ECE1786 - Assignment 1 Report

Sepehr Ahmadi - 1010550360

September 2023

Section 1

Question 1.3:

Report the relationship you have chosen for this section and comment on quality of results of your 10 generated examples.

Answer:

The relationship I have chosen is **Nationality Adjective**. By adding the vector of the word *"citizen"* to a nation's name, the closest word vectors to the new vector seem to be the nationality adjective of the word. However this might not be true for every case; for example, for the nation **"Brazil"** we have *"mexico"*, *"spain"* and *"republic"* closer to the new vector opposed to the actual nationality adjective, which is **brazilian**. This shows that this analogy-like relationship established with adding vectors to each other is fallible.

Question 1.4

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.

Answer:

One bias I have observed in the glove word vectors involves the words *underweight* and *overweight*; such that for the word *"dancer"*, the vector $dancer + underweight - overweight$ is closest to words *ballerina, go-go, cabaret* and *choreographer*. However, the vector $dancer + overweight - underweight$ is closest to words *performer* and *entertainer*, which suggest different types of profession. This is caused by the inherent bias that has existed in the training data.

Question 1.5

Change the the embedding dimension (also called the vector size) from 50 to 300 and re-run the notebook including the new cosine similarity function from part 2 above. (a) How does the euclidean difference change between the various words in the notebook when switching from d=50 to d=300? (b) How does the cosine similarity change? (c) Does the ordering of nearness change? (d) Is it clear that the larger size vectors give better results - why or why not?

Answer:

(a) when we raise the dimension of our word embedding, the euclidean distance between every word **increases** proportionally. This is expected since we are dealing with a much larger vector space with added dimensions.

(b) The cosine similarity between words **decrease** with increased dimensions since the vector space is now much larger

(c) Yes, the ordering of nearness in most cases does change and the closest words to a word vector are rearranged.

(d) In this case, it is true that raising the dimensionality of word vectors from 60 to 300 is providing us with more accurate results; However, further increase in dimensionality (compared to the size of training data) may not prove as fruitful and may have detrimental effect on whatever model we may be running on our vector space, since optimizing word vectors and converging them may become problematic. in a exorbitantly large vector space. So it is better to find the right balance between data amount/variety and representation space size.

Question 1.6

Modify the notebook to use the FastText embeddings. State any changes that you see in the Bias section of the notebook.

Answer:

Both GloVe and FastText are techniques for learning word embeddings from text data, and neither of them explicitly aims to reduce biases in word vectors; However, for the examples of bias in word vectors described in the notebook, FastText performs better and displays much less bias regarding gender and weight in word vectors compared to the gloVe word embedding.

Section 2

Question 2.2

Compute the similarity (using both methods (a) and (b) above) for each of these words: “greenhouse”, “sky”, “grass”, “azure”, “scissors”, “microphone”, “president” and present them in a table. Do the results for each method make sense? Why or why not? What is the apparent difference between method 1 and 2?

Answer:

Word Similarities to Category ”Colour”		
Word	Method (a)	Method (b)
Greenhouse	0.18	0.20
Sky	0.60	0.67
Grass	0.51	0.56
Azure	0.41	0.46
Scissors	0.29	0.32
Microphone	0.31	0.34
President	0.30	0.33

For the most part, the results do seem to make sense; We can expect ”Sky” and ”Grass” to be most similar to the ”colour” category, and it can be seen in the table that they have a higher score. And words like ”Scissors”, ”Microphone” and ”President” do not have a discernible connection with colour, so the lower score is suitable; However, it is unexpected to see the word ”Greenhouse” score the lowest in this metric, since this word heavily suggests the color green. Also for ”Azure”, the mediocre score of 0.41 in the method (a) was also unpredicted, since it is primarily a color. However, one can surmise that ”Azure” is not a common color and is mostly familiar to people as a cloud platform.

The most apparent difference between the first and second method is that for a given word, the cosine similarity is always higher for the method (b) compared to method (a).

