# DL4NLP 2022 — Exercise 3

**Jonas Belouadi, Steffen Eger**
**Natural Language Learning Group (NLLG),**
**University of Bielefeld,**
**Summer Semester 2022**

## Task 1 (15min), $n$-gram language models

Implement a unigram, bigram and trigram language model completing the accompanying code `task1-ngram.py`. 'Train' it on the accompanying texts 'Moby Dick' and 'William Shakespeare' (complete work).

(i) Fix the missing code in `YOUR CODE HERE`.

(ii) Generate language from both texts individually and across the n-grams.

(iii) What differences do you observe, between the texts and across the $n$-grams?

(iv) Are your generated sentences always full sentences?

## Task 2 (15min), training of word embeddings

In this task, you will train your own word2vec-style word embeddings on the transcripts of the TV show Spongebob Squarepants. For that matter, use the gensim[1] library which specializes on representing text as semantic vectors and complete the code in `task2-w2v-train.py`.

(i) Using the transcripts, train your embeddings for 10 epochs. Add the missing code in `YOUR CODE HERE` and report the ten nearest neighbors of the word "pineapple".

Hints:

- Look at the examples and documentation of gensim, especially `https://radimrehurek.com/gensim/models/word2vec.html`.

(ii) What stands out when you look at this result?

(iii) Play with the parameters `vector_size`, `window`, `min_count`, and `sg`. How do they affect your results?

## Task 3 (20min), word similarity

In this task, you will evaluate how well pretrained word2vec embeddings perform in the word similarity task on the WS-353 dataset[2]. This is a classic dataset consisting of 353 word pairs and human annotations of their similarity. Complete the code in `task3-task4-w2v-sim.py` marked with `YOUR CODE HERE`.

(i) Download the pretrained 300-dimensional word2vec embeddings from Google[3] and load them with gensim.

Hints:

- Reading `https://radimrehurek.com/gensim/models/keyedvectors.html` might be helpful.

[1] `https://radimrehurek.com/gensim`
[2] `http://alfonseca.org/eng/research/wordsim353.html`
[3] `https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit?resourcekey=0-wjGZdNAUop6WykTtMip30g`

- If you run into memory errors because the embeddings are too large, you may use the `limit` parameter of the `load_word2vec_format` method.
- `https://radimrehurek.com/gensim/downloader.html` might be useful for downloading the word embeddings programmatically.

(ii) Cosine similarity[4] is a measure of similarity for vectors. Using gensim compute the cosine similarities of the word pairs from WS-353.

(iii) Spearman's rank correlation coefficient[5] is a typical choice for measuring the ranking of two variables. Compute the coefficient between the values assigned by humans and your results from (ii) using scipy.

(iv) What does your resulting coefficient mean?

## Task 4 (10min), bias in word vectors

Word vector representations are known to retain significant semantic and syntactic structure of languages. But they are also prone to carrying and amplifying bias which can perpetrate discrimination in various applications. In this task we will investigate the inherent gender bias of the Google word2vec embeddings. Complete the remaining snippets in `task3-task4-w2v-sim.py` marked with `YOUR CODE HERE`.

(i) Compare the distances of the words `he` and `she` to the words `boss` and `receptionist`. What do you notice? Aggregate the absolute biases into a single value.

(ii) Bonus task: Bias is a serious problem in neural networks in general, but there are techniques which can help to attenuate it. Bias Linear Projection[6] takes a list of word pairs $W_j = \{w_j^+, wj^-\}$ that should be debiased and using their embeddings $e_j$ computes a distance vector $\overleftarrow{e_i} = e_i^+ - e_i^-$ of each pair. These distances are used to compute a bias vector $v_b$ which supposedly captures the bias direction. Finally, all embeddings are modified by subtracting their projections onto this bias vector, i.e.:

$$f(e_j) = e_j - (e_j \cdot v_b)\, v_b, \tag{1}$$

where $\cdot$ is the dot-product. Implement Equation (1) and report any changes to the biases you observed in (i).

[4] `https://en.wikipedia.org/wiki/Cosine_similarity`
[5] `https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.spearmanr.html`
[6] `http://proceedings.mlr.press/v89/dev19a.html`