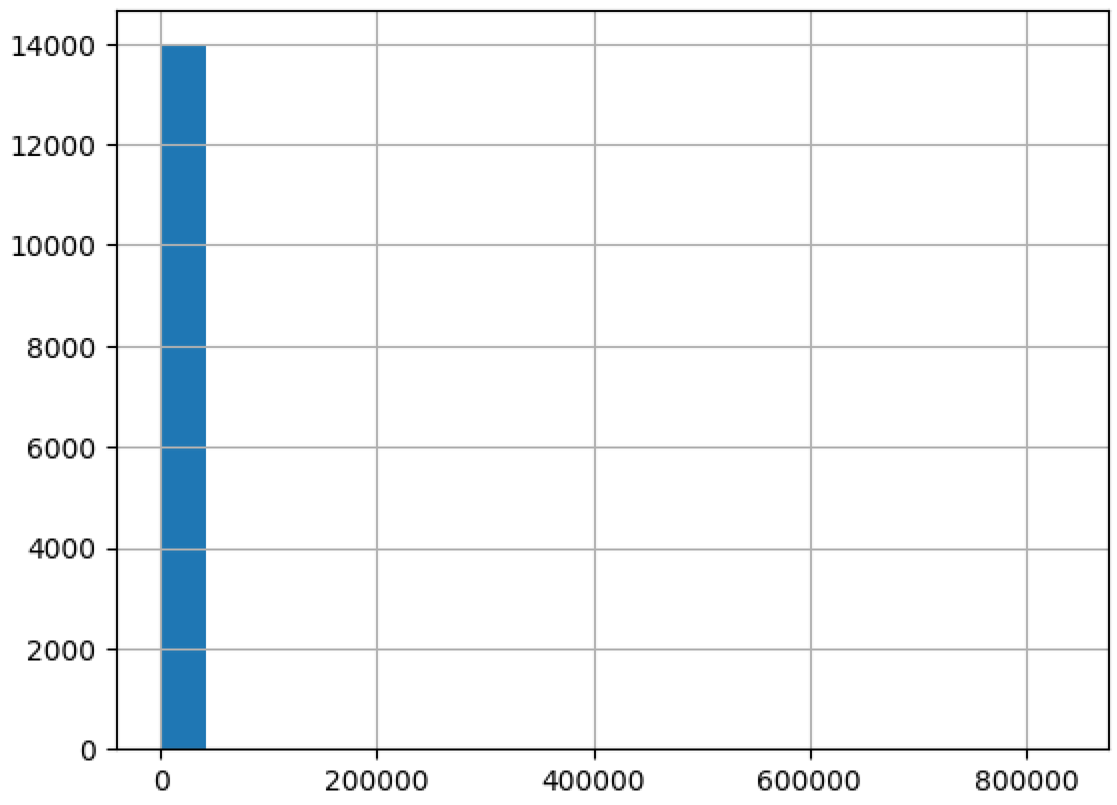
Analysis:

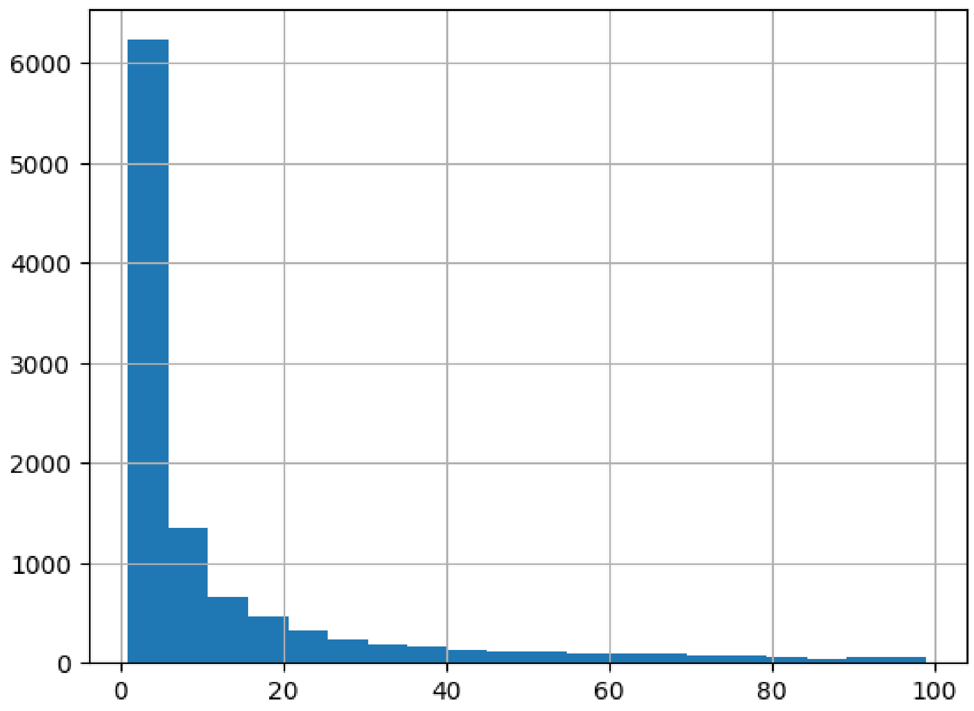
1. Word frequency analysis:

