CSE 416: Introduction to Machine Learning
Assignment #8
Instructor: Pemi Nguyen
due: Friday, May 27, 2022, 11:59 pm.

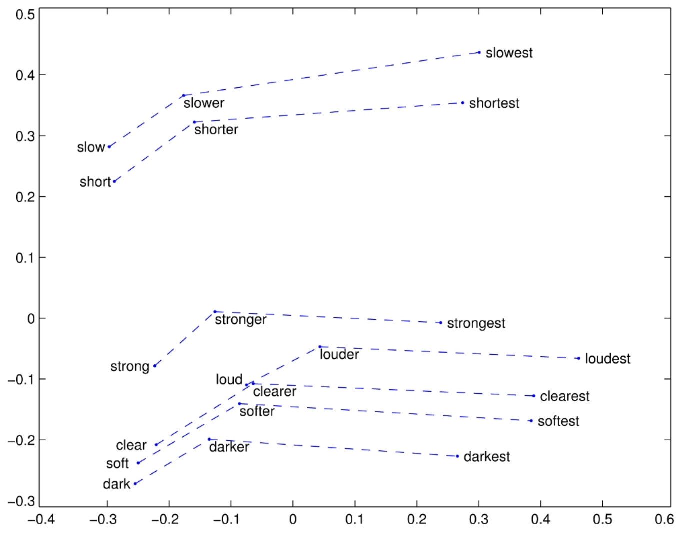
Instructions:

- **Answers:** Please provide detailed yet concise explanations for your work.
- **Turn-in:** Do not write your name on your pages (your Gradescope account will identify you to us) and do not include a copy of the exercise's question in what you turn in. You must use Gradescope to upload your written homework solutions. You will submit a single PDF file containing your solutions to all the exercises in the homework. You must follow the Gradescope prompts that have you link exercise numbers to your pages. **We do not accept handwritten solutions**, so please typeset them, such as Microsoft Word or LaTeX editor.

1. In lectures, we have learned two methods to represent **document embeddings**: bags-of-words and TF-IDF by representing each document as a vector. Now, we're exploring a lower-level way of representing each word as a vector.

   A **word embedding** is a mapping from words, to vector representations of the words. Ideally, the geometry of the vectors will capture the semantic and syntactic meaning of the words. For example, words similar in meaning should have representations which are close to each other in the vector space. A good word embedding provides a means for mapping text into vectors, from which you can then apply all the usual learning algorithms that take, as input, a set of vectors.

   Word embeddings have taken NLP (natural language processing) by storm in the last few years, and have become the backbone for numerous NLP tasks such as question answering and machine translation.



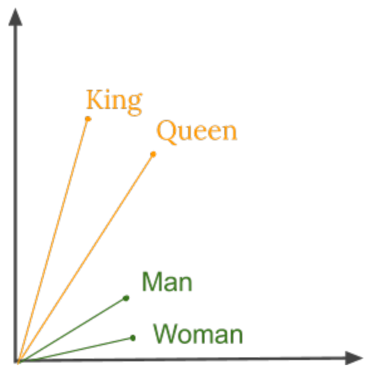
In this Colab link, you're going to explore an example of pretrained word embebeddings called GLoVe and see how they capture semantic and syntactic concepts through two tasks:

- `get_similar_tokens` (Finding k similar words to a particular word): Using k-NN algorithm and cosine similarity metric, we find top k words that are similar to a word w by querying from a corpus all k word vectors that are most similar to it.
- `get_analogy`: Solving word analogy tasks For example, consider an analogy question like:

  king is to queen as man is to ...

  where the goal is to fill in the blank space. This can be solved by finding the word whose embedding that is most similar to the vector:

$$w_{\text{king}} - w_{\text{queen}} + w_{\text{man}}$$



Use the above Colab with GLoVe embedddings of 50 dimensions to:

(a) *(1 point)* Find top 6 words that are *synonyms* of the word **hungry**.

(b) *(1 point)* Find top 4 words that are *antonyms* of the word **good**. (Hint: Find a possible antonym of **good** by yourself and find the remaining ones that are similar to it.)

(c) *(1 point)* Find top 5 words that can fill out the blank:

boat is to plane as captain is to ...

(d) *(2 points)* Find two examples of your own where the embedding does not give correct analogies (an example is provided in the Colab file).

(e) *(2 points)* **(Extra credit)**: There is a dataset in the file analogy_task.txt, which tests the ability of word embeddings to answer analogy questions. Find and report the accuracy of the word embedding approach for solving the analogy task. Comment on the results and which analogies seemed especially difficult for this approach.

Note:

- You don't have to understand most of the code (but it will be helpful if you're interested). Just run the Colab and pass in suitable parameters to the given functions for each question.
- The above code contains more lower-level implementations to load a pretrained word embedding and use kNN algorithm with cosine similarity to find similar words and solve analogy tasks. This alternative Colab link contains more high-level implementations of these two tasks, with a faster runtime. For example, for the previous Colab, you can't load GLoVe embeddings of 300 dimensions, but this current one guarantees a more compact implementation that allowed you to load much larger embeddings.

