# Statistical NLP — Homework Exercise 1

**Jonas Belouadi, Philipp Cimiano**
**Semantic Computing Group**
**Bielefeld University**
**Winter Semester 2023/2024**

**Note on Submission** You are allowed to submit in groups of two people. When you want to submit in a group put the name of your colleague in the comment field. Only one person needs to submit.

All solutions have to be uploaded together as a single zip file to LernraumPlus. Provide some information about how to execute your Python code. Non-code answers should be included in a PDF-file. You are only allowed to use the Python standard library for this task! If you're new to Python the official Python Tutorial available under `https://docs.python.org/3/tutorial/index.html` is a good resource.

## Task 1 [3+6+1 points]

Download the file `corpus.zip` from LernraumPlus. The file `corpus.txt` contains the corpus we will use in this worksheet, there is exactly one sentence in each line of this file. A sentence is a sequence $w_1, \ldots, w_N$ of words, where $w_1$ is the first word in the sentence, $w_N$ is the last word and $N$ is the number of words in the sentence. Now we define some distributions of words for the provided corpus:

- $P(w)$ is the distribution of all words in the corpus
- $P(w_i|w_{i-1})$ is the distribution of words given the previous word in a word sequence is $w_{i-1}$
- $P(w_i|w_{i-1}, w_{i-2})$ is the distribution of words at position $i$ in a word sequence given the word at position $i-1$ is $w_{i-1}$ and the word at position $i-2$ is $w_{i-2}$

a) First of all you need to tokenize the corpus. Therefore, implement a Python function `tokenize_sentence` which takes a single string as input (representing a sentence) and returns a list of words. You may ignore commas, semicolons and colons.

b) Provide Python code for representing and learning the distributions $P(w)$, $P(w_i|w_{i-1})$ and $P(w_i|w_{i-1}, w_{i-2})$. To do this, implement the following functions which should return a probability distribution over the whole vocabulary:

- `unigram_distribution()`
- `bigram_distribution(w1)`
- `trigram_distribution(w1, w2)`

c) How does the number of parameters of these distributions scale with the number of different words in the corpus? Explain your answer!

**Hint** Introduce special words to model the beginning and the end of a sentence!

## Task 2 [3+6+1 points]

a) Implement a function `sample(distribution)` for drawing a sample from the distributions $P(w)$, $P(w_i|w_{i-1})$ and $P(w_i|w_{i-1}, w_{i-2})$ according to the algorithm presented in the lecture. Make use of your solution of Task 1.

b) Use the statistical information of the provided corpus to implement three different sentence generators, i.e., use the distributions $P(w)$, $P(w_i|w_{i-1})$, $P(w_i|w_{i-1}, w_{i-2})$ and your code of a) to successively generate single words. In other words, your first sentence generator should use the distribution $P(w)$, the second one $P(w_i|w_{i-1})$ and the third one $P(w_i|w_{i-1}, w_{i-2})$. The sentence should be returned as a string. Use the following naming conventions:

- `generate_sentence_unigram()`
- `generate_sentence_bigram()`
- `generate_sentence_trigram()`

c) Describe the results of the three sentence generators you implemented. Try to explain the results.