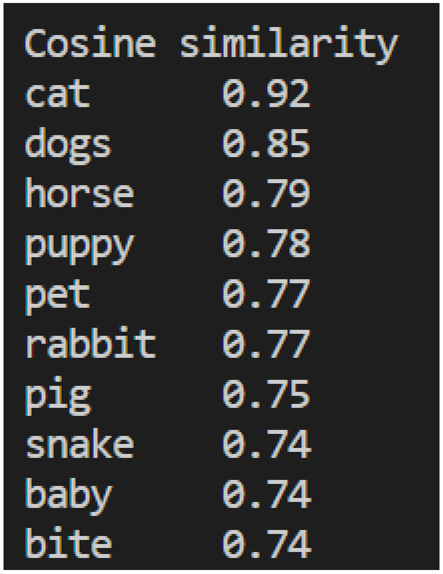
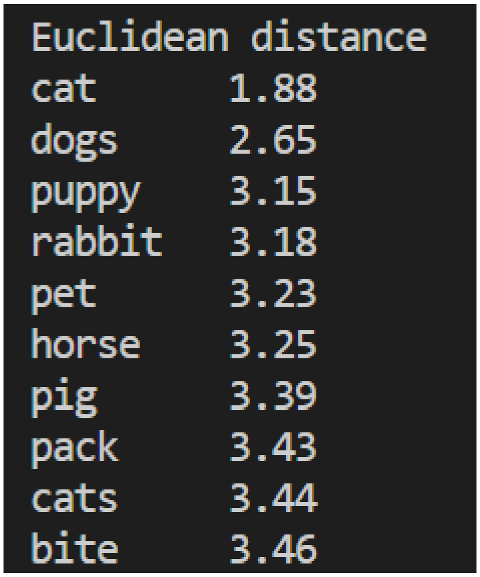
A1P1_2

Provide a table that compares the 10-most cosine-similar words to the word ‘dog’, in order, alongside to the 10 closest words computed using Euclidean distance.

Dog:

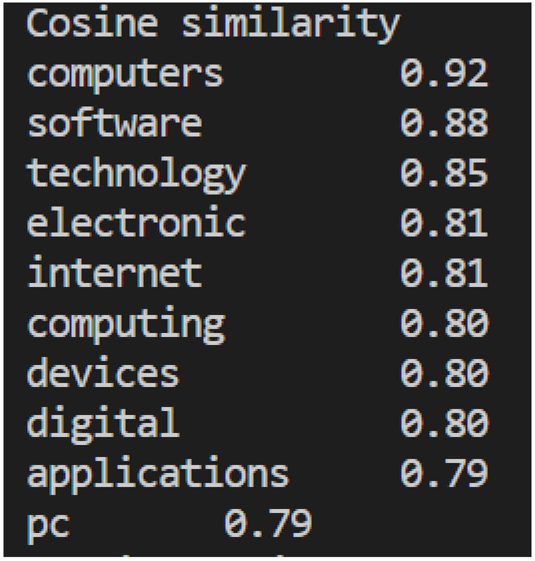


```
Cosine similarity
cat       0.92
dogs      0.85
horse     0.79
puppy     0.78
pet       0.77
rabbit    0.77
pig       0.75
snake     0.74
baby      0.74
bite      0.74
```

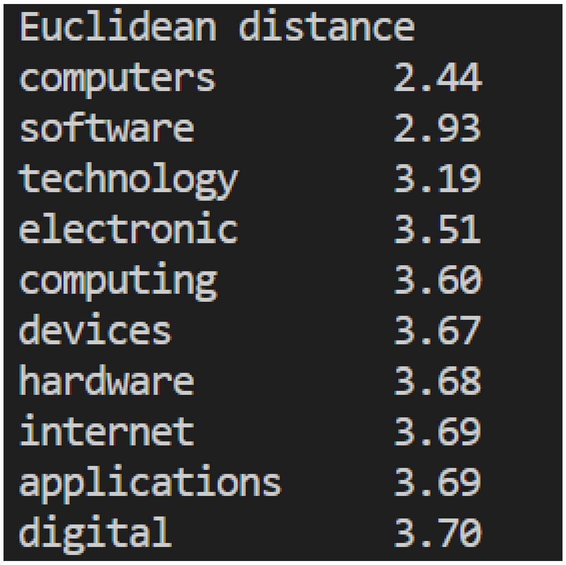


```
Euclidean distance
cat       1.88
dogs      2.65
puppy     3.15
rabbit    3.18
pet       3.23
horse     3.25
pig       3.39
pack      3.43
cats      3.44
bite      3.46
```

Computer:



```
Cosine similarity
computers        0.92
software         0.88
technology       0.85
electronic       0.81
internet         0.81
computing        0.80
devices          0.80
digital          0.80
applications     0.79
pc       0.79
```



```
Euclidean distance
computers        2.44
software         2.93
technology       3.19
electronic       3.51
computing        3.60
devices          3.67
hardware         3.68
internet         3.69
applications     3.69
digital          3.70
```

Looking at the two lists, does one of the metrics (cosine similarity or Euclidean distance) seem to be better than the other?

No. Lots of words in the list are identical. The performance of both metrics seems reasonable regarding to the close/similar words. One cannot tell which one is better just based on the two lists above.

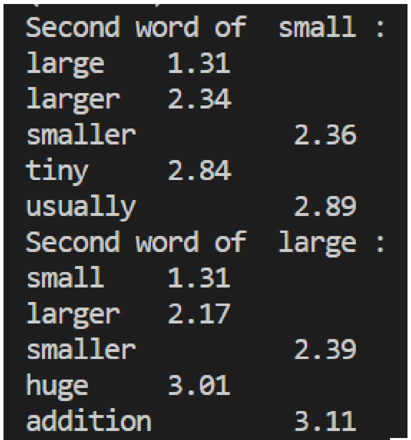
A1P1_3

Generate 10 more examples of that same relationship from 10 other words, and comment on the quality of the results.

