

Question 1.2

Provide a table that compares the 5-most cosine-similar words to the word 'dog', 'computer', alongside to the 10 closest words computed using Euclidean distance.

-----Cosine Similarity for word: dog-----		-----Cosine Similarity for word: computer-----	
cat	0.92	computers	0.92
dogs	0.85	software	0.88
horse	0.79	technology	0.85
puppy	0.78	electronic	0.81
pet	0.77	internet	0.81
-----Euclidean distance for word: dog-----		-----Euclidean distance for word: computer-----	
cat	1.88	computers	2.44
dogs	2.65	software	2.93
puppy	3.15	technology	3.19
rabbit	3.18	electronic	3.51
pet	3.23	computing	3.60
horse	3.25	devices	3.67
pig	3.39	hardware	3.68
pack	3.43	internet	3.69
cats	3.44	applications	3.69
bite	3.46	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. From the table, we can observe that the result words generated by cosine similarity are similar to the results that are generated from Euclidean distance.

We know that Euclidean distance measures the absolute distance between two word vectors, while Cosine similarity measures the difference of the degree of two word vectors' orientation.

When Euclidean distance reach the similar result with Cosine similarity, it means two vectors are close to each other (this reaches similar orientation and similar orientation)we can conclude that:

Dog and its similar words obtain the similar magnitude, orientation in vectors.

Computer and its similar words obtain the similar magnitude, orientation in vectors.

Question 1.3

Choose one of these relationships.

I choose present participle: think::thinking.

----- second word given: run-----		----- second word given: speak-----	
ran	2.99	speaks	3.31
started	3.23	learned	3.35
running	3.23	communicated	3.37
went	3.29	explaining	3.44
runs	3.30	listening	3.45
----- second word given: dance-----		----- second word given: eat-----	
dancing	3.12	eating	2.62
dances	3.32	ate	2.93
singing	3.54	eaten	3.06
improvisation	3.80	eats	3.45
musical	3.82	soup	3.73
----- second word given: listen-----		----- second word given: drink-----	
listen	2.33	drinks	2.74
listens	3.10	sodas	3.20
communicate	3.25	drinking	3.47
listened	3.33	soda	3.47
thoughts	3.36	bottled	3.49

second word given: draw	
draws	3.23
drawing	3.42
advances	3.50
drawn	3.59
hosts	3.67

second word given: play	
playing	2.99
plays	3.46
played	3.48
starting	3.56
game	3.57

second word given: write	
writing	2.86
translating	3.05
edit	3.40
compose	3.54
publish	3.57

second word given: walk	
walking	2.44
walks	2.91
stairs	3.32
stroll	3.35
paces	3.52

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

10 words: ["run", "dance", "listen", "speak", "eat", "drink", "draw", "play", "write", "walk"]

The quality of the generation is highly acceptable. There are Five words that shows its target present tense in the first place. Three words that contain the target present tense in top 5. And rest of two words fail to provide its present tense in top5 results, which are “listen” and “speak”. It is quite interesting to notice that “listening” appears in the top5 when generating results from “speak”. “Speak” can be heavily associated with “listening” in a context, so it is comprehensive to get “listening” from “speak.”

Question 1.4

Choose a context that you’re aware of 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.

If a word has multiple meanings, and different meanings fall into different word categories (nouns, verbs, or adjectives), it may take a bias towards one of the meanings.

I first tried to get the opposite word from “negative”. “Negative” is a word that widely used as a single meaning in single category: (means something bad, used as an adjective)

```
print_closest_words(glove["possible"] - glove["impossible"] + glove["negative"])
```

concern	4.05
concerns	4.14
positive	4.19
suggesting	4.30
response	4.32

From the result, we can somehow guarantee an opposite adjective can be generated.

Then I tried to get the opposite word from “fat”, which has multiple meanings: a natural substance in animal bodies when considered as a noun, OR describe the body shape when considered as an adjective.

```
print_closest_words(glove["possible"] - glove["impossible"] + glove["fat"])
```

milk	4.77
diet	4.80
blood	4.92
supplements	4.98
soft	5.05

We can observe that, it took bias to treat “fat” as a noun other than an adjective and fail to generate any adjective that has the opposite meaning.

Question 1.5

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?

