# Homework 7:
# SkipGram (Word2Vec)

Dr. Benjamin Roth
Computerlinguistische Anwendungen

Due: Friday June 01, 2018, 16:00

In this homework you will implement training word vectors using stochastic gradient descent. You can check your progress using unit tests:
`python3 -m unittest -v hw07_skipgram/test_skipgram.py`

## Exercise 1: Defining the Vocabulary [2 points]

Complete the function `vocabulary_to_id_for_wordlist(word_list, vocab_size)` in the file `utils.py`. It returns a dictionary which maps the `vocab_size` most frequent words in a word list to a number (= their row in the embedding matrizes).

## Exercise 2: Logistic Sigmoid Function [2 points]

Complete the function `sigmoid(x)` in the file `utils.py`. It calculates the logistic sigmoid function. Make sure you understand how the sigmoid function can be used to turn dot products of vectors into probabilities.

## Exercise 3: Creating the Training Instances (Tuples) [6 points]

Complete the function `positive_and_negative_cooccurrences(...)` in the file `utils.py`. It takes a corpus (list of words), and returns a generator of trainings triples of the form (target_word_id, context_word_id, Label), as discussed in the lecture.

- The training corpus contains the **positive instances** as co-occurrences of one *target word* and several *context words* surrounding it, as in the previous homework. They are tuples of the form:
  `(target_word_id, context_word_id, True)`
- The **negative instances** are obtained from the positive instances by adding additional word pairs where the context word id is replaced with a random id uniformly sampled from the entire vocabulary size (you can use `random.randint(...)`). They

are tuples of the form:
`(target_word_id, random_word_id, False)`

- Use the `yield` statement to return a `generator` over the training instances. [1]

## Exercise 4: Performing a Gradient Update [6 points]

The class `SkipGram` will perform training on the positive and negative tuples. Understand how its members are initialized, and how one training iteration over all instances is called from `train_iter`.

Your task is to complete `SkipGram.update(...)` which performs an update for one training instance.

- First, calculate `prob_pos`, the probability $P(True|\text{context, target})$ that a word is a positive co-occurrence from the corpus. Check the lecture materials, how this probability is calculated, and how it is used in the updates.
- Update the context and target embedding matrices according to the update rule from the lecture. **ATTENTION:**
  - When obtaining a vector from a Numpy matrix using indexing (`v=m[i,:]`), this vector is **backed** by the matrix: Any change to the matrix will be reflected in the vector, and vice versa.
  - Since you need to perform two updates (context and target embedding), one vector would have changed after the first update.

    In order for the updates to work correctly, you must use the unchanged vector for the second update. Use `v2 = numpy.copy(v)` to get a vector that is not backed by the matrix.

## Exercise 5: Similarity on the Brown corpus

If you have implemented all functionality, you can train the skipgram model on the Brown corpus by calling:
`python3 -m hw07_skipgram.interactive_skipgram_similarity`

Since we optimized our implementation for readability, rather than for speed, it is very slow and training may take up to 20 minutes.

If you want to train skipgram embeddings on larger corpora, you should use an optimized implementation such as gensim `https://radimrehurek.com/gensim/models/word2vec.html`.

[1] If you don't remember the functionality of the yield statement check the lecture sileds on "Iteratoren, Generatoren, List Comprehensions" of last semester's "Symbolische Programmierspache".