Question 2.3

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. You should explore different choices and try to make this succeed as much as possible. Comment on how well your approach worked.

Word Similarities to Category "Temperature"		
Word	Method (a)	Method (b)
Frozen	0.49	0.56
Sun	0.54	0.61
Sea	0.50	0.56
Airplane	0.29	0.33
Car	0.31	0.34
Burn	0.51	0.58
Water	0.68	0.77
Gold	0.35	0.40
Oven	0.50	0.58
Pillow	0.28	0.32

Answer:

For the category "Temperature", I have chosen the words:
$\{temperature, heat, hot, warm, cool, cold\}$ to define this category. For the most part, this arrangement in overall does well to separate words by their relation to temperature. "Frozen", "Sun", "Sea", "Oven", "Burn" and "Water" all have a high similarity score and they all represent the influence of temperature in these words. "Water" is by far the most similar word to the category, which can be rationalized since it can inhabit all the words in the category, as opposed to more intense words such as "Sun" and "Burn" which are more aligned with certain words of the category such as "hot" and "warm".
Additionally, words such as "Airplane", "Car", "Gold" and "Pillow" have scored lower since their relation to "Temperature" is much more limited.

Question 2.4

Use these two categories (colour & temperature) to create a new word vector (of dimension 2) for each of the words given in Table 1, in the following way: for each word, take its (colour, temperature) cosine similarity numbers (try both methods and see which works better), and apply the softmax function to convert those numbers into a probabilities. 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?

Answer:

Certain words do seem to cluster together. Words like "heated" and "cool" are aligned close together and they are both temperature adjectives. "wind" and "rain" are both natural occurrences and have appeared clustered together. Also "Sun" and "Glow" appear close to each other. However, many words such as "sun" and "moon" are located far from each other and some unrelated words are very close together; This is happening due to dimensionality reduction of our data (word vectors). By transforming 50 dimension words to 2 dimensional

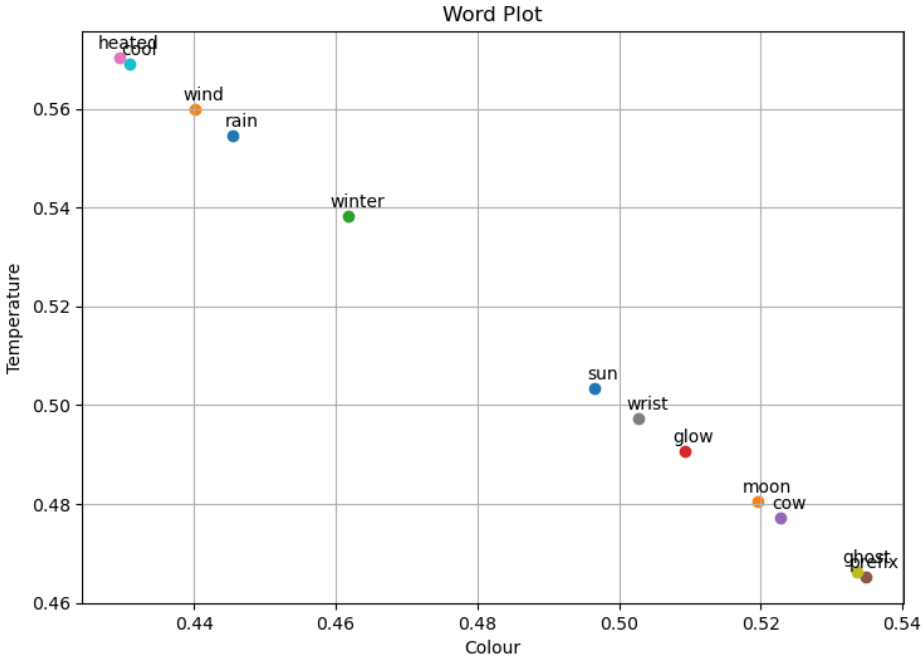

Figure 1: Word Vectors according to 2 categories

words, more than 90% of the word vector data is lost. This introduces noise to our dataset and causes poor alignment in our word vector plot. This problem can be mitigated by introducing finer measures to categorize our word data (i.e. using PCA) and represent them in a less lossy fashion.

