Neuroscience 299: Computing with High-Dimensional Vectors
Assignment 4: Representation and manipulation of data structures
Due September 29, 1pm

Reminder: Please do *either* the writing assignment *or* the programming assignment. Expected length for the writing assignment is approximately 250-500 words, but there is no strict minimum or maximum.

Writing assignment:

Compare and contrast the representations of data structures in HD computing/VSA and in conventional computing. Discuss the relative pros and cons of each, and give examples of situations or applications when one computing paradigm is preferable to the other.

For example, in HD computing/VSA, data structures are represented with a fixed amount of memory, but the vector dimensionality constrains the number of items that can be recovered after being stored. In conventional computing, however, the amount of memory used depends on the number of items stored, and the retrieval of items is also different.

As a starting point for your discussion, consider checking Section V in "Vector Symbolic Architectures as a Computing Framework for Nanoscale Hardware" (2021).

Programming assignment:

You will work with the task of recognizing various European languages as formulated in "Language Geometry Using Random Indexing" (2016).

The task is to identify a language of a given text sample (i.e., for a string of symbols). The language recognition is done for 21 European languages. The list of languages is as follows: Bulgarian, Czech, Danish, German, Greek, English, Estonian, Finnish, French, Hungarian, Italian, Latvian, Lithuanian, Dutch, Polish, Portuguese, Romanian, Slovak, Slovene, Spanish, Swedish.

The training data are based on the Wortschatz Corpora. The average size of a language's corpus in the training data is 1,075,634.8 ± 118,965.5 symbols.

These data are provided in a folder called "training_texts". Please download it using the following link:
https://www.dropbox.com/sh/hux29e50hzzwpki/AACcfkhLBOvBXP4yai_QOjKHa?dl=0

We will focus on the process of constructing the profile HD vector of each language, which contains the *n*-gram statistics observed in the training data for that language. The task of classification itself will be a future assignment, and we will provide the test dataset then.

For this assignment, you are provided with the Jupyter notebook (see course website) that includes the code to support loading data and various placeholders.

The provided Jupyter notebook is missing a few steps, which you will implement (search for ***#ToDo*** comment in the notebook). We will use the MAP model to construct the profile HD vectors, so please use your implementation of MAP from Assignment 1.

Note. Depending on the efficiency of your implementation, processing the dataset may take considerable time. If processing takes too long, consider changing the variable "dataSizeLimit" in the "Parameters" cell. This variable regulates the length of the training sequence. By default it is set to 0, which means that the whole training sequence is considered.

Once you have implemented the process of forming profile HD vectors and measuring their similarity, run your code to process the training data, in order to obtain the profile HD vector for each language.

Please perform the following experiments and answer the following questions:

1. Visualize the cosine similarities between languages' profile HD vectors (for example, by showing a heatmap of pairwise similarities). Are your results qualitatively similar to Figure 1 in "Language Geometry Using Random Indexing"?
2. By default, the size of *n*-grams is set to 3. How does your visualization change if you change the value of *n*? What value of *n* seems to be optimal? What happens to cosine similarities when *n* is either too small or too large?
3. By default, the dimensionality of HD vectors is set to 1,024. Is this dimensionality enough to visualize the similarity between languages' n-gram statistics? What happens if the dimensionality of HD vectors is reduced dramatically?
4. **[Extra]**: Use a dimensionality reduction method (e.g., MDS) to make a figure similar to Figure 1 in "Language Geometry Using Random Indexing". Are your results consistent with that figure?
5. **[Extra]**: Run a clustering algorithm (e.g., *k*-means) using the obtained languages' profile HD vectors as input data. For this and for (4), it may be helpful to normalize the profile HD vectors. What clusters do you obtain? Are there an optimal number of clusters (by some definition of optimal, such as Bayesian information criterion)?