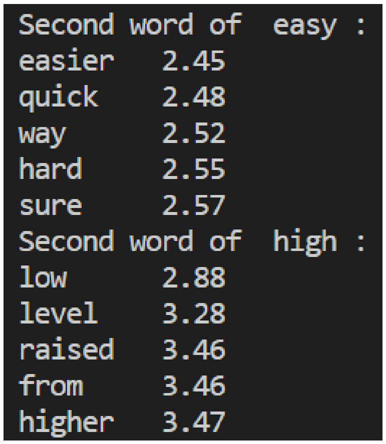
Relationship: Comparative. Example: Greater – Great

Test set: 'small', "large", "easy", "high", "low", "slow", "tall", "short", "fat", "thin"

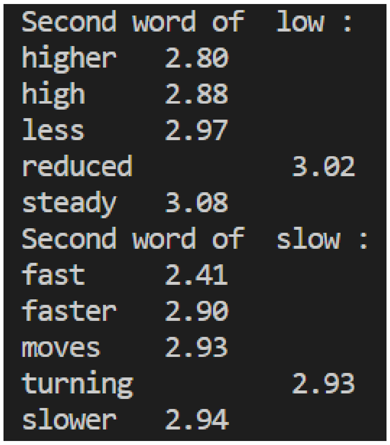
Results:



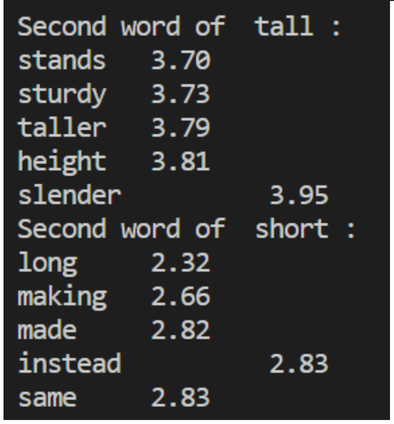
```
Second word of  small :
large    1.31
larger   2.34
smaller          2.36
tiny     2.84
usually          2.89
Second word of  large :
small    1.31
larger   2.17
smaller          2.39
huge     3.01
addition         3.11
```



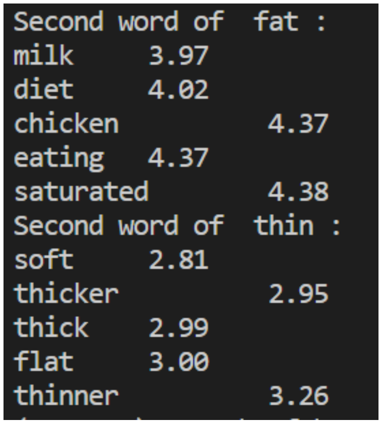
```
Second word of  easy :
easier   2.45
quick    2.48
way      2.52
hard     2.55
sure     2.57
Second word of  high :
low      2.88
level    3.28
raised   3.46
from     3.46
higher   3.47
```



```
Second word of  low :
higher   2.80
high     2.88
less     2.97
reduced          3.02
steady   3.08
Second word of  slow :
fast     2.41
faster   2.90
moves    2.93
turning          2.93
slower   2.94
```



```
Second word of  tall :
stands   3.70
sturdy   3.73
taller   3.79
height   3.81
slender          3.95
Second word of  short :
long     2.32
making   2.66
made     2.82
instead          2.83
same     2.83
```



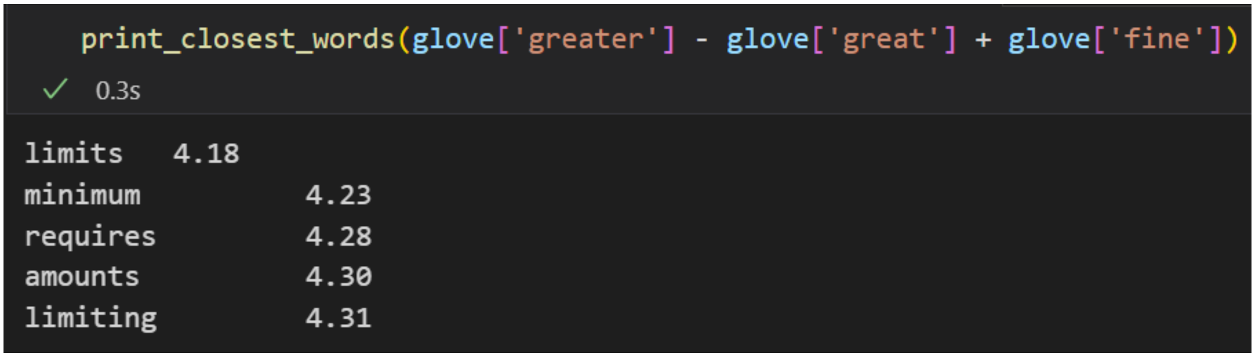
```
Second word of  fat :
milk     3.97
diet     4.02
chicken          4.37
eating   4.37
saturated        4.38
Second word of  thin :
soft     2.81
thicker          2.95
thick    2.99
flat     3.00
thinner          3.26
```

I use Euclidean distance for the second word by giving the first word. The pattern is testing word + “greater” – “great”. However, only “easy” get the expected answer that the closest distance of the adjective is the comparative form “easier”. The rest of the words in the test set does not generate the expected answer from the closest one, but we could find the answer from top 5 closest words except “tall”.

A1P1_4

Choose a context that you’re aware of (different from those already in the notebook) and see if you can find evidence of a bias that is built into the word vectors. Report the evidence and the conclusion you make from the evidence.

Bias example/evidence：



```
print_closest_words(glove['greater'] - glove['great'] + glove['fine'])
```

✓ 0.3s

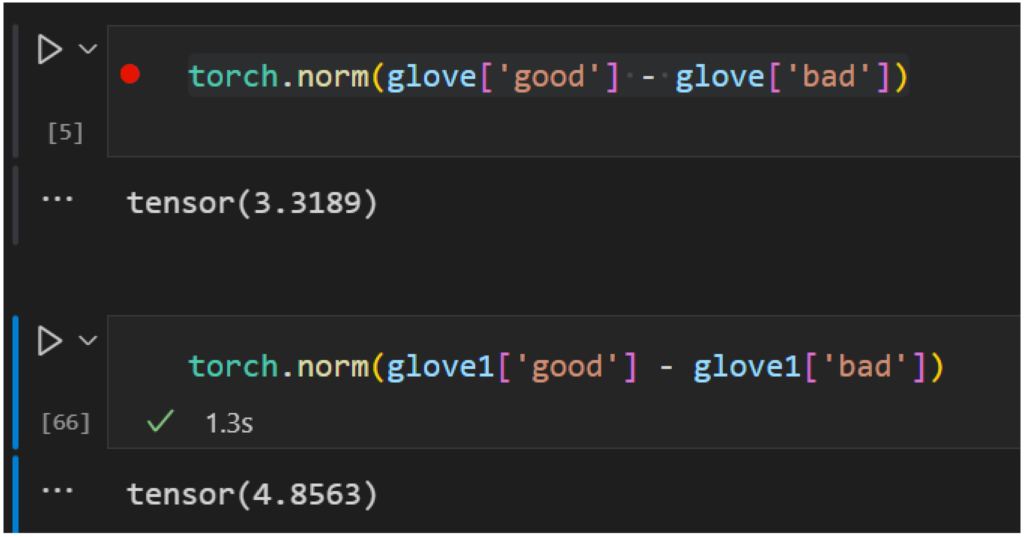
```
limits   4.18
minimum          4.23
requires         4.28
amounts          4.30
limiting         4.31
```