We try to measure the Euclidean difference and cosine similarity between two opposite words: “Possible” and “impossible”:

```
-----d = 50-----  
-----Euclidean distance for word: possible and good-----  
tensor(4.3129)  
-----Cosine similarity for word: possible and good-----  
tensor([0.6478])
```

```
-----d = 300-----  
-----Euclidean distance for word: possible and impossible-----  
tensor(5.6155)  
-----Cosine similarity for word: possible and impossible-----  
tensor([0.4767])
```

Euclidean distance increases and cosine similarity decreases in d=300, states that these two words obtain larger distance compared to the result that we generate when d=50.

Does the ordering of nearness change?

We change the dimension to 300, and rerun Question 2:

d = 50

```
-----Cosine Similarity for word: dog-----  
cat      0.92  
dogs     0.85  
horse    0.79  
puppy    0.78  
pet       0.77  
-----Euclidean distance for word: dog-----  
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
```

d = 300

```
-----Cosine Similarity for word: dog-----  
dogs     0.79  
cat       0.68  
pet       0.63  
puppy     0.59  
hound     0.55  
-----Euclidean distance for word: dog-----  
dogs     4.36  
cat       5.20  
pet       5.70  
puppy     5.86  
hound     6.22  
pets      6.40  
animal    6.41  
-----  
canine    6.45  
cats      6.46  
6.43
```

When finding similar word for dog, in d = 300 we can observe that some words that are more related to “dog”(“dogs”, “pet”, “puppy”) become the nearest. While d = 50 still has some words such as “horse”, “pigs” that are another species than dogs. (both Euclidean distance and Cosine similarity applies to this conclusion)

d = 50

```
-----Cosine Similarity for word: computer-----  
computers 0.92  
software  0.88  
technology 0.85  
electronic 0.81  
internet  0.81  
-----Euclidean distance for word: computer-----  
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
```

d = 300

```
-----Cosine Similarity for word: computer-----  
computers 0.82  
software  0.73  
pc         0.62  
technology 0.62  
computing  0.62  
-----Euclidean distance for word: computer-----  
computers 4.18  
software  5.32  
technology 6.10  
laptop    6.15  
computing 6.20  
pc         6.22  
hardware  6.32  
internet  6.38  
instance  6.41  
programmer 6.47
```

It cannot be concluded that $d=300$ brings more related words for computer. As shown in $d = 50$, words generated are all computer related. My guess is computer related words are initially close to “computer” itself, unlike “dog” that might be surrounded by other similar animal words. Therefore, increase the dimension does not improve the result for “computer” much.

Is it clear that the larger size vectors give better results - why or why not?

From the above example about “dog” and “computer”, we can conclude that larger size vectors SOMETIMES give better results depends on the word that you input. If the word input is diverse and appear in various context, increasing dimension will improve results(such as dogs), but if the word is quite concrete and specific to a single concept(such as computer), we can probably obtain a high quality result through low dimension.

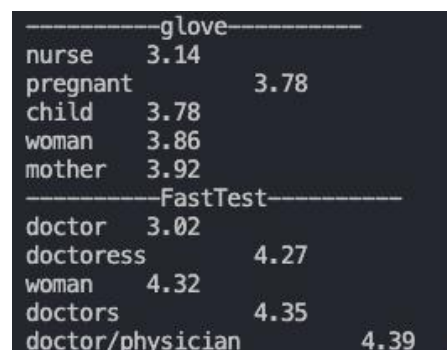
In addition, larger size vectors consume more calculation resources and drag the calculation performance. Therefore, we should choose the size of the vectors wisely based on the target words.

Question 1.6

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

We try the gender bias using FastText embeddings.

```
print_closest_words(glove['doctor'] - glove['woman'] + glove['man'])
```



The image shows a terminal window with two sections of output. The first section, titled 'glove', lists words and their similarity scores: nurse (3.14), pregnant (3.78), child (3.78), woman (3.86), and mother (3.92). The second section, titled 'FastTest', lists words and their similarity scores: doctor (3.02), doctress (4.27), woman (4.32), doctors (4.35), and doctor/physician (4.39).

