# Natural Language Processing – Assignment #1

## Assignment description

In this assignment you will perform two tasks. Both make use of a large English dataset -- download a subset of 2018 Wikipedia dump (large well-formed English corpus) via this link: https://drive.google.com/file/d/15H-pg40Epx2u6GE14vquCdF_-VlOgOG7/view?usp=sharing. These are 10M rows from the full dataset that should suffice for the purpose of this assignment.

## Task #1: testing the Heaps' law in natural language

1. The python file plot_heaps_zipf_laws.py contains a function plotting log-log scale Zipf's distribution. Run it and make sure you see the expected output for plot_zipf_law(), and fully understand the code producing it.

2. Implement a new function that plots the Heaps' law for the dataset you downloaded. Does the plot fit your intuition? Copy your plot into a document to be submitted for this assignment.

## Task #2: statistical language models – solving a cloze

This task deals with solving a cloze: a short text where certain words were removed, and you're required to fill in the missing items from a given list. Cloze is a popular task in language proficiency level assessment and requires understanding of words and their context.

**Input**:

(1) a large well-formed English corpus (a small portion of Wikipedia 2018)

(2) a file with cloze: a short text where removed words were replaced with "_____"

(3) a file with words removed from the text (in a random order) – use these words to solve the cloze

**Output**: print to the console a list of words in the order they are assigned to placeholders in (2)

For example, for this input paragraph:

The wooden bridge, dating from the Middle Ages, across the Aare was destroyed by floods three _____ in thirty years, and was replaced with a steel suspension _____ in 1851. This was replaced by a _____ bridge in 1952.

And the list of words: bridge, concrete, times

The output should be (a list of words):

['times', 'bridge', 'concrete']

You are given two files: config.json and main.py, where config.json contains corpus, input cloze, and candidates files location. You can change these locations in config.json, but **do not change** the code in main(). Your task is to implement the function solve_cloze():

```
def solve_cloze(input, candidates, corpus, left_only):
    # todo: implement this function
    ...
```

left_only is a boolean (true or false) parameter that indicates if we should only use the left (preceding) context of the missing word for solution, or we are not restricted in any way.

The accuracy of the solution will be estimated as the ratio of correctly assigned words out of total.

Question: Do you have any estimation of the accuracy of a random word selection?

One way to estimate that is via experiment: generate 100 random solutions for the given cloze and compute their mean accuracy: you are supposed to get a very low number, meaning the **chance accuracy** is very low. Write down to the document the chance accuracy you got for the candidate list in the assignment. Compare the chance accuracy to the accuracy you got for your solution – although not perfect, it should be much better than chance.

Comments:

- Although generally desirable, you are not required to perform tokenization and lemmatization in this assignment due to runtime constraints. Working with large enough data, we hope that the quantity makes up for the quality in this case.
- Assume that the words used to solve a cloze appear in the Wikipedia fragment you are using.
- You may not get a perfect result: the task is hard, we are limited to the set of tools we learned so far, and runtime constraints. However, you will get accuracy much higher than a chance.
- Include the function computing chance results in your submission.
- It's recommended to test your code on additional input files, not just the attached one. Create your own examples to test various cases! Keep the number of missing words small, e.g., 10.
- In case you need to use lexicon size in your implementation – assume a 50,000 word lexicon.
- You can (and encouraged) to break down your code into additional functions, for good design and readability. Document your code with comments where needed.
- Make sure your code runtime does not exceed 15 minutes.

## Submission

Submit a single zip file – assignment1_ xxxxxxxxx_xxxxxxxxx.zip , where “xxxxxxxxx” stands for a student id. Please specify two student ids (your and your partner’s). It should include three files:

(1) A pdf or word document with your Heaps’ law plot for task #1, and the chance accuracy for random cloze solution you got for task #2.

(2) The attached plot_heaps_zipf_laws.py file including your implementation for task #1.

(3) Your implementation for task #2 (including computation of chance accuracy) -- main.py.

Grading criteria include: correctness (the major part), code documentation, design and readability.

Good Luck!