Conclusion: Vector similarity: By calculating the cosine similarity between word vectors, we may observe that words like “fine” has some ambiguous meaning. One is adjective, one is noun means penalty. The evidence shows the context associated with these words is for the seconding meaning, which is penalty. So, the similar words of “fine” was shown as above. This indicates a bias in the word vectors, where certain words are implicitly associated with one of the meanings from the training data. This bias can have implications in various applications, such as natural language processing systems, the more frequently used meaning will be shown as more similar words for the Analogies.

A1P1_5

How does the Euclidean difference change between the various words in the notebook when switching from d=50 to d=300? How does the cosine similarity change?

Euclidean example:



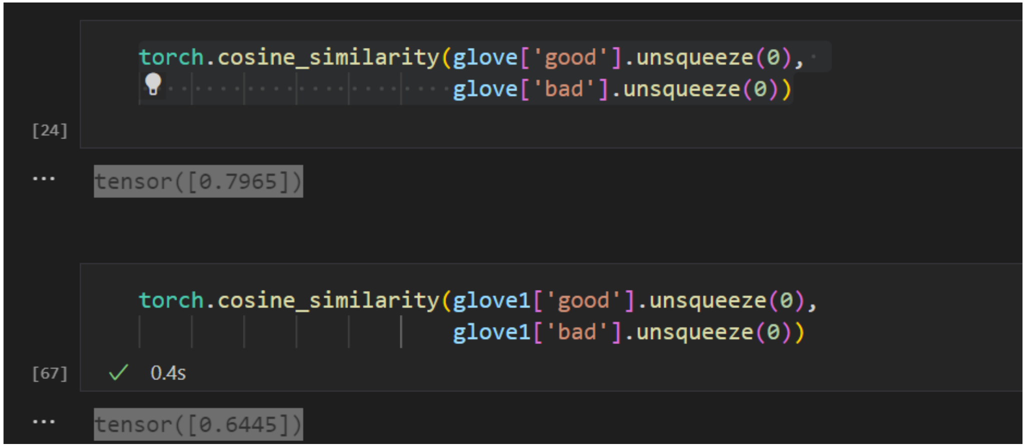
```
[5]
torch.norm(glove['good'] - glove['bad'])
```

```
tensor(3.3189)
```

```
[66]
torch.norm(glove1['good'] - glove1['bad'])
✓ 1.3s
```

```
tensor(4.8563)
```

Cosine example:



```
[24]
torch.cosine_similarity(glove['good'].unsqueeze(0),
                        glove['bad'].unsqueeze(0))
```

```
tensor([0.7965])
```

```
[67]
torch.cosine_similarity(glove1['good'].unsqueeze(0),
                        glove1['bad'].unsqueeze(0))
✓ 0.4s
```

```
tensor([0.6445])
```

Euclidean distance increases when dimension increases, which makes sense because more dimensions will increase the norm based on the formula. Cosine similarity distance was reduced a little bit. That might because the cosine value is in [-1,1], more dimension, may cause the distance for the similar words angle smaller.

Does the ordering of nearness change? Is it clear that the larger size vectors give better results - why or why not?

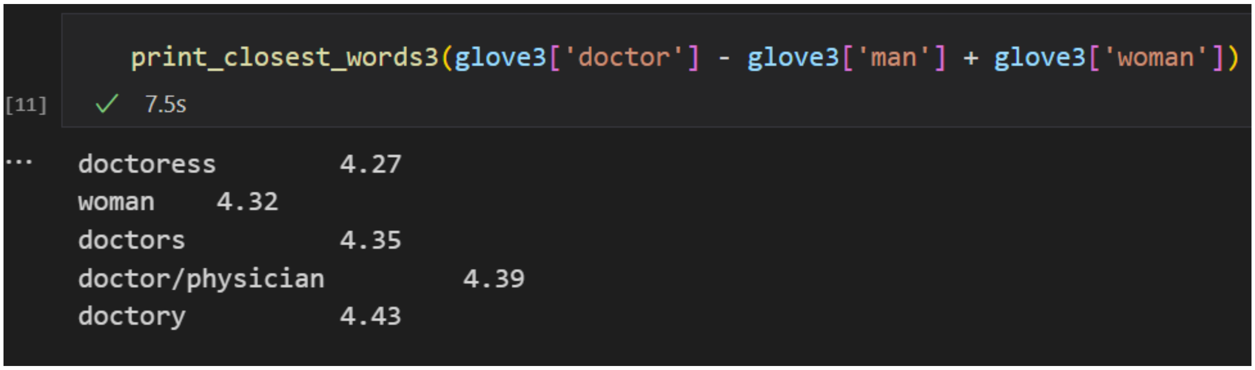
Yes, nearness changes. In summary, while larger vector sizes generally lead to better results due to their increased expressiveness and ability to capture finer linguistic nuances, there may be practical considerations such as computational resources that could influence the choice of vector dimensionality. But it will require larger test sample to verify this.

A1P1_6

State any changes that you see in the Bias section of the notebook.

