Neuroscience 299: Computing with High-Dimensional Vectors
Assignment 3: Semantic vectors
Due September 22, 1pm

Writing assignment:

You have two topics to choose from. We ask you to focus on one topic and provide thorough support for your answer, rather than trying to answer both. Both topics have multiple good answers, and we encourage you to prioritize depth over breadth. Though the write-up should be done individually, we also strongly encourage discussing with your peers (for example, over Piazza)!

a) *Why use high-dimensional vectors for word embeddings, specifically?* What properties would we like to achieve by embedding words into a high-dimensional space? What are the tradeoffs between the dimensionality of embeddings and their practicality? Could embeddings with certain types of components (e.g., complex, real-valued, binary) be preferable over the others? Might sparsity of embeddings be a desirable feature?

   Note: You should consider representations other than "one-hot encodings", i.e. where the dimension of the vectors is the number of words in the dictionary, and all vector elements are 0 except element *i* for word *i*.

b) *Compare and contrast the RI and BEAGLE approaches to word embedding to the popular Word2vec.* While initially the approaches might look completely different, upon a closer look one can find similarities. Some factors worth considering: how embeddings are generated, the duration of training, the how to take into account word order information, etc.

   For a detailed comparison of RI and BEAGLE approaches, please consult Recchia et al., "Encoding Sequential Information in Semantic Space Models: Comparing Holographic Reduced Representation and Random Permutation" (2015).

Programming assignment:

In this assignment, we will work with data-driven models for forming representations of words (commonly referred to as embeddings). Given a corpus of text, the goal is to form a high-dimensional vector representing a word in the vocabulary. The challenge here is to provide vector representations of words such that the relationship between vectors mirrors the linguistic relationship between words.

As the training data, you will use a small text corpus, which is a preprocessed subset of the Gutenberg Dataset (link).
Please download the file ("Glemmatized.txt") containing the text corpus using the following link: https://www.dropbox.com/s/grn2pnebjtr2feq/Glemmatized.txt?dl=0.

To evaluate the quality of the obtained embeddings, we will use the Test of English as a Foreign Language (TOEFL) synonymy assessment. The dataset contains 80 synonym tasks. Each task includes a query word and 4 alternatives. One of the alternatives is a synonym of the query word. If the chosen word is the correct answer (i.e., the synonym), then the score on the overall dataset increases by 1. Thus, the overall score on the TOEFL synonymy assessment is an integer number between 0 and 80. Note that the expected score by chance is 20 (i.e., 25%).

Please download the file ("TOEFL_synonyms.txt") containing the TOEFL synonymy assessment using the following link: https://www.dropbox.com/s/vvrelyw1x9xjb7f/TOEFL_synonyms.txt?dl=0.

For this assignment, you are provided with a Jupyter notebook (check the website) that includes the code to support loading data; initializing the embeddings; evaluating the obtained embeddings on the synonym tasks, etc.

Note. Usually before building word embeddings, the original text is pre-processed. Common pre-processing steps include disregarding the most frequent words (e.g., articles "a" and "the") and lemmatization (read more about it https://en.wikipedia.org/wiki/Lemmatisation), which changes words to their default forms (e.g., were -> be; goes -> go; tables -> table; etc.).
For the sake of convenience, the provided files contain words that are already lemmatized so the code does not contain the lemmatization step explicitly.

The provided Jupyter notebook misses a few steps, which are to be implemented by you (search for ***#ToDo*** comment in the notebook).
When implementing these steps, you have two algorithms to choose from:

a) Implement the formation of word embeddings using the Random Indexing approach as described in "An Introduction to Random Indexing" (2005) and "Permutations as a Means to Encode Order in Word Space" (2008).
b) Implement the formation of word embeddings using the BEAGLE approach as described in "Representing Word Meaning and Order Information in a Composite Holographic Lexicon" (2007).

Note 1. For both approaches, please implement the "context" and "order" parts of the embedding.
Note 2. The stem of the code provided in the notebook assumes that the RI approach is going to be implemented. So, implementing BEAGLE will be slightly more challenging as you will have modify the code around accordingly. For example, instead of the window around the focus word (implemented in the notebook) you would need to consider the whole sentence.

Once you have implemented the chosen approach to form the embeddings, please perform the following experiments:

1. Get the performance of the embeddings on the TOEFL synonymy assessment for several different dimensionalities. The dimensionality is up to you, but use different values (e.g., 512, 1024, 2048, etc.). If your computational resources allow run simulations several times (e.g., 5) for each dimensionality.
2. In the case of working with Random Indexing:
    a. How does the size of the window around the focus word affect the results on the TOEFL synonymy assessment?

    In the case of working with BEAGLE:
    b. How does the size of *n*-grams taken into embeddings affect the results on the TOEFL synonymy assessment?
3. Report the accuracy on the TOEFL synonymy assessment for all simulations. Elaborate how accuracy changes with the dimensionality.
4. How is accuracy on the TOEFL synonymy assessment affected when only context or only order parts of the embedding are used?
5. (Optional) How does performance of the algorithm depend on the amount of size of the dataset?