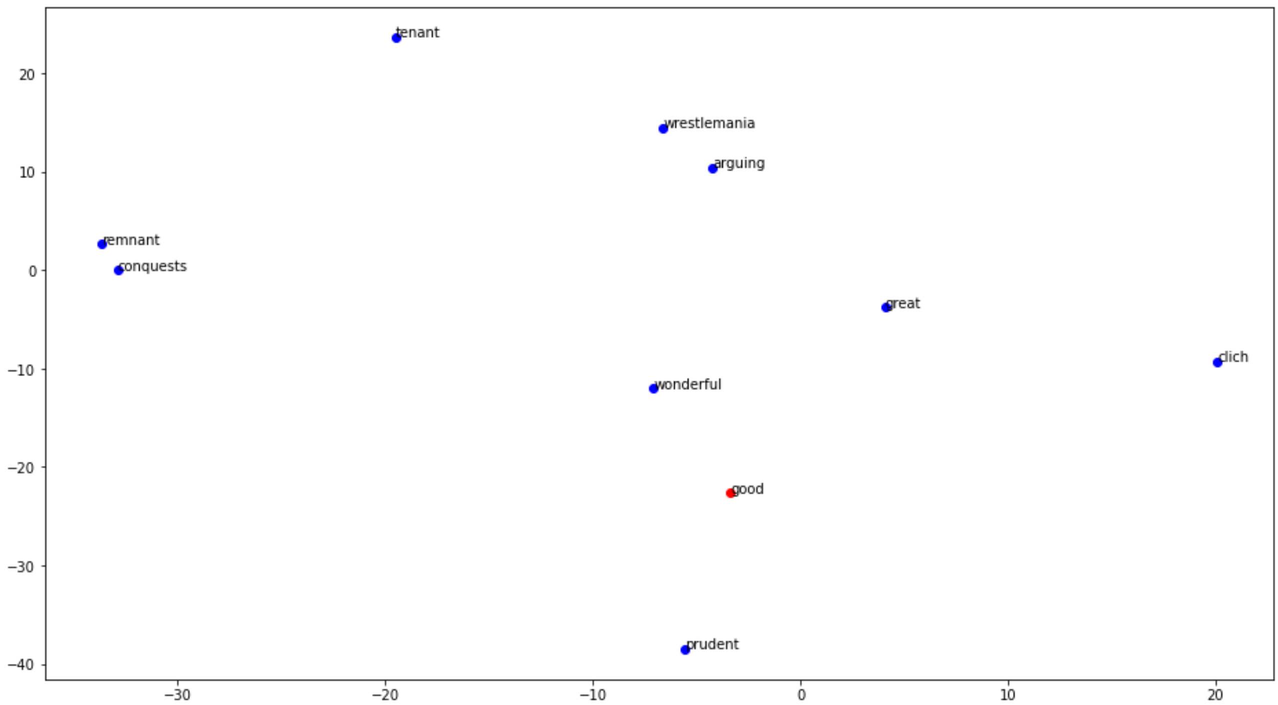


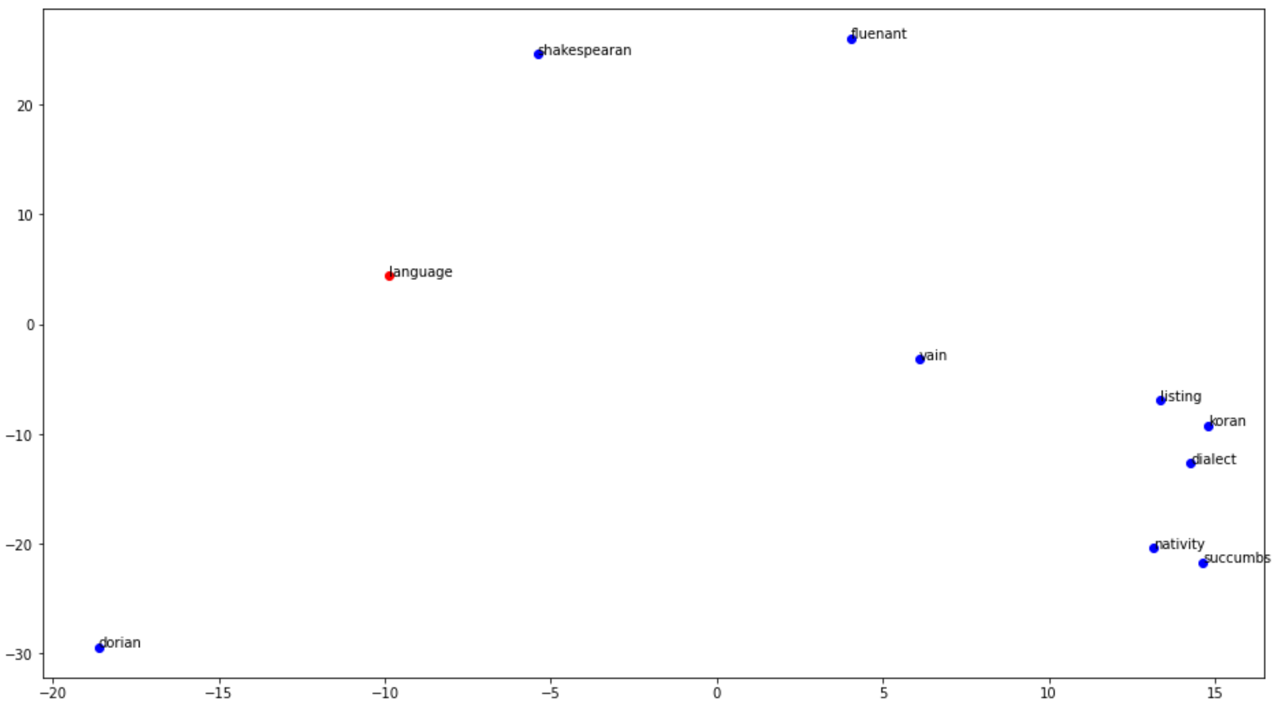
rudolph
controversy
violence
content
truth
text
language
burgermeister
crucifixion
story
brutality
28
26
24
22
