Assignment 5 – vvk5231

A. For this question, I have created 2 Jupyter Notebook files. Since the file with the visualizations exceeds the size limit on github, the file was not rendering, So I created another file with just the code.

All the comments and explanations can be found in the file itself.

A.ipynb : https://github.com/Sumanth-Katnam/IST-526/blob/main/Assignment%205/A.ipynb
- Contains code and comments.
- Will be rendered for display.

A_Outputs.ipynb:
https://github.com/Sumanth-Katnam/IST-526/blob/main/Assignment%205/A_Outputs.ipynb
- Contains code, comments, and visualizations.
- Will give the error “The notebook took too long to render.”

B. The code and comments for this can be found in the file B.ipynb : https://github.com/Sumanth-Katnam/IST-526/blob/main/Assignment%205/B.ipynb

C. The code and comments for this can be found in the file C.ipynb : https://github.com/Sumanth-Katnam/IST-526/blob/main/Assignment%205/C.ipynb

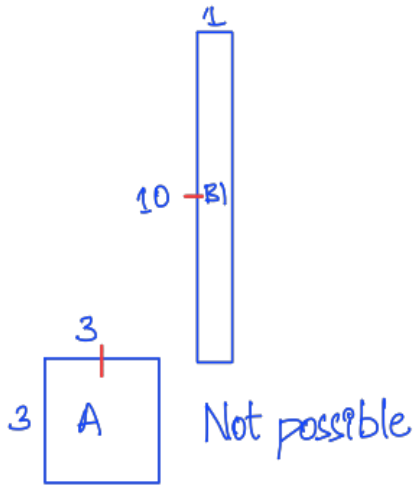
D. Skipped

E. The code and comments for this can be found in the file E.ipynb : https://github.com/Sumanth-Katnam/IST-526/blob/main/Assignment%205/E.ipynb

Assignment 5 – vvk5231

F.