Word Count Histogram:



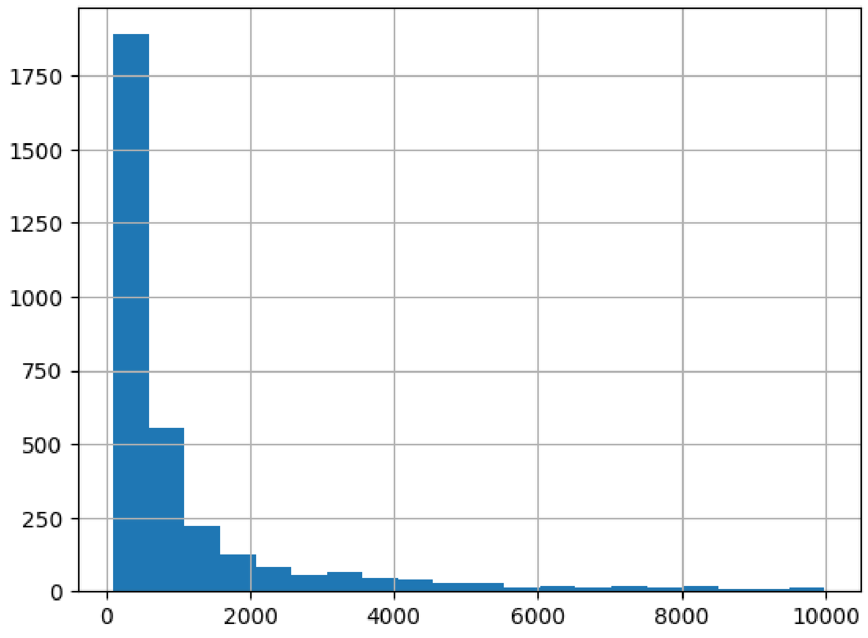
As can be seen, most words in the vocabulary have a count of less than 100. Let's look at the histogram for counts for various ranges

when counts < 100:

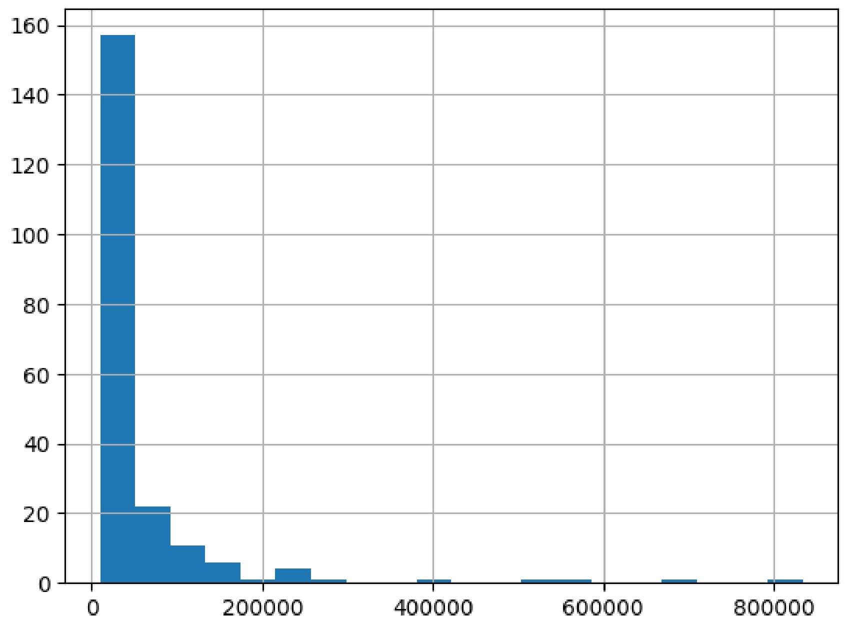


```
len(counts.loc[counts == 1])
3266
```