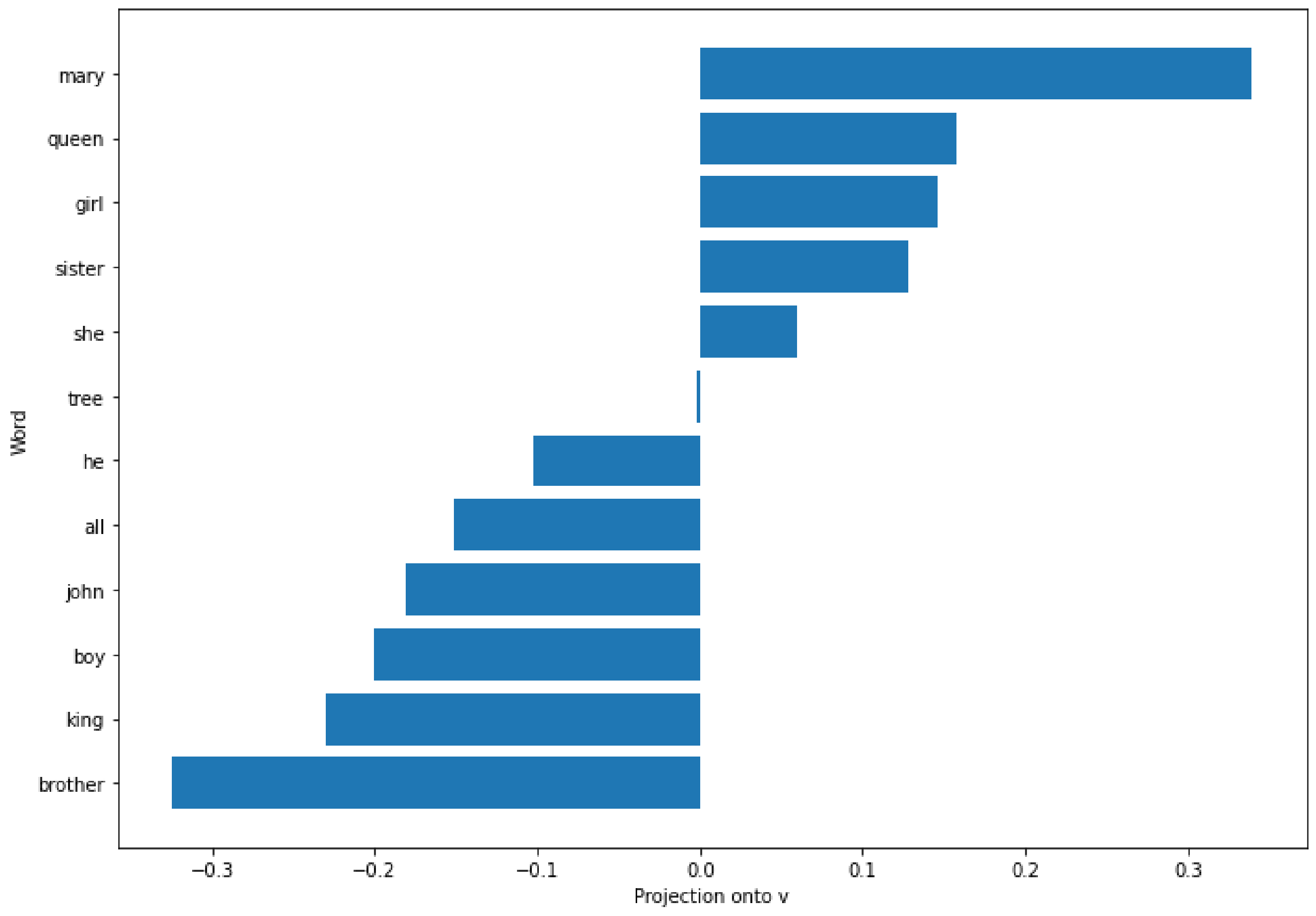
2. Let's train our own word embedding.

   We create a word co-occurrence matrix M of the 10,000 most frequent words from a Wikipedia corpus with 1.5 billion words. Entry M_{ij} of the matrix denotes the number of times in the corpus that the i-th and j-th words occur within 5 words of each other.

   Optional: The file co_occur.csv.gz contains the symmetric co-occurence matrix M. The file dictionary.txt contains the dictionary for interpreting this matrix: The i-th row of the dictionary is the word corresponding to the i-th row and column of M. The dictionary is sorted according to the word frequencies. Therefore the first word in the dictionary "the" is the most common word in the corpus and the first row and column of M contain the co-occurrence counts of "the" with every other word in the dictionary.