Section 3

Question 3.1

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

Answer:

$(he, I), (she, I), (rub, hold), (a, the)$

Question 3.2

Review the code of `prepare_texts` to make sure you understand what it is doing. Write the code to read the corpus `SmallSimpleCorpus.txt`, and run the `prepare_texts` on it to return the text (lemmas) that will be used next. 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?

Answer:

The most frequent word is *"and"* with 160 occurrences and the least frequent is *"I"* with 80 occurrences.
The 2 dictionaries serve as a word embedding which allows us to compute the word from embedding and vice versa. It also ranks the vocabulary used in the training set based on frequency.

Question 3.3

You must generate all training examples across all words in the corpus within a window of size window. Test that your function works, and show with examples of output (submitted) that it does.

Answer:

The file is submitted and here is a screenshot of the output:

In [34]:

```
with open("SmallSimpleCorpus.txt") as f:
    corpus = f.read()

def generate_training_set(corpus, window=3):
    training_set = []
    lemmas, v2i, i2v = prepare_texts(corpus)
    max_word_dis = (window - 1)//2
    dis = np.arange(window) - max_word_dis # create word range from window size
    dis = np.delete(dis, np.where(dis == 0)) # avoid self from pairing with self
    for i,lemma in enumerate(lemmas):
        for d in dis:
            if (i+d>=0 and i+d<len(lemmas)):
                training_set.append((lemma, lemmas[i+d]))
    return training_set
generate_training_set(corpus)
```

Out[34]:

```
[('I', 'hold'),
 ('hold', 'I'),
 ('hold', 'a'),
 ('a', 'hold'),
 ('a', 'dog'),
 ('dog', 'a'),
 ('dog', 'I'),
 ('I', 'dog'),
 ('I', 'hold'),
 ('hold', 'I'),
 ('hold', 'the'),
 ('the', 'hold'),
 ('the', 'dog'),
 ('dog', 'the'),
 ('dog', 'I'),
 ('I', 'dog'),
 ('I', 'hold'),
 ('hold', 'I'),
 ('hold', 'a'),
```

Figure 2: Generated Word pairs

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?

Answer:

The total number is $11 * 2 * 2 = 44$ since we have 2 as an embedding size, 11 words in the vocabulary and bias for our layer.

Question 3.5

Using the default Adam optimizer, find a suitable learning rate, and report what that is. Show the training and validation curves (loss vs. Epoch), and comment on the ap- parent success (or lack thereof) that these curves suggest.

Answer:

I chose $learning\_rate = 0.001$ for my model which seemed to have better results. The graph suggests a successful training model since the loss is acceptable in the final epochs and the validation and training curves have converged so there is no sign of underfitting or overfitting.