when counts > 100 and counts <=10000



when counts > 10000



let’s look at the words with very high counts

```
[('the', (0, 833516)), ('and', (1, 678137)), ('to', (2, 555648)), ('a',
(3, 543555)), ('was', (4, 391267)), ('she', (5, 262987)), ('he', (6,
254659)), ('it', (7, 251852)), ('they', (8, 246987)), ('her', (9,
225202)), ('lily', (10, 193492)), ('day', (11, 157142)), ('with', (12,
149212)), ('said', (13, 148892)), ('his', (14, 147392)), ('in', (15,
135190)), ('that', (16, 133920)), ('but', (17, 116605)), ('s', (18,
111434)), ('you', (19, 110824))]
```

These are mostly articles, prepositions, verbs, adjectives and adverbs, used frequently.

Training Procedure:

Dataset:
Looking at the word count distribution and following the method described in paper, we have used the entire dataset for training. Few reasons are highlighted below:

1. Ideally the quality of embeddings is checked using downstream tasks. Google provided 20000 syntactic, semantic similarity tasks for evaluation.

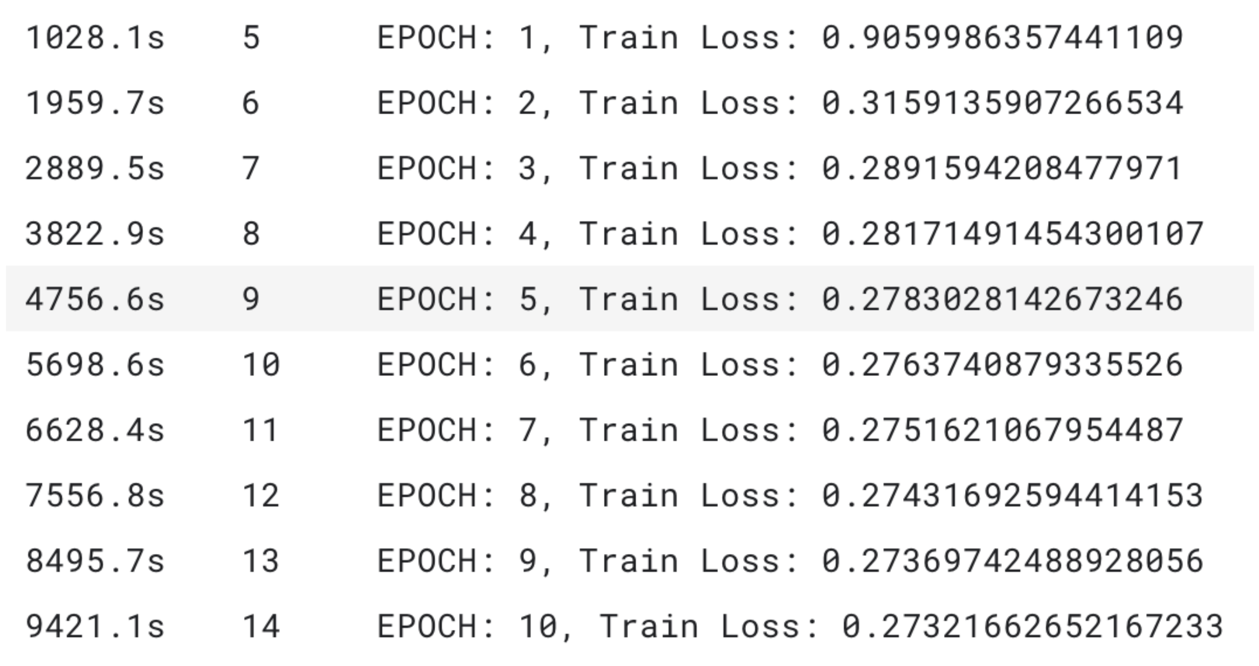
2. We can create a held out set from the training corpus for evaluation using the same loss function however, since a large amount of words (3266) occur only once in corpus, hence random splitting of corpus leads to absence of those words from training, which will result in no training for these words.
3. We train till the loss function flatlines
4. Vocab size = 14040
5. Since the corpus size is less we choose an embedding dimension of 64, to reduce overfitting(but has to be checked with downstream tasks). T value is also increased so that relatively more frequent words are selected for training. Negative sample size is 10, as directed in paper (they instruct to use between 5-20, for small datasets)