Below are two examples

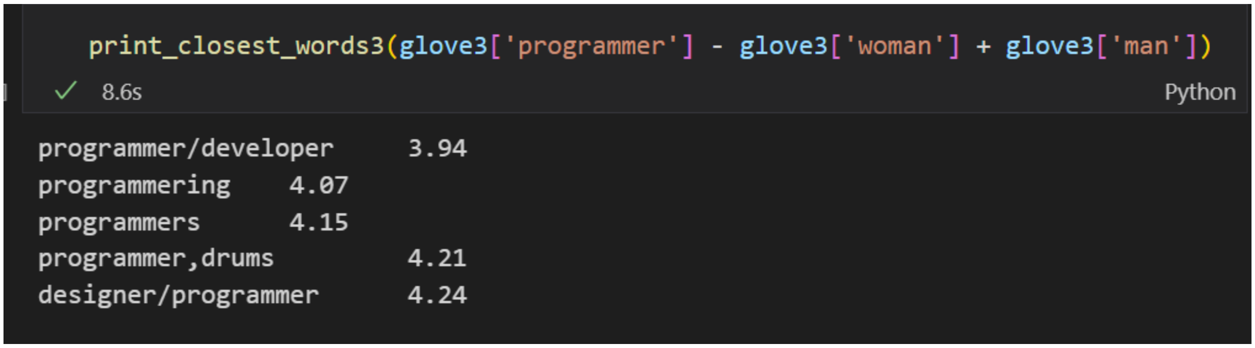
Ex1



```
print_closest_words3(glove3['doctor'] - glove3['man'] + glove3['woman'])
```

```
doctoress        4.27
woman    4.32
doctors          4.35
doctor/physician         4.39
doctory          4.43
```

Ex2



```
print_closest_words3(glove3['programmer'] - glove3['woman'] + glove3['man'])
```

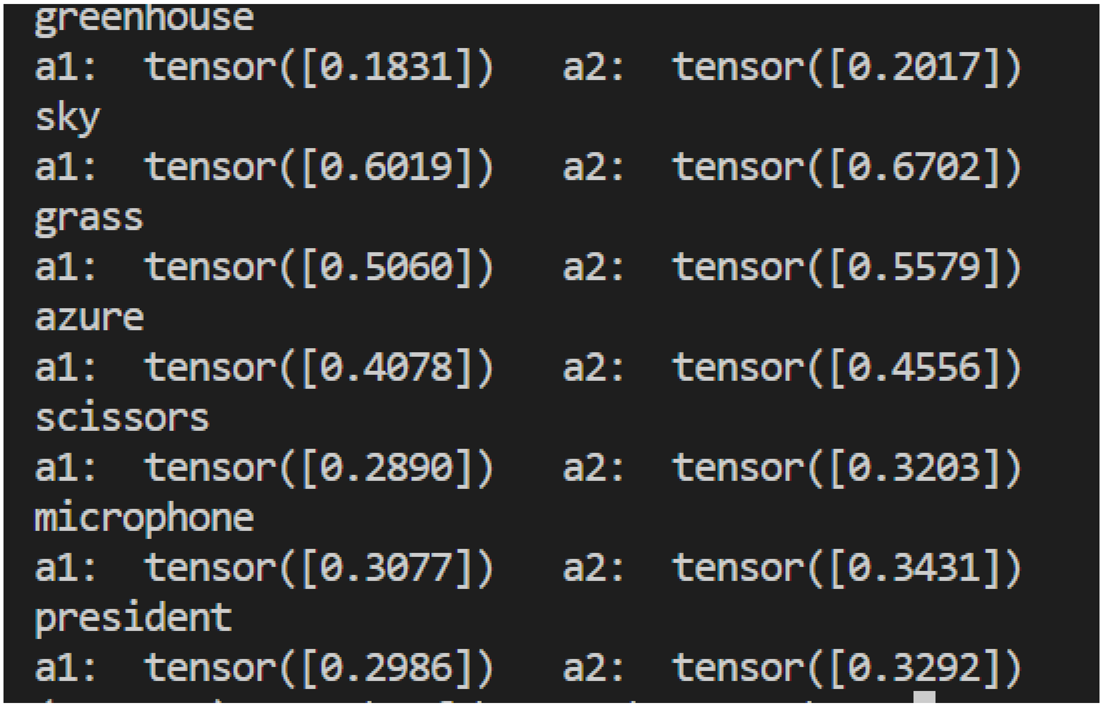
```
programmer/developer     3.94
programmering    4.07
programmers      4.15
programmer,drums         4.21
designer/programmer      4.24
```

The results generated from FastText regarding to the gender factor are much more accurate and reasonable.

A1P2_2

Do the results for each method make sense? Why or why not? What is the apparent difference between method 1 and 2?

Result:



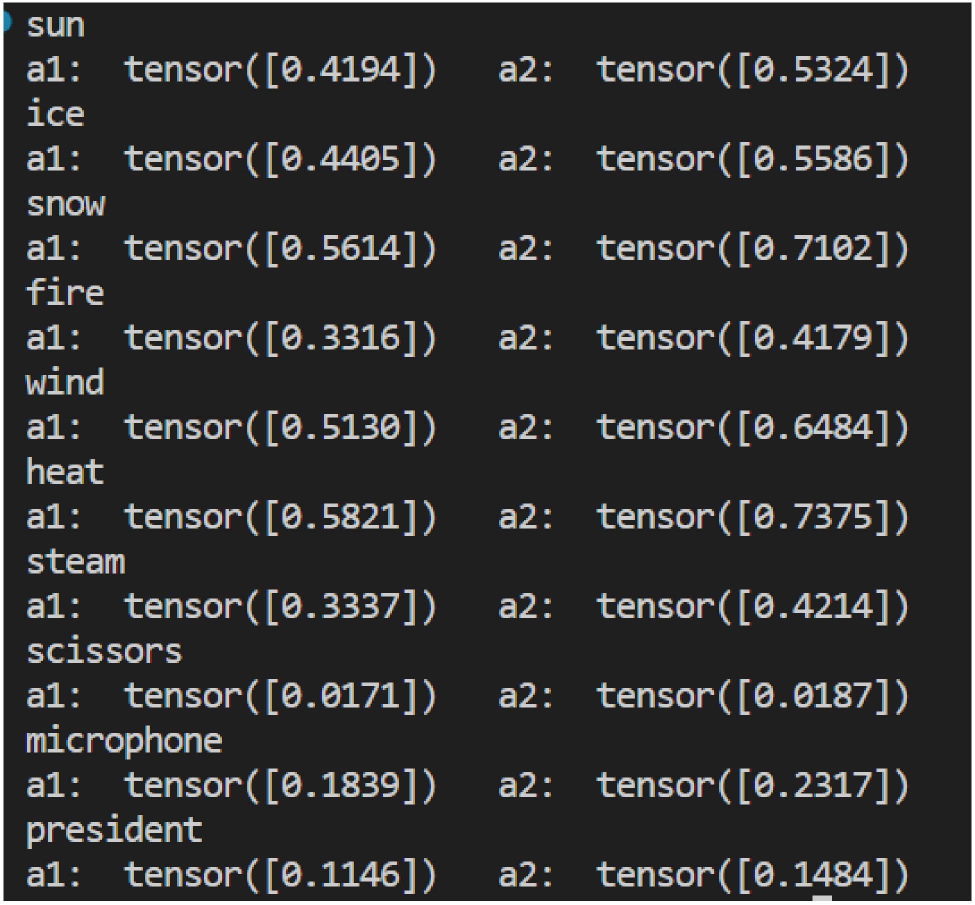
```
greenhouse
a1:  tensor([0.1831])   a2:  tensor([0.2017])
sky
a1:  tensor([0.6019])   a2:  tensor([0.6702])
grass
a1:  tensor([0.5060])   a2:  tensor([0.5579])
azure
a1:  tensor([0.4078])   a2:  tensor([0.4556])
scissors
a1:  tensor([0.2890])   a2:  tensor([0.3203])
microphone
a1:  tensor([0.3077])   a2:  tensor([0.3431])
president
a1:  tensor([0.2986])   a2:  tensor([0.3292])
```

The results make sense. Sky grass and azure which are most colourful objects. The apparent difference is method 2 is larger than method 1.