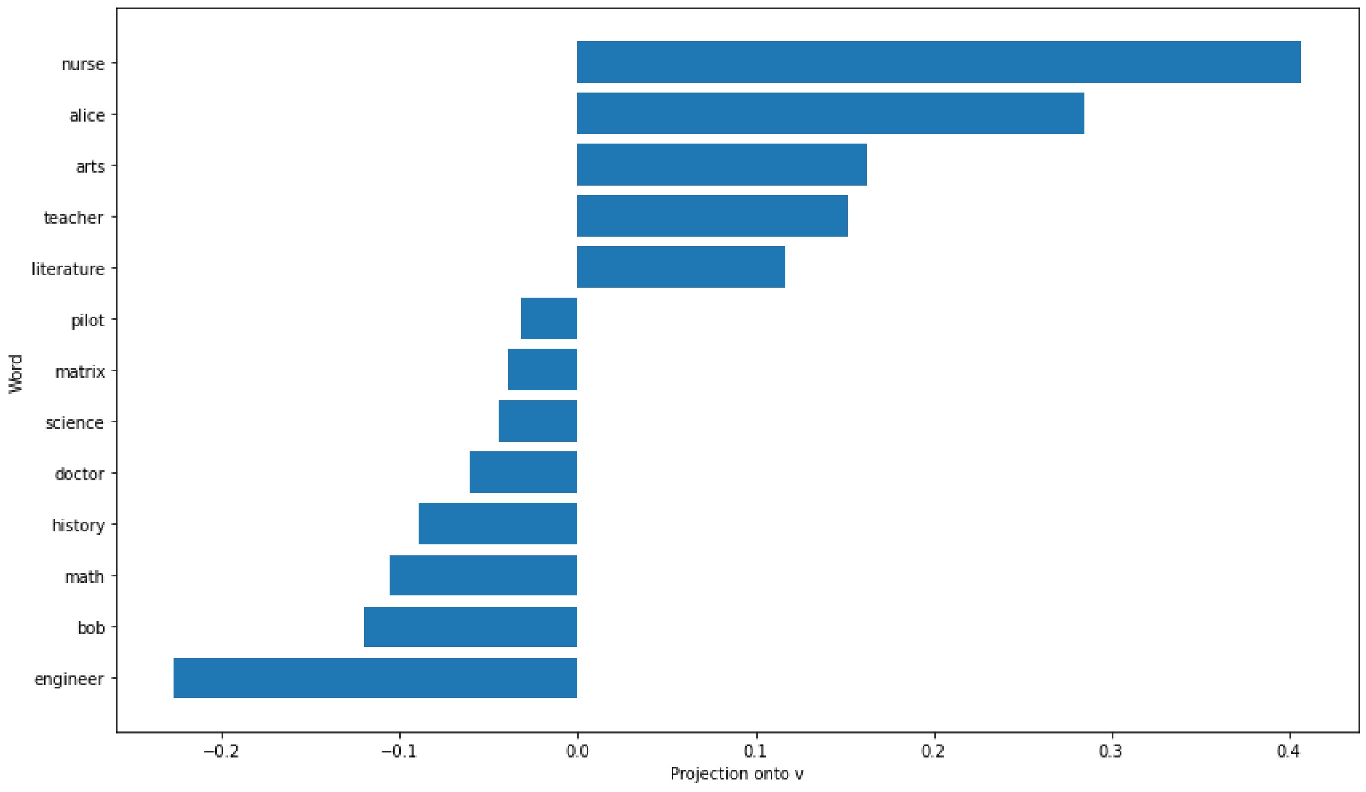
   (a) *(2 points)* Let u_1 be the word embedding for "woman" and let u_2 be the word embedding for "man." Denote $v = u_1 - u_2$.

   Project the embedding of the following words onto v: *boy, girl, brother, sister, king, queen, he, she, john, mary, all, tree*. Following is a plot of the projections of the embeddings of these words marked on a line:



What do you observe based on the values of the projections? Why might it be the case? Explain in no more than 5 sentences.

(b) *(3 points)* We're presenting a similar plot of the projections of the embeddings of the following words onto u: *math, matrix, history, nurse, doctor, pilot, teacher, engineer, science, arts, literature, bob, alice*.



What do you observe? Why do you think this is the case? Do you see a potential problem with this? (For example, suppose LinkedIn used such word embeddings to extract information from candidates' resumes to improve their "search for qualified job candidates" option. What might be the result of this?) Relate to what we have learned about machine learning biases.

(c) **(Extra credit)** *(2 points)* Compare two ways to embed a document: using a document vector through the TF-IDF approach and using an embedding matrix where each word is a vector through the above approach. In your opinion, what is one possible advantage and one possible disadvantage of the word embedding approach?

(d) **(Extra credit)** *(2 points)* Propose one method of mitigating the problem discussed in part b. There are many acceptable answers, and this is an ongoing area of research, so be creative and do not worry about making your answer overly rigorous. [Hint: If you need inspiration, see the original paper that surfaced this issue and discussed some approaches to "debiasing" word embeddings.]