Tutorial 2: Word Vectors

1. Read and understand the example implementation of n-gram language modeling at https://pytorch.org/tutorials/beginner/nlp/word_embeddings_tutorial.html
    a. An n-gram is a chunk of *n* consecutive words.
        i. unigrams: “the”, “students”, “opened”, ”their”
        ii. bigrams: “the students”, “students opened”, “opened their”
        iii. trigrams: “the students opened”, “students opened their”
    b. Language Modeling is the task of predicting what word comes next. Formally, given a sequence of words $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots, \boldsymbol{x}^{(t)}$, compute the probability distribution of the next word $\boldsymbol{x}^{(t+1)}$: $p\left(\boldsymbol{x}^{(t+1)} | \boldsymbol{x}^{(t)}, \cdots, \boldsymbol{x}^{(1)}\right)$
        i. where $\boldsymbol{x}^{(t+1)}$ can be any word in the vocabulary $V = \left\{\boldsymbol{w}_1, \cdots, \boldsymbol{w}_{|V|}\right\}$
2. Give answer codes for the exercise “Computing Word Embeddings: Continuous Bag-of-Words” based on the example implementation of n-gram language modeling, as follows:
    a. Filling up the __init__ and forward functions
    b. Giving codes for training and for displaying word vectors of selected words
3. Run the notebook “Gensim word vector visualization of various word vectors” at http://web.stanford.edu/class/cs224n/materials/Gensim.zip
    a. Changing the datapath as follows:
    !wget http://nlp.stanford.edu/data/glove.6B.zip
    !unzip glove.6B.zip
    glove_file = datapath(‘/content/glove.6B.100d.txt')