A1P2_3

Create a similar table for the meaning category temperature by defining your own set of category words, and test a set of 10 words that illustrate how well your category works as a way to determine how much temperature is “in” the words.

Cat = ['freezing', 'cold', 'chilly', 'brisk', 'cool', 'mild', 'warm', 'balmy', 'hot', 'sweltering', 'scorching']
test = ['sun', 'ice', 'snow', 'fire', 'wind', 'heat', 'steam', 'scissors', 'microphone', 'president']

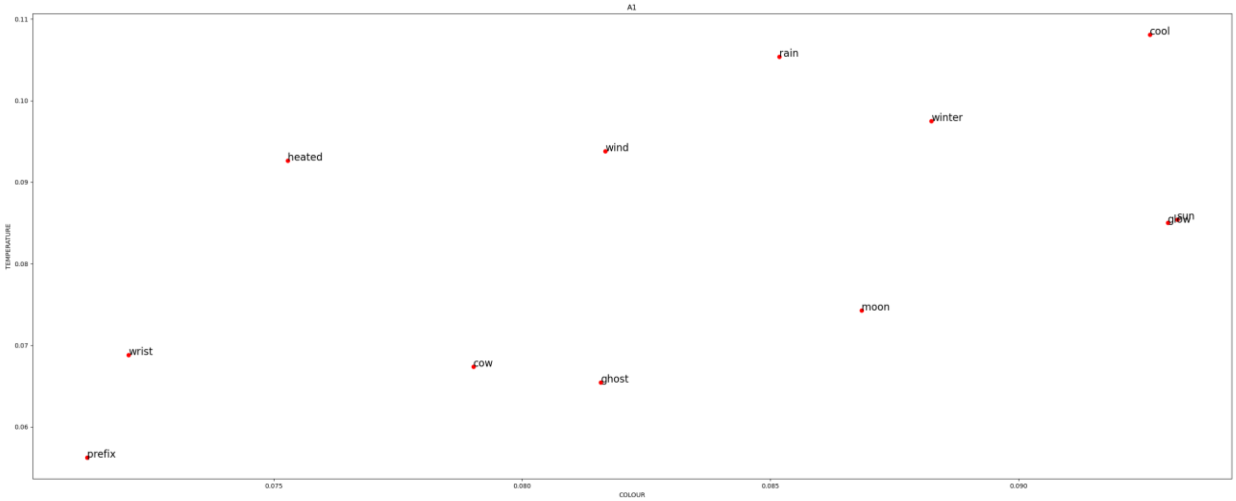


```
sun
a1:  tensor([0.4194])   a2:  tensor([0.5324])
ice
a1:  tensor([0.4405])   a2:  tensor([0.5586])
snow
a1:  tensor([0.5614])   a2:  tensor([0.7102])
fire
a1:  tensor([0.3316])   a2:  tensor([0.4179])
wind
a1:  tensor([0.5130])   a2:  tensor([0.6484])
heat
a1:  tensor([0.5821])   a2:  tensor([0.7375])
steam
a1:  tensor([0.3337])   a2:  tensor([0.4214])
scissors
a1:  tensor([0.0171])   a2:  tensor([0.0187])
microphone
a1:  tensor([0.1839])   a2:  tensor([0.2317])
president
a1:  tensor([0.1146])   a2:  tensor([0.1484])
```

The result are ok. The word more related to the temperature has a larger value.

A1P2_4

Plot each of the words in two dimensions (one for colour and one for temperature) using matplotlib. Do the words that are similar end up being plotted close together? Why or why not?



Most of the wrods similar meaning regarding to temperature and colour. For example. “sun” and “glow” is very close. “rain”, “wind”, “winter” aslo relatively close. Since they have a higher similarity in terms of color and temp, it should be closer in the graph.

A1P3_1

find three pairs of words that this corpus implies have similar or related meanings.

dog – cat; hold – rub; a – the

A1P3_2

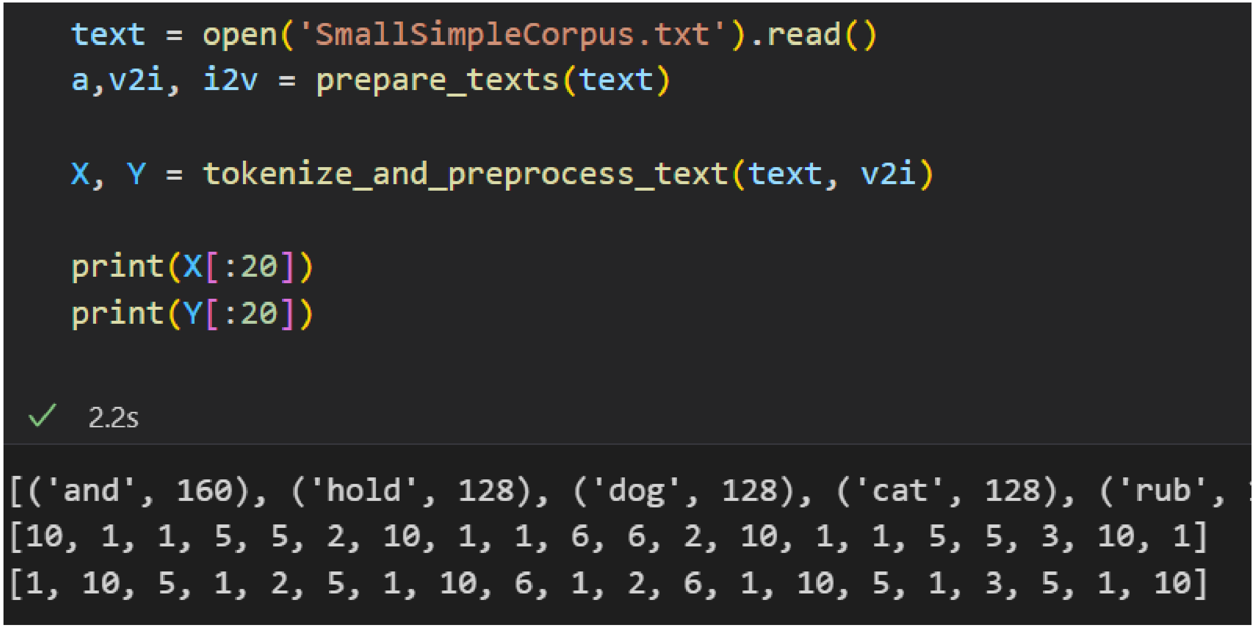
Check that the vocabulary size is 11. Which is the most frequent word in the corpus, and the least frequent word? What purpose do the v2i and i2v functions serve?