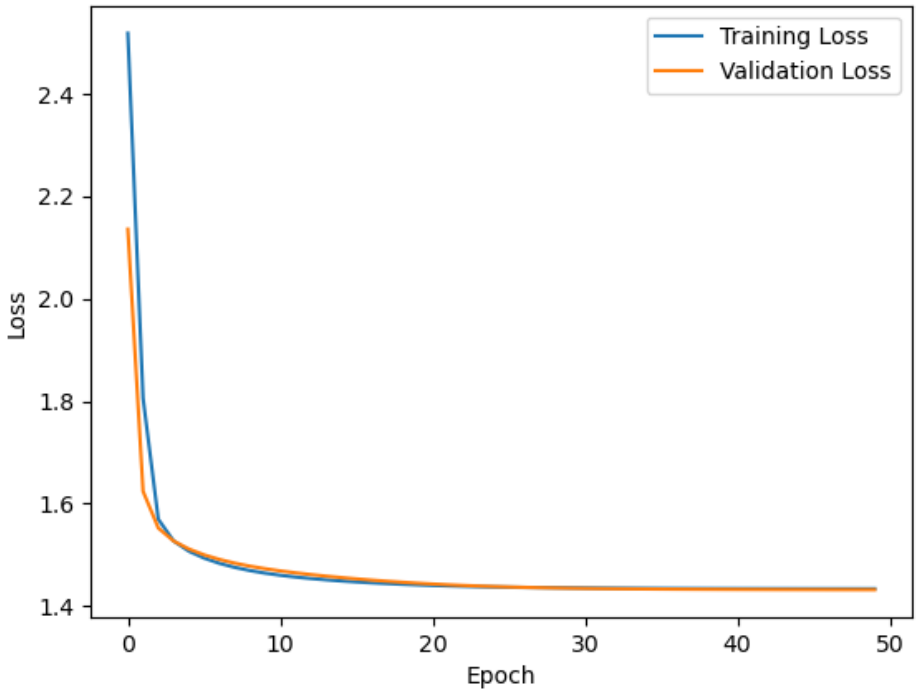

Figure 3: SkipGram Loss Chart

Question 3.6

For your best learning rate, display each of the embeddings in a 2-dimensional plot using Matplotlib. Display both a point for each word, and the word itself. Submit this plot, and answer this question: Do the results make sense, and confirm your choices from part 1 of this Section? What would happen when the window size is too large? At what value would window become too large for this corpus?

Answer:

For the most part, they do make sense; "dog" and "cat", "she" and "he", "rub" and "hold", "the" and "a" all appear together as they should.
If the window size is too large, each word will be paired with almost all other words in the sentence, which reduces the quality of the model. In this case, window spaces bigger than 5 is too large a window space.