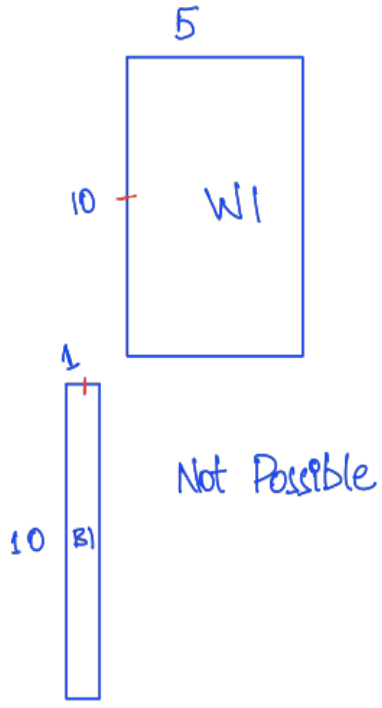
i) R = A * B1



Not possible

---

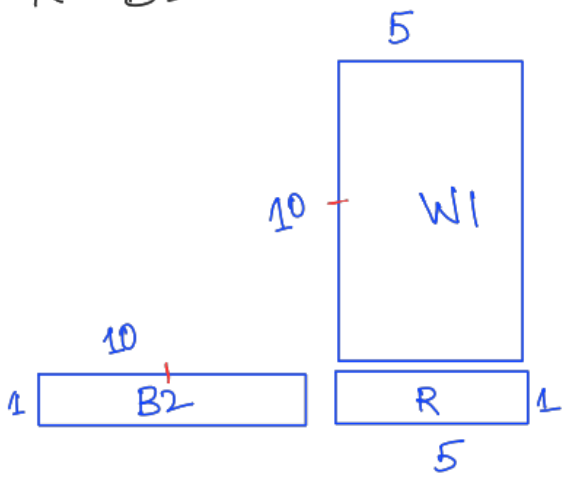
ii) R = B1 * W1



Not Possible

Assignment 5 – vvk5231

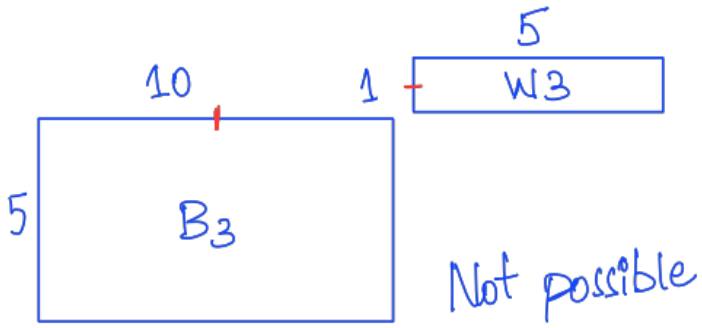
iii) R = B2 * W1



R will be a 1x5 matrix.

---

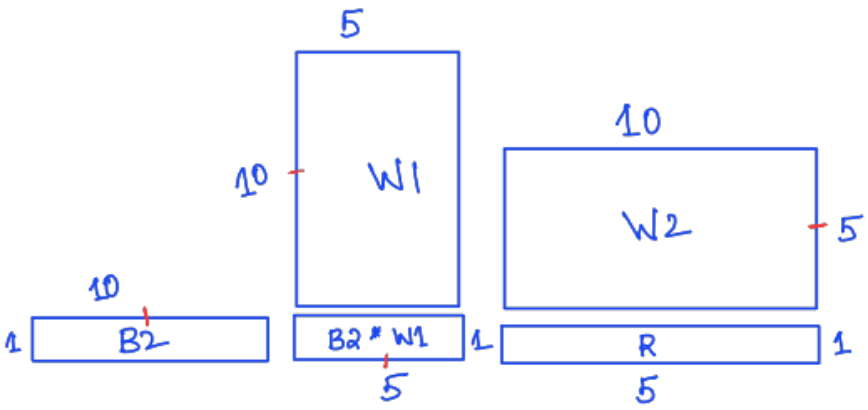
iv) R = B3 * W3



Not possible

Assignment 5 – vvk5231

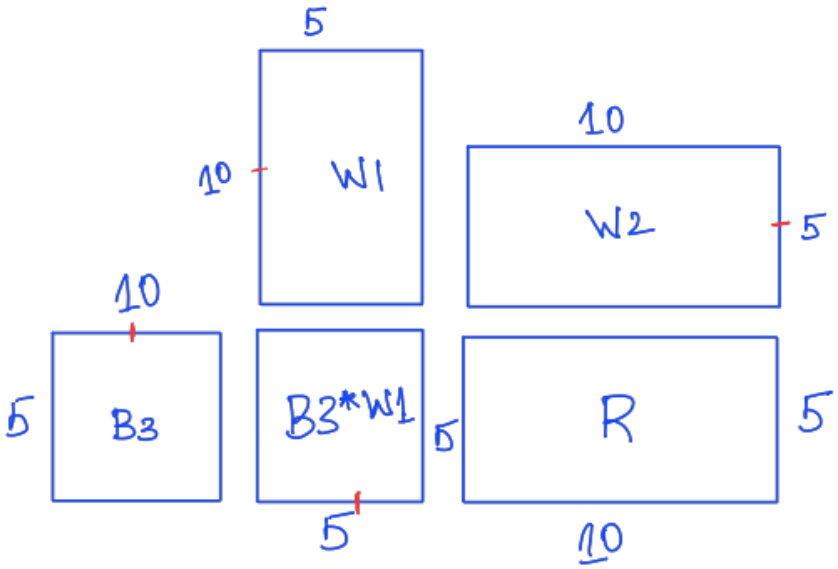
v) R = (B2 * W1) * W2



R will be a 1x5 matrix.

---

vi) R = (B3 * W1) * W2



R will be a 5x10 matrix.

Assignment 5 – vvk5231

G. [10 pt + 5 pt] Describe how Word2Vec model works based on this white paper: [tutorial] Deep NLP: Word Vectors with Word2Vec.pdf

Word2Vec model is used to preprocess the text data for tasks like Sentiment Analysis, Natural entity recognition, etc., in NLP. This model will help us preserve the semantic data of the given text data, which further eases up the tasks mentioned above.