Result: [('and', 160), ('hold', 128), ('dog', 128), ('cat', 128), ('rub', 128), ('a', 104), ('the', 104), ('can', 104), ('she', 96), ('he', 96), ('I', 80)]

Vocabulary size is 11. The most frequent word is "and", the list frequent word is "I". The purpose of v2i and i2v is to covert between word and index (tokenizer) quickly by giving one of them.

A1P3_3

Test that your function works and show with examples of output (submitted) that it does.
First 20 sample pairs



```
text = open('SmallSimpleCorpus.txt').read()
a,v2i, i2v = prepare_texts(text)

X, Y = tokenize_and_preprocess_text(text, v2i)

print(X[:20])
print(Y[:20])
```

✓ 2.2s

```
[('and', 160), ('hold', 128), ('dog', 128), ('cat', 128), ('rub', 
[10, 1, 1, 5, 5, 2, 10, 1, 1, 6, 6, 2, 10, 1, 1, 5, 5, 3, 10, 1]
[1, 10, 5, 1, 2, 5, 1, 10, 6, 1, 2, 6, 1, 10, 5, 1, 3, 5, 1, 10]
```

A1P3_4

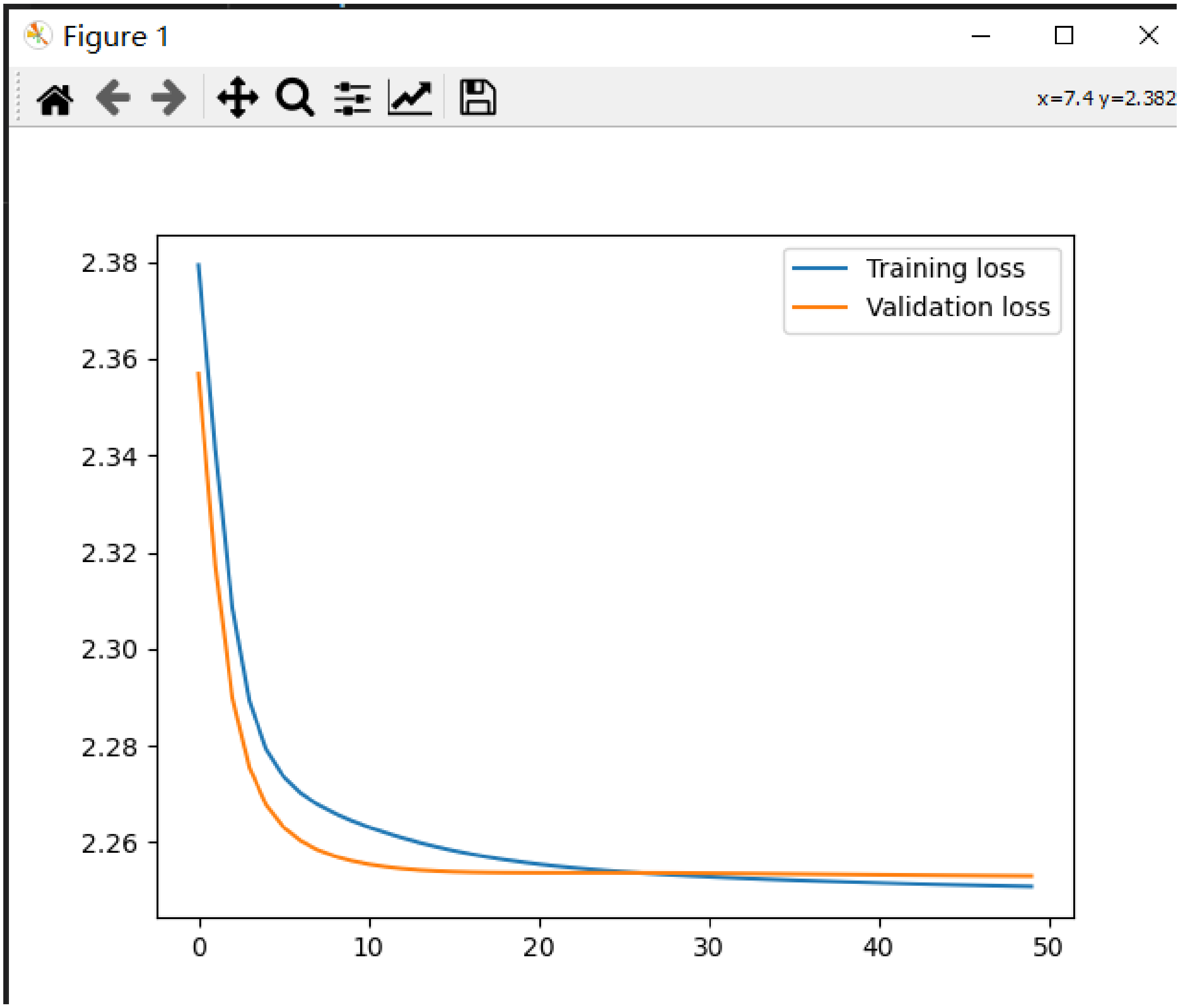
What is the total number of parameters in this model with an embedding size of 2 - counting all the weights and biases?