Word	Score
nurse	3.14
pregnant	3.78
child	3.78
woman	3.86
mother	3.92

Word	Score
doctor	3.02
doctress	4.27
woman	4.32
doctors	4.35
doctor/physician	4.39

We can conclude that FastText has solved the bias by putting more neutral meanings towards doctor.

Question 2.2

Do the results for each method make sense? Why or why not? What is the apparent difference between method1 and method 2?

```
word: greenhouse: method1 result: tensor([0.1831]), method2 result: tensor([0.2017])
word: sky: method1 result: tensor([0.6019]), method2 result: tensor([0.6702])
word: grass: method1 result: tensor([0.5060]), method2 result: tensor([0.5579])
word: azure: method1 result: tensor([0.4078]), method2 result: tensor([0.4556])
word: scissors: method1 result: tensor([0.2890]), method2 result: tensor([0.3203])
word: microphone: method1 result: tensor([0.3077]), method2 result: tensor([0.3431])
word: president: method1 result: tensor([0.2986]), method2 result: tensor([0.3292])
```

The results for each method make sense.

“grass” may be more related to “green”, and “sky”, “azure” may be more related to “blue”.

Therefore, “grass”, “sky”, and “azure” obtain a higher result in method1 and method2 because they have high similarity with one color in the category.

We could also observe method2 result is always higher than method1 result. In method2 approach(we first average all the words in the category and calculate the similarity as a whole), compared with method1 approach (we first calculate the cosine similarity for each word in category and then average), it smooths the single relationship and ends up with a higher similarity.

Question 2.3

Comment on how well your approach worked.

```
category = ["hot", "cold", "boiling", "warm", "freezing"]
candidates = ["ice", "wind", "rain", "water", "body", "food", "cable", "scissors",
              "microphone", "president"]
```

```
word: ice: method1 result: tensor([0.6069]), method2 result: tensor([0.7120])
word: wind: method1 result: tensor([0.5808]), method2 result: tensor([0.6812])
word: rain: method1 result: tensor([0.6306]), method2 result: tensor([0.7395])
word: water: method1 result: tensor([0.6799]), method2 result: tensor([0.7971])
word: body: method1 result: tensor([0.3842]), method2 result: tensor([0.4507])
word: food: method1 result: tensor([0.5138]), method2 result: tensor([0.6016])
word: cable: method1 result: tensor([0.2388]), method2 result: tensor([0.2794])
word: scissors: method1 result: tensor([0.1499]), method2 result: tensor([0.1744])
word: microphone: method1 result: tensor([0.2207]), method2 result: tensor([0.2639])
word: president: method1 result: tensor([0.1838]), method2 result: tensor([0.2186])
```

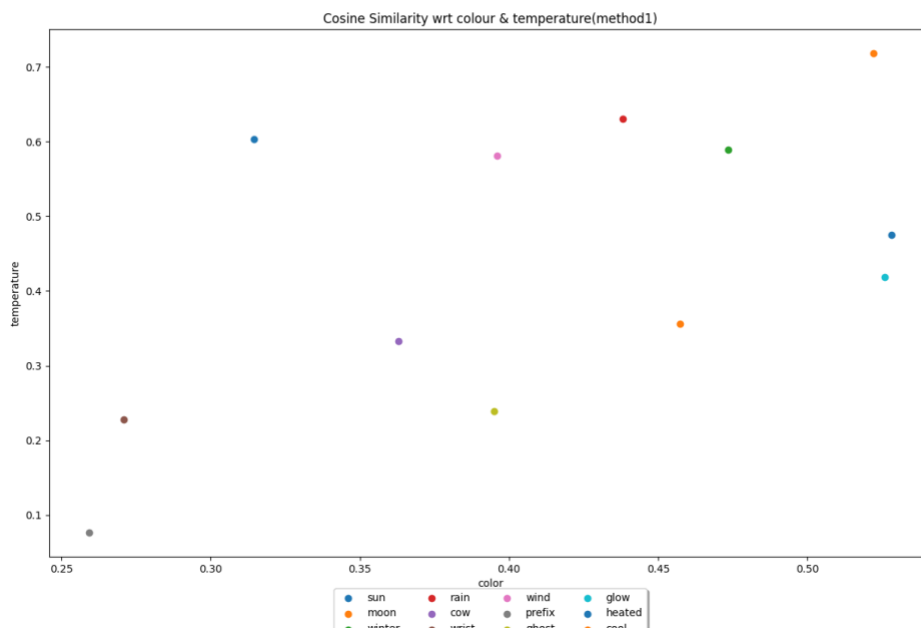
We can conclude that the conclusion in 2.2 still applies to my examples:

Method2 has higher result in general.