Important Parameters:

SKIPGRAM_N_WORDS = 7
T = 1e-3
NEG_SAMPLES = 10
EMBED_DIM = 64
EPOCHS = 10

LOSS = Binary Cross Entropy(as described in paper)

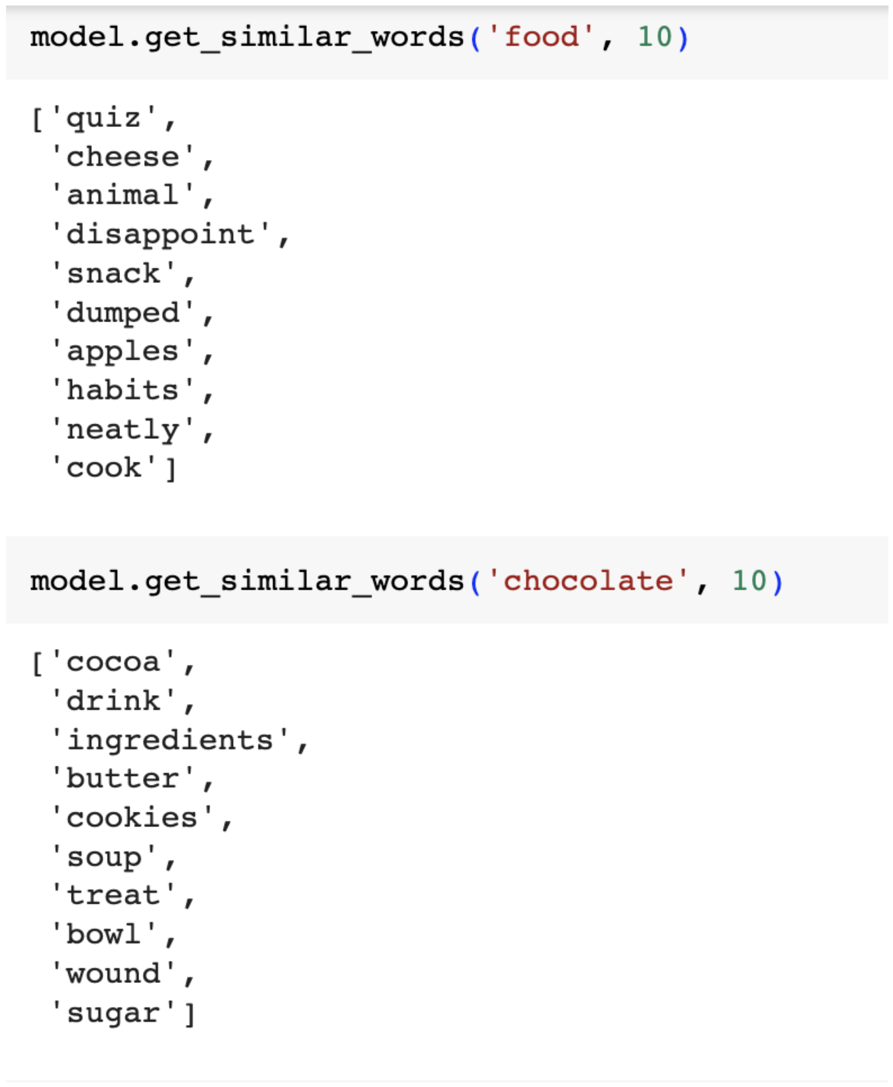
Training loss variation :



```
1028.1s    5      EPOCH: 1, Train Loss: 0.9059986357441109
1959.7s    6      EPOCH: 2, Train Loss: 0.3159135907266534
2889.5s    7      EPOCH: 3, Train Loss: 0.2891594208477971
3822.9s    8      EPOCH: 4, Train Loss: 0.28171491454300107
4756.6s    9      EPOCH: 5, Train Loss: 0.2783028142673246
5698.6s    10     EPOCH: 6, Train Loss: 0.2763740879335526
6628.4s    11     EPOCH: 7, Train Loss: 0.2751621067954487
7556.8s    12     EPOCH: 8, Train Loss: 0.27431692594414153
8495.7s    13     EPOCH: 9, Train Loss: 0.27369742488928056
9421.1s    14     EPOCH: 10, Train Loss: 0.2732166265216723
```

