Before you turn this problem in, make sure everything runs as expected. First, **restart the kernel** (in the menubar, select Kernel→Restart) and then **run all cells** (in the menubar, select Cell→Run All).

Make sure you fill in any place that says `YOUR CODE HERE` or "YOUR ANSWER HERE".

Also, please write how much time it took you to finish the homework. This will not affect your grade in any way and is used for statistical purposes.

In [1]:

```
TIME_SPENT = "01h45m"
```

Homework 1

NLP Basics & NLP Pipelines

Welcome to Homework 1!

The homework contains several tasks. You can find the amount of points that you get for the correct solution in the task header. Maximum amount of points for each homework is *four*.

The **grading** for each task is the following:

- correct answer - **full points**
- insufficient solution or solution resulting in the incorrect output - **half points**
- no answer or completely wrong solution - **no points**

Even if you don't know how to solve the task, we encourage you to write down your thoughts and progress and try to address the issues that stop you from completing the task.

When working on the written tasks, try to make your answers short and accurate. Most of the times, it is possible to answer the question in 1-3 sentences.

When writing code, make it readable. Choose appropriate names for your variables (`a = 'cat'` - not good, `word = 'cat'` - good). Avoid constructing lines of code longer than 100 characters (79 characters is ideal). If needed, provide the commentaries for your code, however, a good code should be easily readable without them :)

Finally, all your answers should be written only by yourself. If you copy them from other sources it will be considered as an academic fraud. You can discuss the tasks with your classmates but each solution must be individual.

Important!: **before sending your solution, do the `Kernel -> Restart & Run All` to ensure that all your code works.**

In [2]:

```
import nltk
from nltk import word_tokenize, sent_tokenize, pos_tag
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords, wordnet

import re
from collections import defaultdict, Counter
import requests
from string import punctuation
```

```
/usr/lib/python3/dist-packages/requests/__init__.py:89: RequestsDependencyWarning: urllib3 (1.26.7) or chardet (3.0.4) doesn't match a supported version!
  warnings.warn("urllib3 ({}) or chardet ({}) doesn't match a supported "
```

In [3]:

```
# Download the text
# The Project Gutenberg eBook of The Adventures of Sherlock Holmes, by Arthur Conan Doyle
url = 'https://www.gutenberg.org/files/1661/1661-0.txt'
raw_text = requests.get(url).content.decode('utf-8')

# Remove the Gutenberg metadata
text_start = "*** START OF THE PROJECT GUTENBERG EBOOK THE ADVENTURES OF SHERLOCK HOLMES ***"
text_end = "*** END OF THE PROJECT GUTENBERG EBOOK THE ADVENTURES OF SHERLOCK HOLMES ***"
raw_text = raw_text.split(text_start)[1].split(text_end)[0].strip()
```

Task 1. Tokenize and count statistics (1 point)

Using either NLTK tools, tokenize your text data.

Compute and output the following:

- number of sentences
- number of tokens
- number of unique tokens (or types)
- average length of a sentence
- average length of a token

In [4]:

```
# Please, use these variables
num_sentences = 0
num_tokens = 0
num_unique_tokens = 0
avg_sentence_len = 0
avg_token_len = 0

# YOUR CODE STARTS HERE
num_sentences = len(sent_tokenize(raw_text))
num_tokens = len([word for word in word_tokenize(raw_text) if word not in punctuation])
num_unique_tokens = len(set([word for word in word_tokenize(raw_text) if word not in punctuation]))
avg_sentence_len = num_tokens/num_sentences
avg_token_len = sum(len(word) for word in word_tokenize(raw_text)) / num_tokens
# YOUR CODE ENDS HERE

print("Number of sentences:", num_sentences)
print("Number of tokens:", num_tokens)
print("Number of unique tokens (or types):", num_unique_tokens)
print("Average sentence length:", avg_sentence_len)
print("Average token length:", avg_token_len)
```

```
Number of sentences: 4716
Number of tokens: 111710
Number of unique tokens (or types): 9638
Average sentence length: 23.687446988973708
Average token length: 4.070459224778444
```

In [ ]:

Task 2. Lemmatization and normalization (1 point)

Using NTLK, lemmatize your data. Make a copy of your data but this time transform all the tokens and lemmas into the lowercase.