Assume we did not reuse the embedding parameters to the linear weight with no bias (bias =0), the total number of weights is 2*11 = 22

A1P3_5

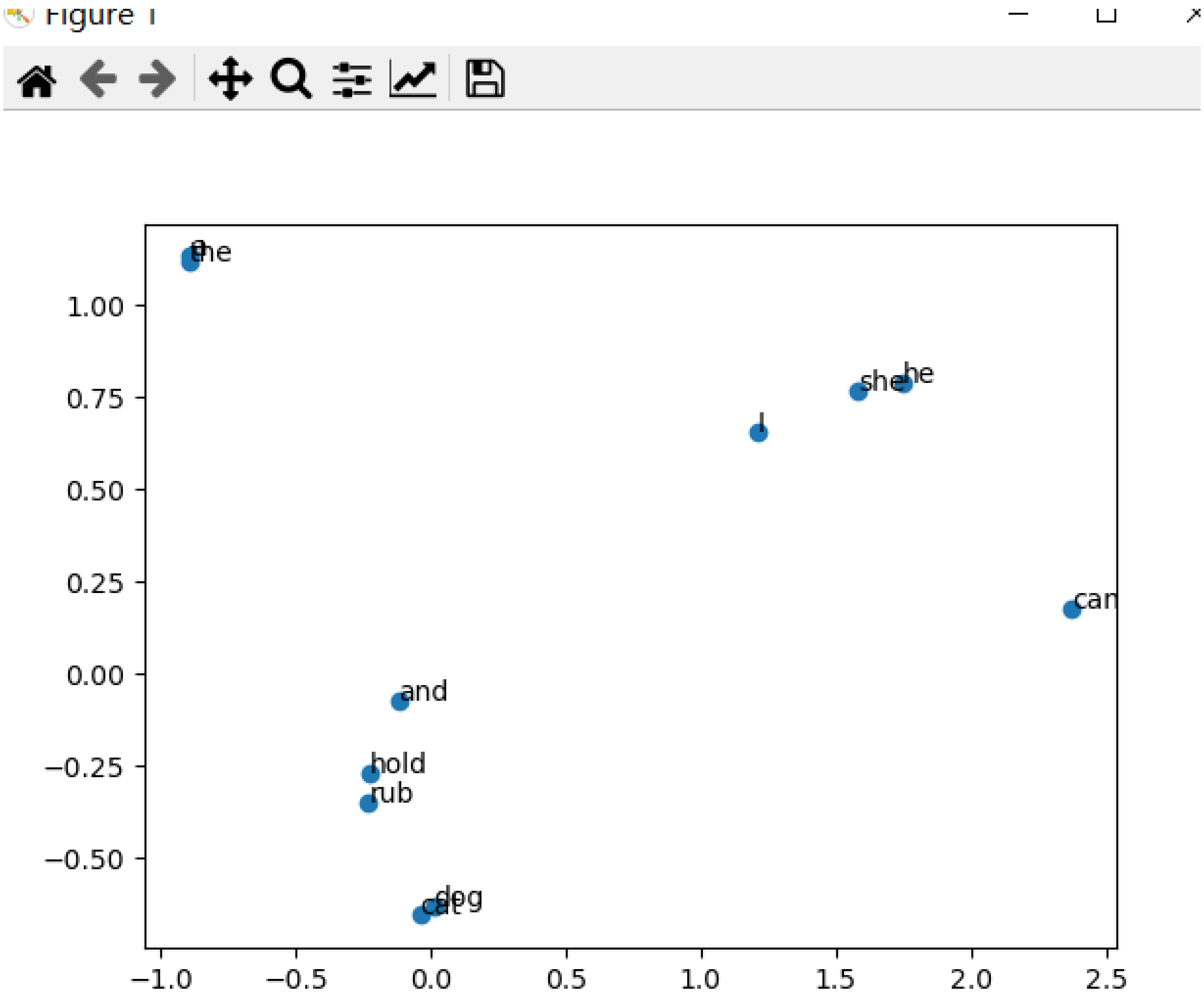
find a suitable learning rate, and report what that is. Show the training and validation curves (loss vs. Epoch), and comment on the apparent success (or lack thereof) that these curves suggest.

Learning rate 0.0008

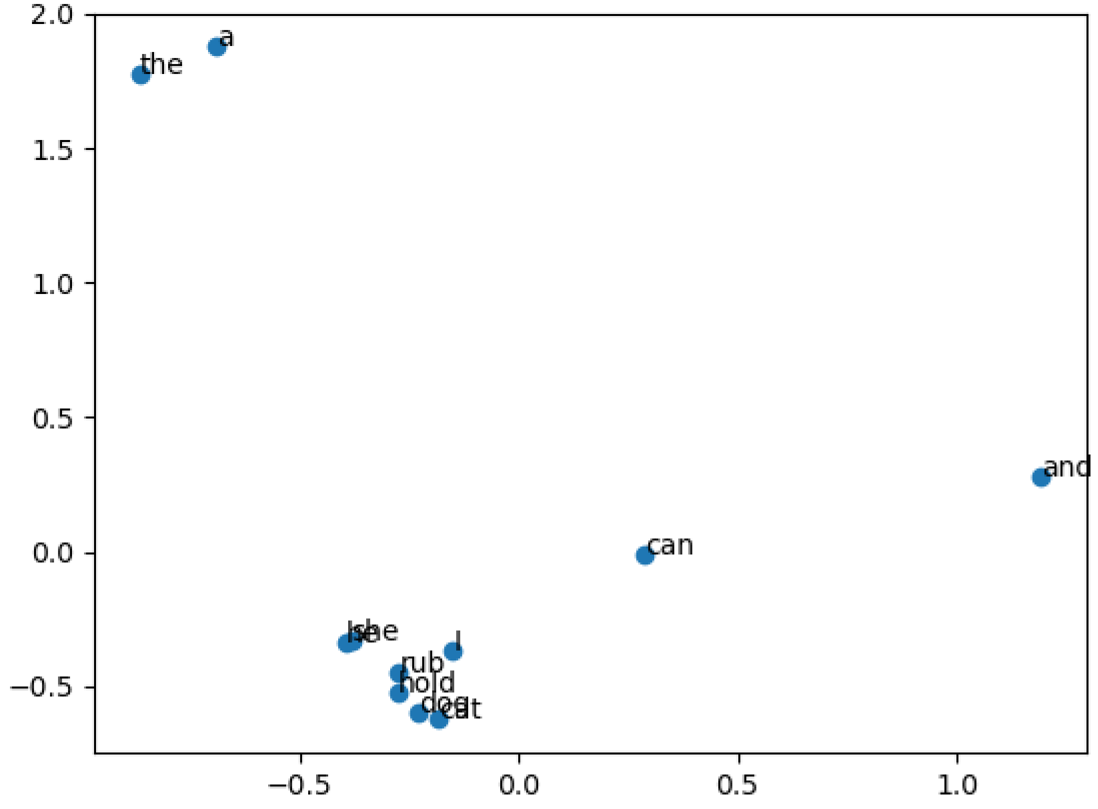


It might be something wrong with the training setting that the training curve does not converge.