Results:

Since the dataset is extremely small, the embeddings are not able to extract all the semantic and syntactic meanings. But we have evaluated the model on some similarity and analogical tasks(`“Germany” : “Berlin” :: “France” : ? ans:"Paris"`). Results are shown below:

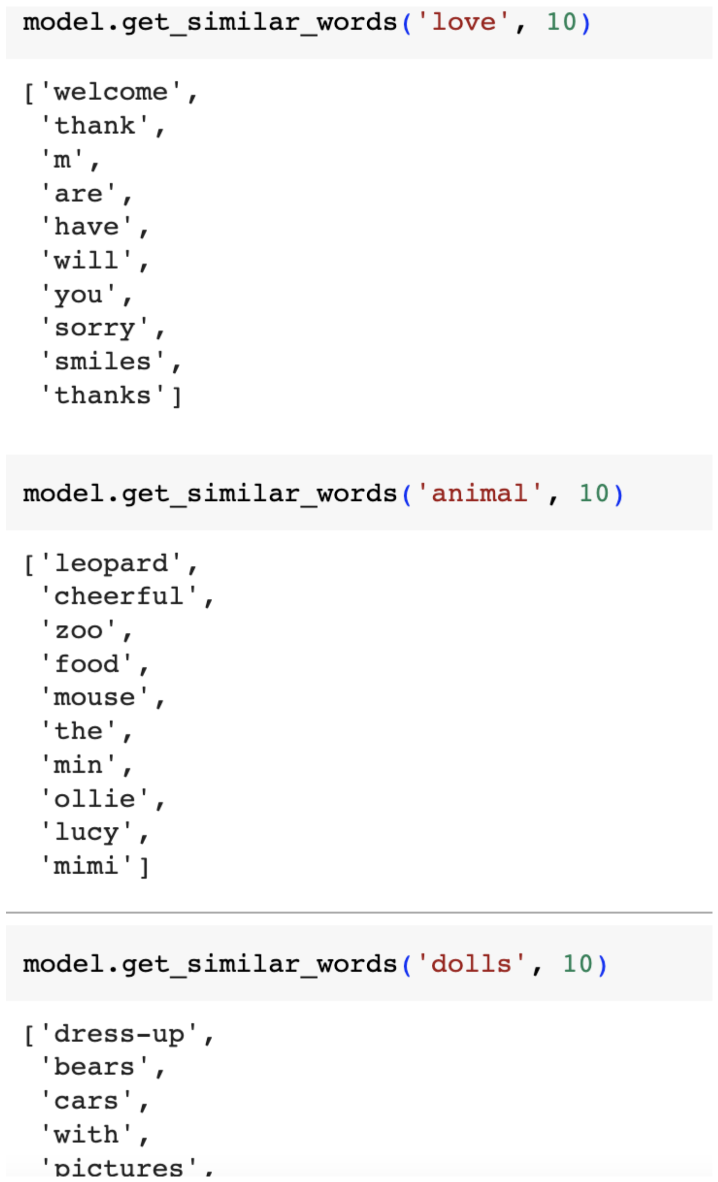


```
model.get_similar_words('food', 10)
```

```
['quiz',
 'cheese',
 'animal',
 'disappoint',
 'snack',
 'dumped',
 'apples',
 'habits',
 'neatly',
 'cook']
```

```
model.get_similar_words('chocolate', 10)
```

```
['cocoa',
 'drink',
 'ingredients',
 'butter',
 'cookies',
 'soup',
 'treat',
 'bowl',
 'wound',
 'sugar']
```



```
model.get_similar_words('love', 10)
```