The main idea of Word2Vec model is to create word vectors of the given data. Word vectors are vector representations of a word (from the data) that takes lesser space and holds semantic information about them. The core principle behind word vectors is that "Similar words occur more frequently together than dissimilar words".

The first thing we do for Word2Vec is to collect word co-occurrence data which tells us which words are occurring closer to each other. For this, we use context window which is the number of neighboring words that have to be considered as a word pair for a given word.

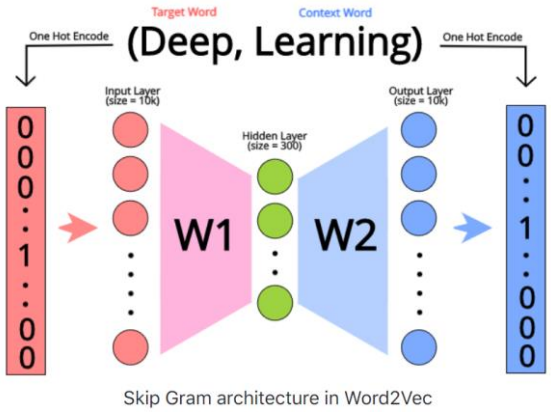
For Example, take the sentence "I am good" and the window size to be 1. For the first word "I", we have only one neighboring word "am", so the window pair (words that occur together) is (I, am). For the second word "am", we have two neighbors with the given window size, so the window pairs would be (am, I) and (am, good). Similarly for the last word we get (good, am). The first word in the pair is called "Target word" and the second word is called as "Context word".

In the Word2Vec, there are two architectures

1) **Skip Gram:** In this architecture, we pass the target word and try to predict each of the context words. A neural network is used for the prediction task. The target word will be fed into the neural network as a single hot encoded form, and the context word will be outputted as a single hot encoded version. Hence, the size of these layers will be equal to the size of the vocabulary.

   Neural network will have one hidden layer, and this will determine the size of the word vectors that are obtained at the end. The architecture is clearly represented in pictorial format in the paper.

Assignment 5 – vvk5231

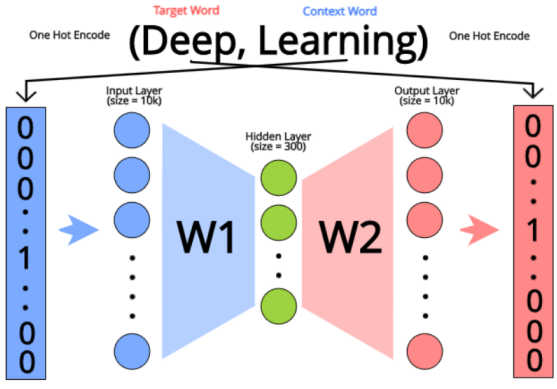


Skip Gram architecture in Word2Vec

Here they have considered the size of the vocabulary to be 10000 and the size of the hidden layer to be 300, So the resultant word vectors will be of the size 300. The W1 and W2 are called weight matrices that are used to convert the data from one layer to another, and this helps us in calculating the word vectors.

The entire (Target word, context word) dataset is passed onto this neural network pairwise to train the network. Essentially, the neural network is attempting to predict which context words are likely to appear given a target word. After the training the network will be able to predict the context words for any given input target word.

2) **CBOW:** The main difference from the previous architecture is that, here we will be trying to predict the target words from the given context words. Once again, I will be using the pictorial representation given in the paper to better explain the approach.



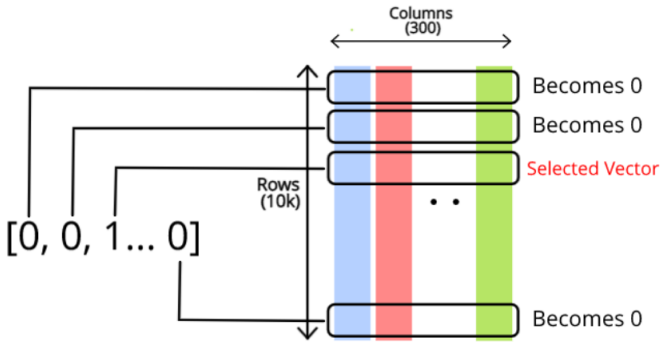
CBOW Architecture in Word2Vec

Here, the size of the vocabulary and the size of the hidden layer are the same from the previous architecture. The main difference is that this is takes the context words as the inputs and gives us the target word that is highly likely to occur as a pair with the given words. This is basically an inverted version of Skip Gram architecture.