20
18
16
14
42.5
45.0
47.5
50.0
52.5
55.0
57.5
60.0

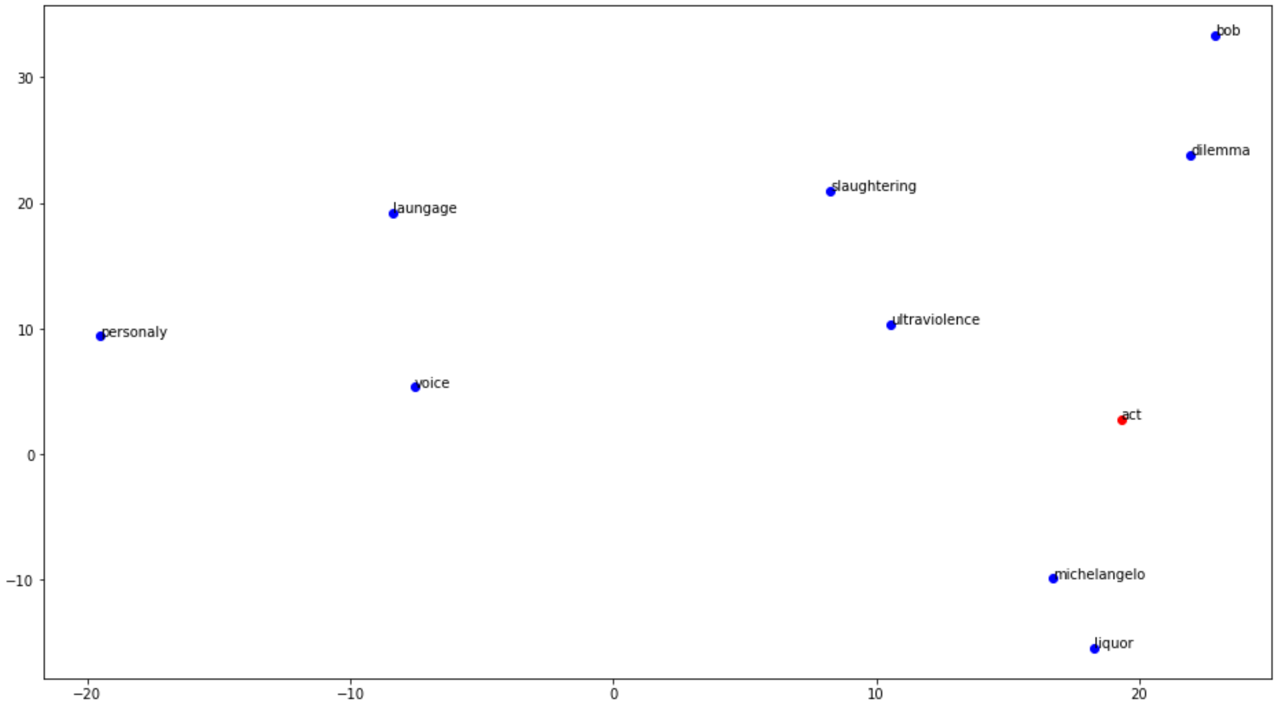

expression
art
methods
pieces
gethsemane
act
knowledge
understanding
interpretation
account
kings