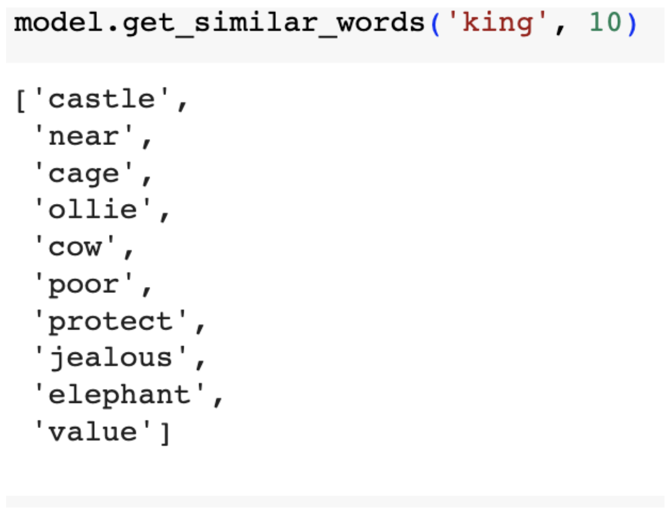
```
['welcome',
 'thank',
 'm',
 'are',
 'have',
 'will',
 'you',
 'sorry',
 'smiles',
 'thanks']
```

```
model.get_similar_words('animal', 10)
```

```
['leopard',
 'cheerful',
 'zoo',
 'food',
 'mouse',
 'the',
 'min',
 'ollie',
 'lucy',
 'mimi']
```

```
model.get_similar_words('dolls', 10)
```

```
['dress-up',
 'bears',
 'cars',
 'with',
 'pictures',
```



```
model.get_similar_words('king', 10)
```

```
['castle',
 'near',
 'cage',
 'ollie',
 'cow',
 'poor',
 'protect',
 'jealous',
 'elephant',
 'value']
```

```
model.get_similar_words('lion', 10)
```

```
['brave',
 'fox',
 'tim',
 'leo',
 'rabbit',
 'monkey',
 'bobo',
 'freddy',
 'monster',
 'frog']
```