After the neural network has been trained on the data, it captures the semantic information about the words. Since we are training the network on the entire data, the network eventually starts

understanding which words are semantically similar to which other words. This information is stored as part of the weight vectors used. These weight vectors are constantly updated when encountering new set of inputs and by the end of the training the network has learnt all semantic information it can retrieve from the inputs.

We will be using the Embedding layer (W1 weight vector) to extract the word vectors for a given word. In the given pictorial examples, the wight vector W1 is of the dimensions 10000x3. A one hot encoded vector of the input is used, which is multiplied with the weight vector and the resultant is the word vector of size 1x300. Since, in the one hot encoded version, all the values are zero except the positional value of the word (value will be 1), when we multiply with the weight vector, we get only the vector that has corresponding information of the given word. The pictorial representation is as follows.



Neural network first layer multiplication operation (Input Layer * W1)

This selected vector is the word vector that stores the learnt information regarding the given word and this is how, the Word2Vec model is used to extract the word vectors.

Write an arbitrary sentence with at least 10 words and manually create a dataset (i.e., a list of tuples <target, context>, window size: 5) that can be used to train the word2vec model. See below:

For this, I have written a code snippet (can be found at this link) that takes the given word and builds a list of tuples from the given sentence with a given window size.

Sentence Chosen: "In Their Last Moments, People Show You Who They Really Are."

Window Size: 5

NOTE: I have stripped the punctuation from the sentence to make sure the pairs contain only the words.

Output: [('In', 'Their'), ('In', 'Last'), ('In', 'Moments'), ('In', 'People'), ('Their', 'In'), ('Their', 'Last'), ('Their', 'Moments'), ('Their', 'People'), ('Their', 'Show'), ('Last', 'Their'), ('Last', 'In'), ('Last', 'Moments'), ('Last', 'People'), ('Last', 'Show'), ('Last', 'You'), ('Moments', 'Last'), ('Moments', 'Their'), ('Moments', 'In'), ('Moments', 'People'), ('Moments', 'Show'), ('Moments', 'You'), ('Moments', 'Who'), ('People', 'Moments'), ('People', 'Last'), ('People', 'Their'), ('People', 'In'),

Assignment 5 – vvk5231

('People', 'Show'), ('People', 'You'), ('People', 'Who'), ('People', 'They'), ('Show', 'People'), ('Show', 'Moments'), ('Show', 'Last'), ('Show', 'Their'), ('Show', 'You'), ('Show', 'Who'), ('Show', 'They'), ('Show', 'Really'), ('You', 'Show'), ('You', 'People'), ('You', 'Moments'), ('You', 'Last'), ('You', 'Who'), ('You', 'They'), ('You', 'Really'), ('You', 'Are'), ('Who', 'You'), ('Who', 'Show'), ('Who', 'People'), ('Who', 'Moments'), ('Who', 'They'), ('Who', 'Really'), ('Who', 'Are'), ('They', 'Who'), ('They', 'You'), ('They', 'Show'), ('They', 'People'), ('They', 'Really'), ('They', 'Are'), ('Really', 'They'), ('Really', 'Who'), ('Really', 'You'), ('Really', 'Show'), ('Really', 'Are'), ('Are', 'Really'), ('Are', 'They'), ('Are', 'Who'), ('Are', 'You')]

H. The code and comments for this file can be found in the file H.ipynb : https://github.com/Sumanth-Katnam/IST-526/blob/main/Assignment%205/H.ipynb

NOTE: Since I have uploaded the files in the Github using LFS (Large File Storage), some of the plots may not be rendered when you view the Notebook using the link. Running them in google Colab fetches the results perfectly. Kindly let me know in case of any issues/concerns.