1. Intro & Word Vectors

- Words as discrete symbols
    - Before Word2vec, words were encoded as distinct one-hot vectors
        - i.e. [0, 0, …, 1, …, 0, 0]
    - This had drawbacks
        - No way to assess similarity of words
        - Because the encodings were essentially standard basis vectors, even similar words (e.g. "king" and "queen") were orthogonal
- Distributional semantics
    - Rather than one-hot vectors, place words that occur near one other in natural language near each other in a 300-dimensional latent space
        - i.e. [-0.207, 0.065, -0.004, 0.703, …, -0.831]
- Word2vec algorithms
    - Basic idea:
        - 1. Take a piece of text
        - 2. Choose a center word
        - 3. Predict the probability of words around a center world occurring using the dot products of the compared words
        - 4. Optimize the model to raise the probability of the words around the center word occurring
        - 5. Turn this into a probability distribution using a softmax function
        - 6. Move to next word
    - These predicted probabilities never end up being big (e.g. impossible to predict exactly what word will be before or after "croissant" with great accuracy), but it's fairly accurate given the difficulty of the problem
- Vector analogies:
    - Simple vector calculus can be used to look at analogies
        - If you take "king", subtract the vector "man" and add the vector "woman", the new vector should be close to "queen"
            - In English this is to say "man" is to "king" as "woman" is to "queen"
        - This extends to analogies in definitions to grammatical functions