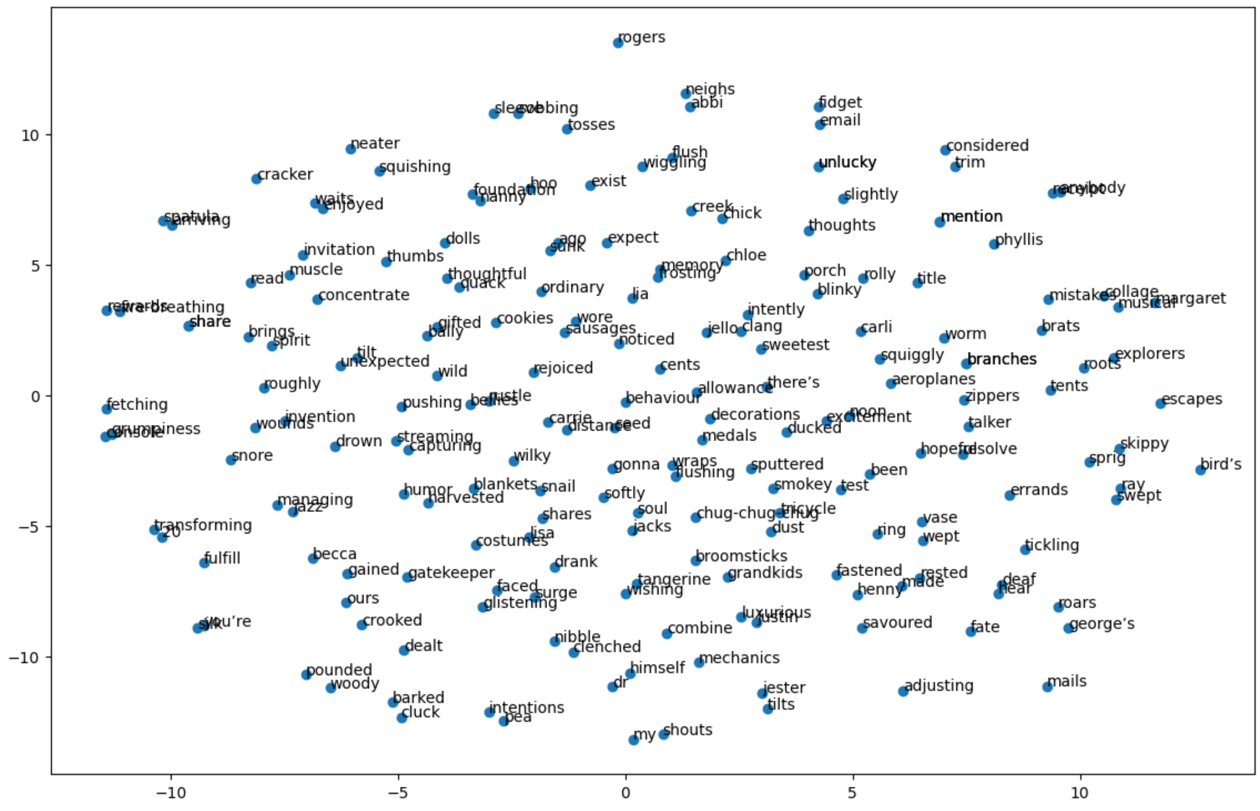
```
model.get_analogy('prince', 'princess', 'boy', 10)
```

```
['friends',
 'ducky',
 'above',
 'brave',
 'heel',
 'pigeon',
 'task',
 'problem',
 'ending',
 'little']
```

As can be seen some of the similar words do make sense, like a king will live in a castle, chocolate and cocoa are similar, children do make their dolls play dress-up etc. Results on analogy tasks are not that great in our observations hence only one example is shown.

We can do more experiments by changing the size of embeddings, and the size of negative sampling for better contrastive estimation.

Here is a 2d plot of some sampled word embeddings using T-SNE dimensionality reduction.


rogers
neighs
abbi
fidget
email
sleeve
sobbing
tosses
considered
trim
neater
flush
wiggling
unlucky
squishing
cracker
foundation
hoo
nanny
exist
slightly
everybody
receipt
waits
enjoyed
creek
chick
mention
spatula
arriving
thoughts
phyllis
dolls
ago
sunk
expect
invitation
thumbs
chloe
muscle
read
thoughtful
quack
memory
frosting
porch
rolly
title
ordinary
concentrate
lia
blinky
mistakes
collage
margaret
musical
rewards
fire-breathing
intently
share
cookies
gifted
wore
sausages
jello
clang
carli
brats
brings
spirit
bally
noticed
sweetest
worm
squiggly
branches
explorers
tilt
unexpected
cents
roots
wild
rejoiced
aeroplanes
roughly
allowance
there’s
tents
fetching
pushing
bellies
rustle
behaviour
zippers
escapes
invention
wounds
carrie
distance
seed
decorations
noon
excitement
grumpiness
console
streaming
capturing
medals
ducked
talker
drown
hopeful
resolve
skippy
snore
wilky
gonna
wraps
flushing
sputtered
been
sprig
bird’s
humor
harvested
blankets
snail
smokey
test
ray
errands
swept
managing
jazz
softly
soul
tricycle
shares
chug-chug-chug
transforming
jacks
dust
vase
ring
wept
costumes
lisa
tickling
becca
fulfill
drank
broomsticks
gained
gatekeeper
grandkids
fastened
rested
made
deaf
faced
tangerine
henny
hear
surge
wishing
ours
glistening
luxurious
justin
roars
you’re
silk
crooked
combine
savoured
fate
george’s
nibble
dealt
clenched
mechanics
pounded
woody
himself
dr
jester
adjusting
mails
barked
intentions
tilts
cluck
pea
shouts
my