Temperature related words (ice, wind, rain, water, food, body) obtain higher result.

Question 2.4

Do the words that are similar end up being plotted close together? Why or why not?



We can observe the similar words end up being lotted close together: “sun” and “glow”, “rain”, “wind” and “winter”. Similar words share the common “amount” of similarity with words that are associated with color and temperate, which indicates similar amount in x-axis and y-axis, which further means similar positions.

Question 3.1

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

hold & rub

I & he

Cat & dog

Question 3.2

Check that the vocabulary size is 11. Which is the most frequent word in the corpus, and the least frequent word?

```
🔍 vocabulary size: 11  
[('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 as tested above.

By printing the vocab dictionary(word: frequency), the most frequent word in corpus is “and”, and the least frequent word is “I”.

What purpose do the v2i and i2v functions serve?

```
word->index dictionary:  
{'and': 0, 'hold': 1, 'dog': 2, 'cat': 3, 'rub': 4, 'a': 5, 'the': 6, 'can': 7, 'she': 8, 'he': 9, 'I': 10}  
index->word dictionary:  
{0: 'and', 1: 'hold', 2: 'dog', 3: 'cat', 4: 'rub', 5: 'a', 6: 'the', 7: 'can', 8: 'she', 9: 'he', 10: 'I'}
```

The v2i and i2v serve as the function to create word->index and index->word dictionary. We may use index to represent each different word during training so that it is easy to program, and we can use i2v at the end where we transfer back to word format to provide prediction result.

Question 3.3

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