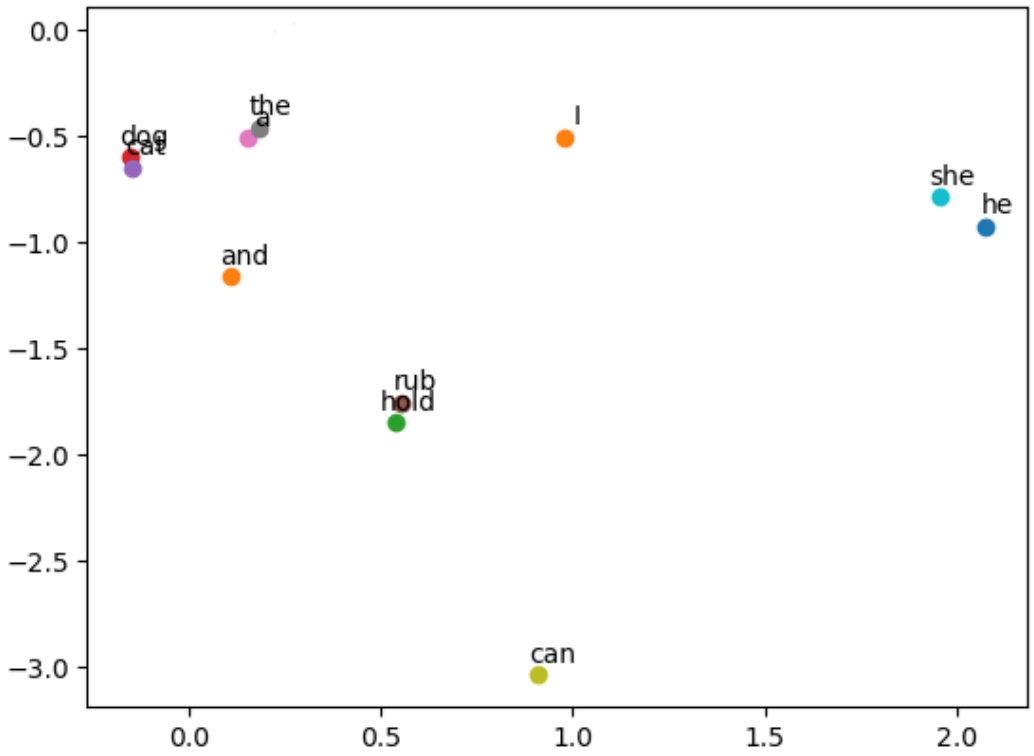

Figure 4: Word Embeddings Chart

Section 4

Question 4.1

Take a quick look through LargerCorpus.txt to get a sense of what it is about. Give a 3 sentence summary of the subject of the document.

Answer:

History of coin from the past until now,
History of currency and use of coin.
The economical journal of the Mints in America.

Question 4.2

The `prepare_texts` function in the starter code is a more advanced version of that same function given in Section 3. Read through it and make sure you

understand what it does. What are the functional differences between this code and that of the same function in Section 3?

Answer:

The function is designed to process more complex inputs that may include various symbols and numbers. As part of its functionality, the function also creates a vocabulary from the input data. Additionally, it filters out words that are not present in the vocabulary and replaces them with the placeholder "oov ."

Question 4.3

Determine the number of words in the text, and the size of the filtered vocabulary, and the most frequent 20 words in the filtered vocabulary, and report those. Of those top 20 most frequent words, which one(s) are unique to the subject of this particular text?

Answer:

Words in the text: 69267 words
Filtered vocabulary size: 7359 different words
Most frequent 20 words in the filtered vocabulary:

$('the', 5047), ('of', 3438), ('be', 2283), ('and', 1943), ('in', 1588), ('to', 1379), ('a', 1226), ('for', 531), ('as', 518), ('by', 493), ('he', 483), ('with', 471), ('coin', 417), ('this', 387), ('on', 377), ('his', 368), ('which', 346), ('at', 334), ('it', 332), ('from', 326)$

Word unique to the subject of this particular text is **"coin"**.

Question 4.4

How many total examples were created?

Answer:

453632

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 lack thereof) that these curves suggest.