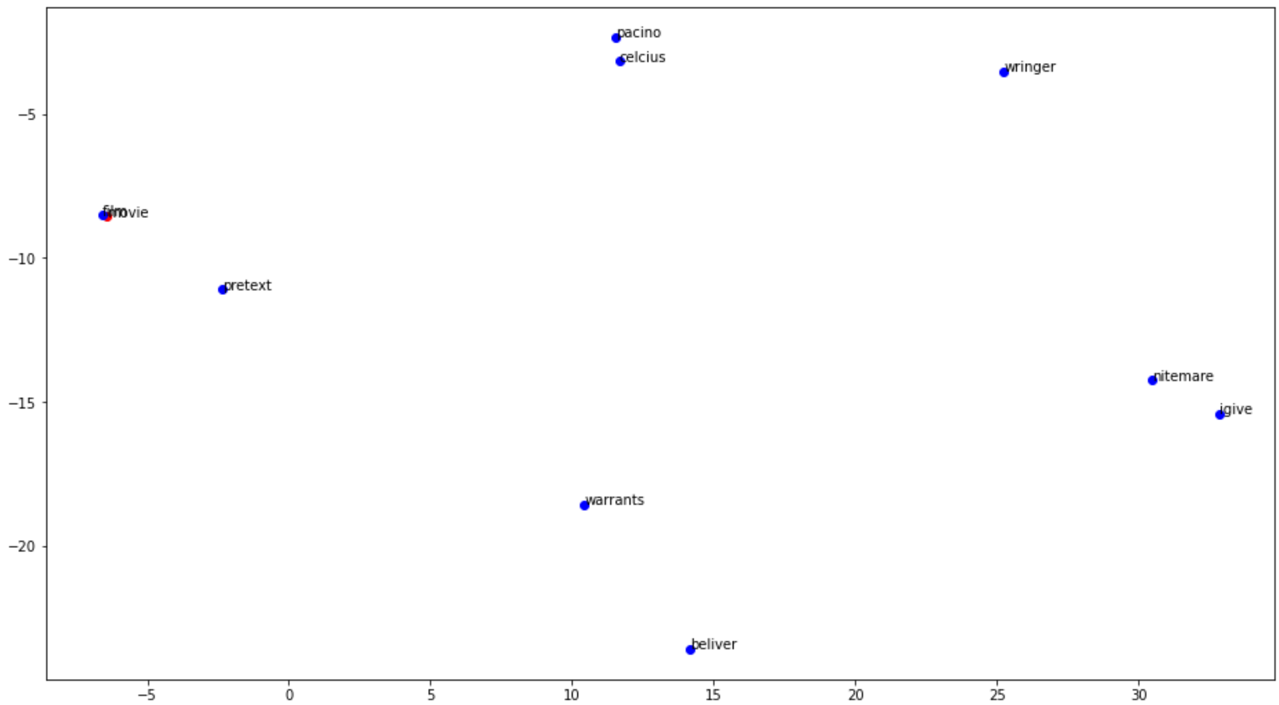

hotel
screenwriters
heavens
centuries
room
award

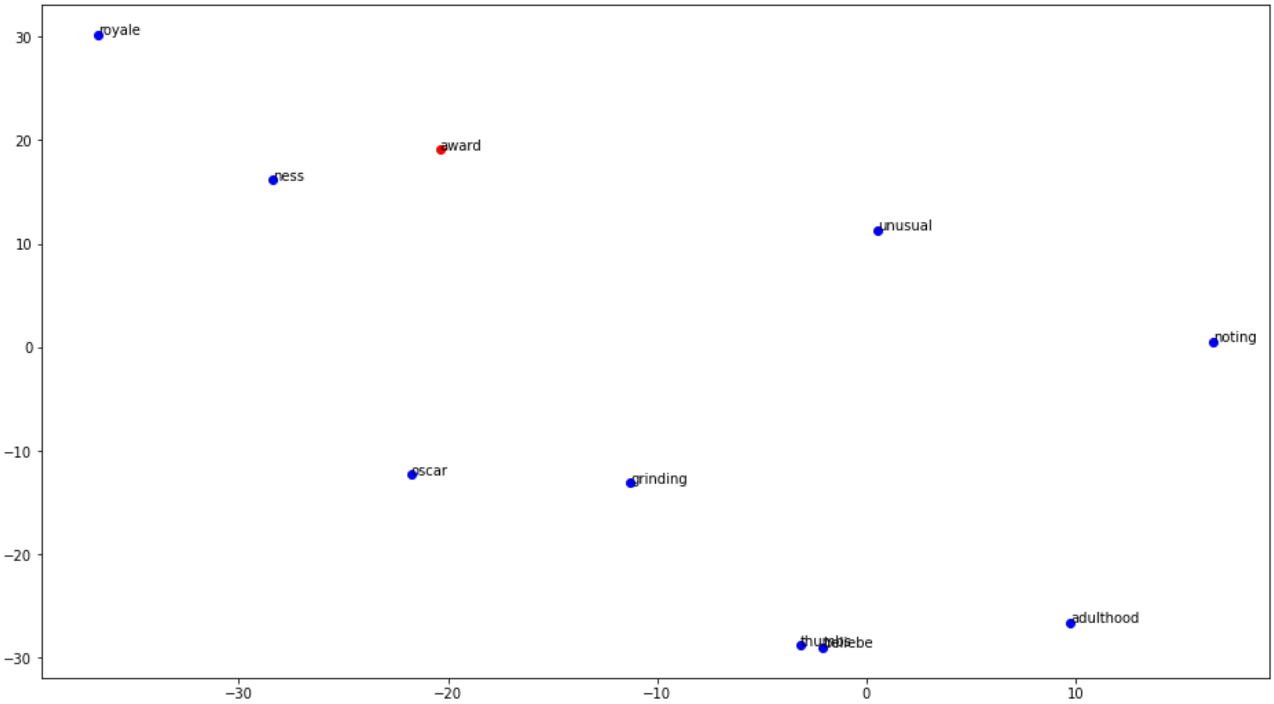
CBOW


tenant
wrestlemania
arguing
remnant
conquests
great
clich
wonderful
good
prudent
20
10
0
−10
−20
−30
−40
−30
−20
−10
0
10
20


fluenant
shakespearan
language
vain
listing
koran
dialect
nativity
succumbs
dorian
20
10
0
−10
−20
−30
−20
−15
−10
−5
0
5
10
15


bob
dilemma
slaughtering
laungage
personaly
ultraviolence
voice
act
michelangelo
liquor
30
20
10
0
−10
−20
−10
0
10
20


pacino
celcius
wringer
film
movie
pretext
nitemare
igive
warrants
beliver
−5
−10
−15
−20
−5
0
5
10
15
20
25
30



Cbow

```
[7] words_similar = model.similarity("titanic")
    print(words_similar[0:20])

    [['1.0' 'titanic']
     ['0.43075538' 'magedeline']
     ['0.40516475' 'licorice']
     ['0.3797103' 'cameron']
     ['0.36835608' 'adl']
     ['0.36531895' 'indulged']
     ['0.3641403' 'innoscense']
     ['0.35763428' 'conscientious']
     ['0.35566515' 'finalize']
     ['0.35406017' 'smelling']
     ['0.3510035' 'spoliers']
     ['0.34380376' 'hubris']
     ['0.34120697' 'precluded']
     ['0.34069604' 'christi']
     ['0.33160335' 'expounded']
     ['0.32953504' 'cylinders']
     ['0.32919604' 'dissappoints']
     ['0.32826293' 'monkey']
     ['0.32642934' 'knifes']
     ['0.3255321' 'revisionists']]
```

The nearest words according to google-news-300:

('colossal', 0.5896502137184143)
('gargantuan', 0.5718227028846741)
('titanic proportions', 0.5610266923’)
('titantic', 0.5592556595802307)
('monumental', 0.5530510544776917)
('monstrous', 0.5457675457000732)
('epic_proportions', 0.543700397014’)
('gigantic', 0.5176911950111389)
('mighty', 0.5088781118392944)
('epic', 0.6006164550781 25)

SVD

```
# Define the word for which to find similar words
word = "titanic"

# Find the 10 most similar words to the given word
most_similar = [vocab[i] for i in np.argsort(np.linalg.norm(word_vectors_normalized - wor
for a in most_similar:
    print(a)
```

```
thou
barbarian
older
goriness
other
profundity
transformers
awaited
julie
python
lastly
anyhow
forceful
thus
misplaced
implore
victor
dated
workbook
eloquently
```

Observation:
Cbow with negative samplings out performs svd.