```
🔍 first 6 items in X:  
[10, 1, 1, 5, 5, 2]  
first 6 items in Y:  
[1, 10, 5, 1, 2, 5]  
check them in pair format  
( 'I', 'hold' )  
( 'hold', 'I' )  
( 'hold', 'a' )  
( 'a', 'hold' )  
( 'a', 'dog' )  
( 'dog', 'a' )
```

The function will store all target words (index)in X, the corresponding context words(index) in Y. The pairs are only selected within the sentence.

For example, the first sentence: “I hold a dog.” will only obtain 6 valid pairs in window size = 3, as shown in the screenshot above.

Question 3.4

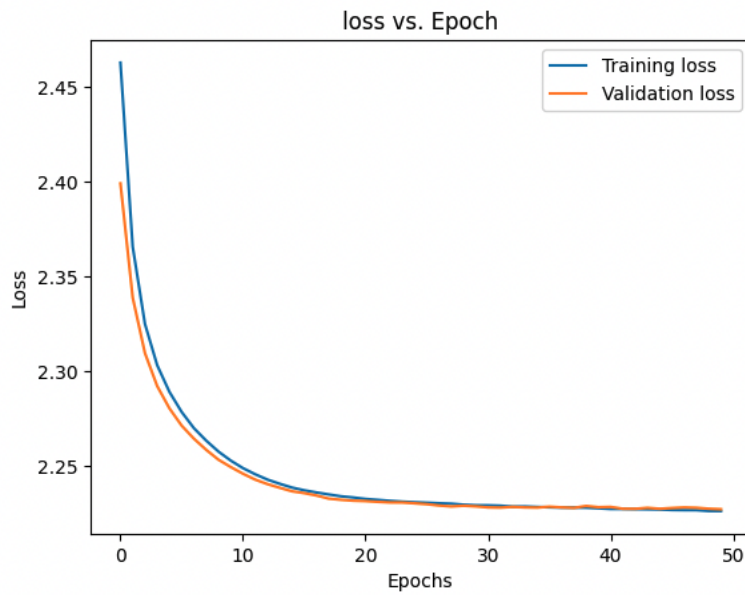
What is the total number of parameters in this model with an embedding size of 2-counting all the weights and biases?

The first “node” in model will contain $w[0][1]$, $w[0][0]$, and $\text{bias}[0]$, the last “node” will contain $w[10][0]$, $w[10][1]$, and $\text{bias}[10]$. Therefore, we have $11 \times 2 = 22$ weights, and 11 bias in total in this model.

Question 3.5

Find a suitable learning rate, and report what that is.

I found a suitable learning rate = 0.001.

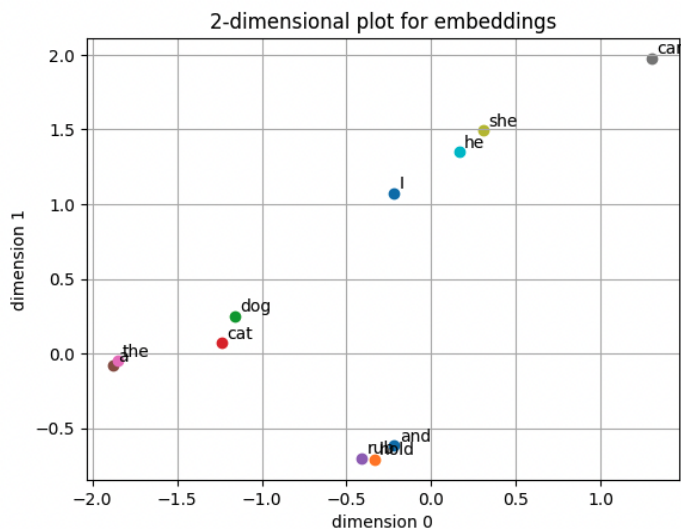


Show the training and validation curves (loss vs. Epoch), and comment on the apparent success (or lack thereof) that these curves suggest.

The curve above shows that the training loss and validation loss quickly decrease, and both converge at a low value of loss ≈ 2.23 . Additionally, the gap between training and validation is quite small, indicating the model is not overfitting.

Question 3.6

Display each of the embeddings in a 2-dimensional plot using Matplotlib.



Do the results make sense, and confirm your choices from part 1 of this Section?

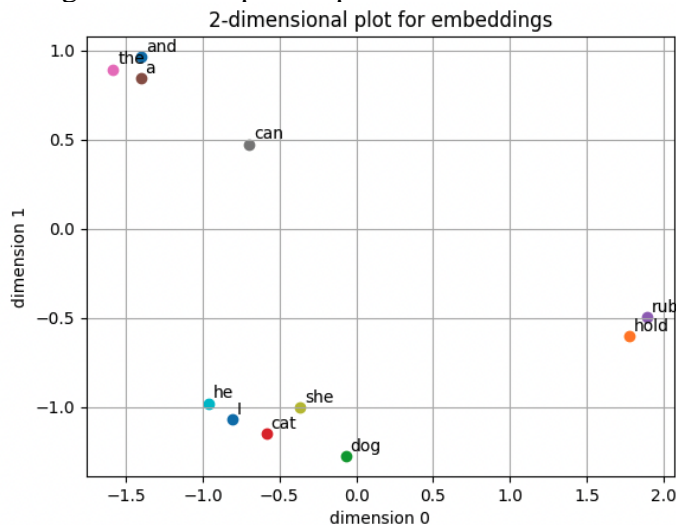
From the plot above, we can observe that similar words have similar word embeddings, for example: “a” & “the”, “I”&“he”&“she”, “rub”&”hold”, “cat”&”dog”. This confirms with my choices in part 1.

Notice that “and” is also somehow close to “rub”&”hold”. SmallSimpleCorpus only contains brief sentences (5-7 words per sentence), “and” will always be marked as a label with “rub” & “hold” since we are keeping window=5.

What would happen when the window size is too large? At what value would window become too large for this corpus?

I tried to train the network using window = 7, as shown in below plot, “cat”&“dog” starts to near “I”, “she”, & “he”. We may label the context word too far from the original target word, which misleads the prediction and lower the accuracy in result.

The largest sentence in SmallSimpleCorpus only contains 7 words, and we only mark the label within one single sentence. Therefore, there is no point to increase the window size larger than 7. In conclusion, I consider window=7 or any window size larger than 7 as a window size that is too large for this corpus scope.



Question 3.7

Run the sequence twice-and observe whether the results are identical or not.

The results are not identical.

Then set the random seeds, verify and confirm this in your report that the results are always the same every time you run your code.

Once I set the random seeds, results in every run became identical.

Question 4.1

Give a 3 sentence summary of the subject of the document.

The document talks about the money from past to present. It mentions the history of the coin. It also illustrates the history of US Mint.

Question 4.2

What are the functional differences between this code and that of the same function in Section 3?

In nlp lemmatize process, it not only removes punctuation and space, but also removes symbol, number, unclassified words, and special characters.

We still process vocabulary in (word, occurrence) but drop the words that has frequency less than `<min_frequency>`. For those words dropped, we replace the content in lemma with “<oov>”, and register “<oov>” in w2i and i2w.

Question 4.3

There are in total 62255 words in the text, 7508 words (if we count each word once).

The size of filtered vocabulary is: 2568.

20 most frequent words:

```
[('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)]
```

Of those top 20 most frequent words, which one(s) are unique to the subject of this particular text?

Coin is unique compared with other frequent words. Other frequent words are the words that are frequently used in the sentence. But coin is frequently used specifically because of the context of the article.

Question 4.4

How many total examples were created?

By setting window size = 5, there are in total 455080 examples (the length of X, Y and T) being created.

To understand what each output represents.

For the first sentence in LargerCorpus: “MONEY OF THE PAST AND PRESENT.”

```
['money', 'money', 'money', 'money']
```

```
['of', 'the', 'shield', 'MINT']
```

```
[1, 1, -1, -1]
```

We can observe, money has “of” and “the” as context words with labels 1, and randomly generated negative words “shield”, “MINT” with labels -1.

Question 4.5

State how many examples remain for the corpus using this reduction.

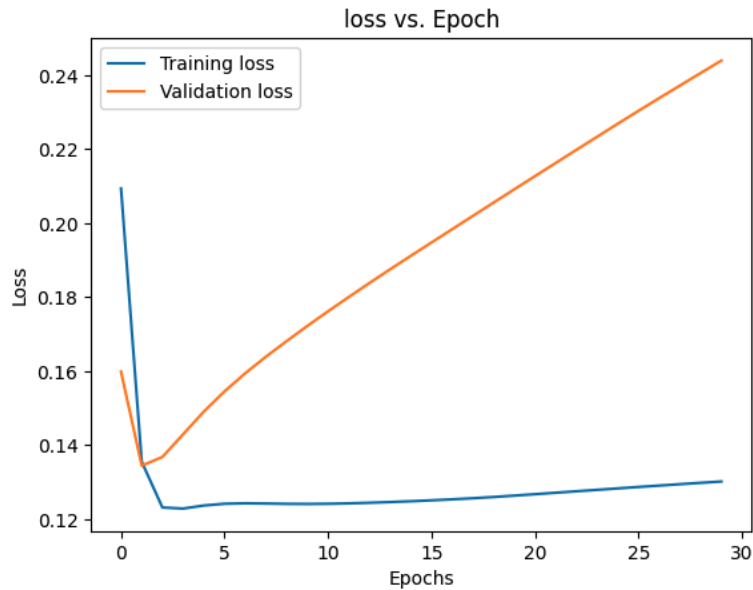
Filtered out the words that has frequency less than 3. After reduction, there are 408032 examples remain.

Question 4.7

Using the default Adam optimizer, find a suitable learning rate, and report what that is.

Show the training and validation curves vs Epoch, and comment on the apparent success (or larch thereof) that these curves suggest.

The learning rate chosen in 0.001. I observe the training loss and validation loss drops at first 5 epochs, then validation starts to increase, this indicates the model starts to overfit. My guess is embedding size = 8 lets the model “learn fast”, if we train in 30-epoch loop, the model will behave overfit.



Question 4.8

Comment on how well the embeddings worked, finding two examples each of embeddings that appear correctly placed in the plot, and two examples where they are not.

Since it's a bit hard to recognize each word on the plot from 20th frequent word to 80th frequent word, I change the function to only visualize the 20 to 55 most frequent words.

As shown in the plot below:

“on”, “that”, “to” appear together, “from”, “this” appear together.

“his”, “have” mistakenly clustered together, “he”, “of” mistakenly clustered together.

Visualizing the 20 to 55 most frequent words