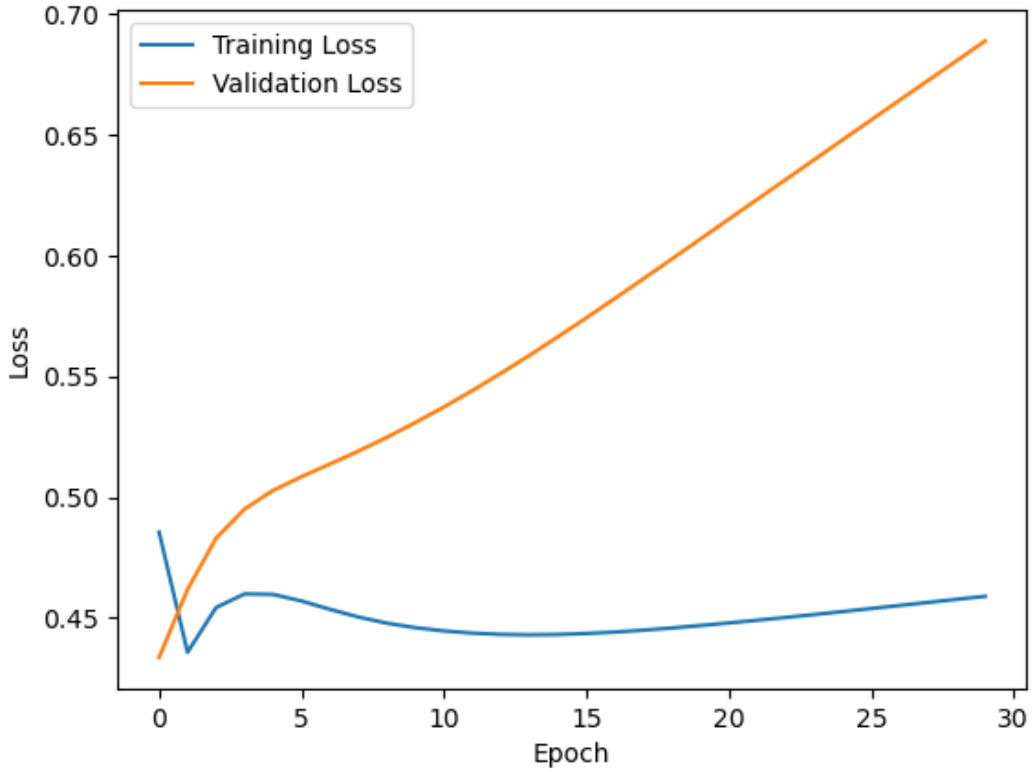

Figure 5: SGNS Loss Chart

Answer:

The training leaves much to desire; the increase of validation loss signifies overfitting. the learning rate assinged to the model is 0.0005. And as for negative sampling, I have used the `BCEWithLogits` loss function.

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.

Answer:

The embedding is working in a mediocre way, for example, "which", "from" and "for" appear together, "silver" and "coin" are also close together; whereas "their" and "use", and "weight" and "I" should not be clustered together in this context, but in this embedding they are.

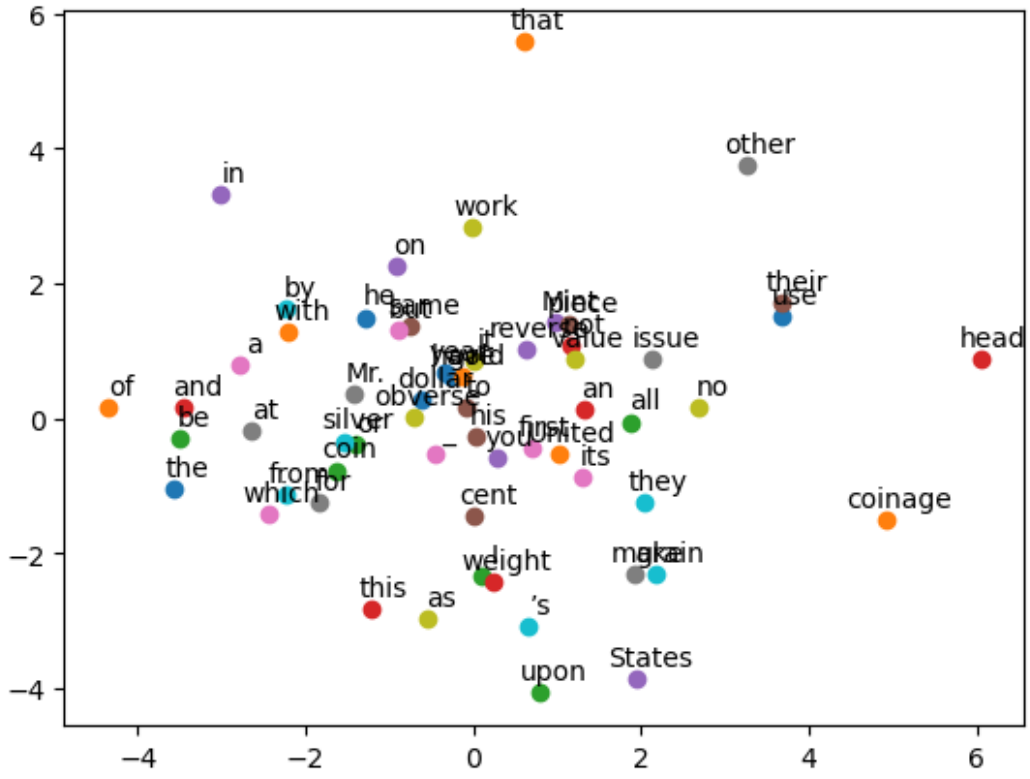

Figure 6: SGNS Word Embedding Chart