A1P3_6



With window = 7



A1P4_1

Coin history. Past coin in ancient. Present coin in America

A1P4_2

Remove more symbol and numbers. Filter the vocabulary below the threshold frequency

A1P4_3

Number of lemmas: 62255

Number of frequent lemmas: 2568

Top 20 most frequent lemmas: [('the', 5048), ('of', 3438), ('be', 2283), ('and', 1943), ('in', 1588), ('to', 1379), ('a', 1225), ('for', 531), ('as', 518), ('by', 493), ('he', 483), ('with', 471), ('coin', 427), ('this', 388), ('on', 377), ('his', 368), ('which', 346), ('at', 334), ('it', 332), ('from', 326)]

“Coin”

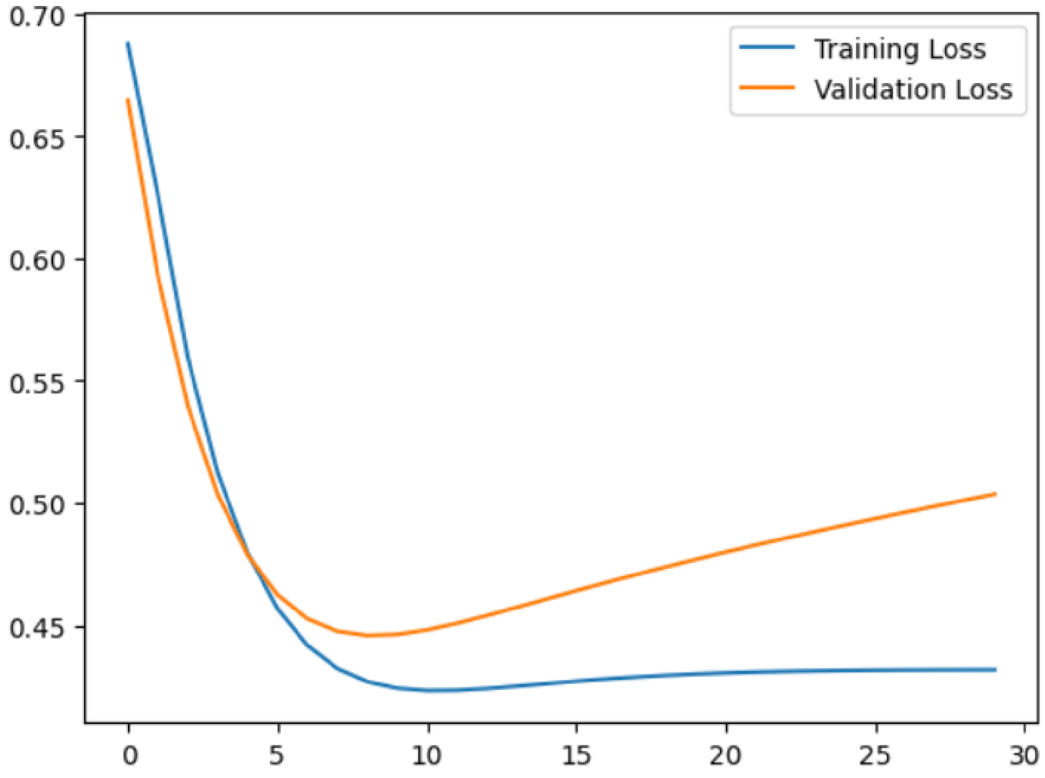
A1P4_4

Number of training samples: 231480

A1P4_5

Number of training samples: 191108

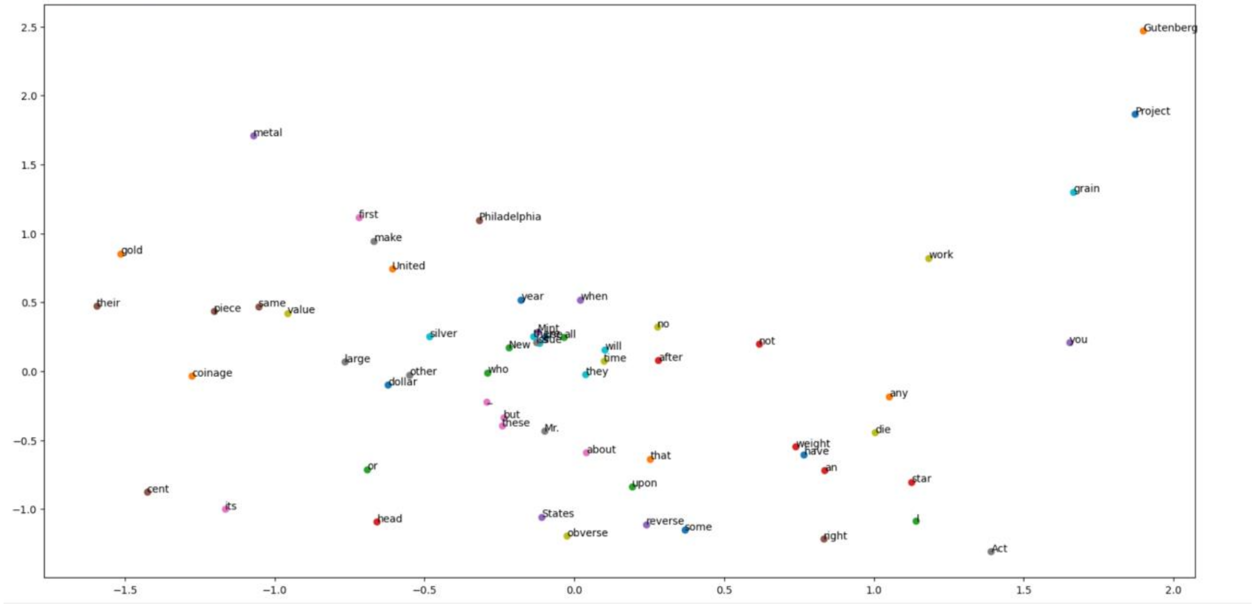
A1P4_7



```
<Figure size 640x480 with 0 Axes>
```

A1P4_8

Visualizing the 20 to 80 most frequent words



No-not close

When year

Gold – metal close

Philadelphia – Gutenberg far away