Exercise 1 (Word Embeddings Theory)

Answer the following questions regarding distributed vector representations:

a) What is the basic intuition behind word2vec and similar methods?

Basically, predict between every word and its context words

Need to go from symbolic to distributed representation since w/ atomic representation we have problem of missing similarity information.

- make use of distributional similarity to solve this problem
    - represent a word by means of its neighbors → predict the contextual content the word is appearing in to represent/predict its meaning
    - build dense vector for each word type chosen so it's good to predict other words appearing in its context

We have a vector for center words and context words and we aim to predict between a center word and the context words in terms of word vectors
For each estimation step take one word (the target word) and define some window size for the number of preceeding and succeeding words (the context words)

- maximize the probability of any context word given the current center/target word
- loss func: negative log likelihood
- hyper parameters: window size, ... (these can be adjusted)

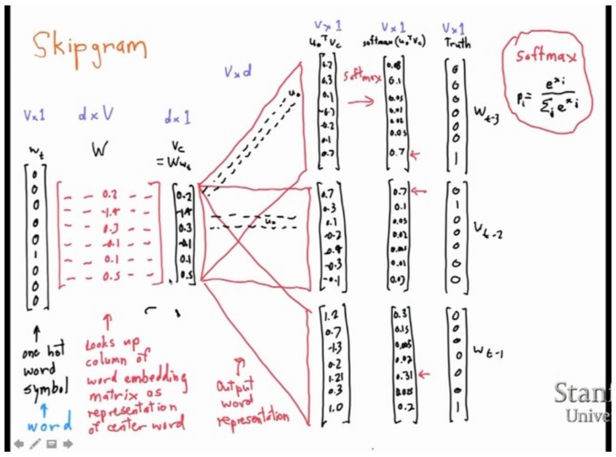
Training: turn numbers into probability distribution with softmax function

b) How are those distributed representations of words computed?

There are two algorithms for computing distributed representation of words. Either with skip-grams or continuous bag of words

- skip-grams (SG)
    - is position independent -> aim to predict context words given the target word
- continuous bag of words (CBOW)
    - aims to predict the target word from a bag of context words

For SG: (graphic taken from [Lecture 2 | Word Vector Representations: word2vec, Stanford School of Engineering, Chris Manning])



- ➢ First we have given the one-hot word representation for the center word
- ➢ Then we lookup this center word in a matrix where all center words are represented in - this is simply done by multiplying the one-hot vector representation of the center word we want to look up with the word embedding matrix representing all center words
- ➢ We get out a dense vector representation of the center word
- ➢ Have a second matrix storing the representation of context words
- ➢ For each position in the context we multiply the vector with the context word matrix
- ➢ Then softmax the resulting context word representation vectors to get out a probability distribution