Provide the following statistics:

- Number of unique lemmas (original case)
- Number of unique lemmas (lower case)
- Number of unique tokens (original case)
- Number of unique tokens (lower case)

In [5]:

```
def tagset_map(tag):
    tag = re.sub('^N[A-Z]{1,3}$', 'n', tag)
    tag = re.sub('^J[A-Z]{1,2}$', 'a', tag)
    tag = re.sub('^R[A-Z]{1,2}$', 'r', tag)
    tag = re.sub('^V[A-Z]{1,2}$', 'v', tag)
    if tag not in list('narv'):
        tag = 'n'
    return tag


# Lemmatize your data
# YOUR CODE STARTS HERE
from nltk import pos_tag_sents
nltk.download('averaged_perceptron_tagger')
nltk.download('wordnet')
nltk.download('omw-1.4')

lemmatizer = WordNetLemmatizer()


def lemmatize(sents):
    lemmas = []
    for sent in pos_tag_sents(sents):
        for token, tag in sent:
            word_tag = tagset_map(tag)
            lemma = lemmatizer.lemmatize(token, pos=word_tag)
            lemmas.append(lemma)

    return lemmas


sent_raw = [word_tokenize(s) for s in sent_tokenize(raw_text)]
lemmatized_sents = lemmatize(sent_raw)
# YOUR CODE ENDS HERE


# Make a copy of your tokens but in lowercase
# YOUR CODE STARTS HERE
lemmatized_sents_lowercase = [word.lower() for word in lemmatized_sents if word not in punctuation]
tokens_lowercase = [word.lower() for word in word_tokenize(raw_text) if word not in punctuation]
# YOUR CODE ENDS HERE


# Count statistics (no need to calculate the number of unique tokens in original case since we did it in Task 1
# Please, use these variables
num_unique_lemmas = 0
num_unique_lemmas_lower = 0
num_unique_tokens_lower = 0

# YOUR CODE STARTS HERE
num_unique_lemmas = len(set([lemma for lemma in lemmatized_sents if lemma not in punctuation]))
num_unique_lemmas_lower = len(set(lemmatized_sents_lowercase))
num_unique_tokens_lower = len(set(tokens_lowercase))
# YOUR CODE ENDS HERE

# Print out the numbers
print("Number of unique lemmas (original case):", num_unique_lemmas)
print("Number of unique lemmas (lower case):", num_unique_lemmas_lower)
print("Number of unique tokens (original case):", num_unique_tokens)
print("Number of unique tokens (lower case):", num_unique_tokens_lower)
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /home/utlab/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package wordnet to /home/utlab/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to /home/utlab/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
Number of unique lemmas (original case): 8103
Number of unique lemmas (lower case): 7564
Number of unique tokens (original case): 9638
Number of unique tokens (lower case): 9049
```

In [ ]:

Task 3. Preprocessing function (0.5 points)

To make preprocessing easier in the future, wrap everything into a function that takes raw text as input and outputs tokens and lemmas. The function will also have special arguments for removing stopwords, punctuation, and lowercasing the outputs.

NB: NLTK morphological analyzer takes word context into account, so you might want to assign pos tags to the tokens before normalization.

Tip: This book has some punctuation characters that are not present in Python's `punctuation` variable. You might want to return to this task after looking at the results of the next one.

In [6]:

```
nltk.download('stopwords')
def preprocess(raw_text, remove_stopwords=True, remove_punctuation=True, lowercase=True):
    """Preprocess raw text.

    Args:
        raw_text (str): Text to preprocess.
        remove_stopwords (bool, optional): Whether to remove the stopwords or not.
        remove_punctuation (bool, optional): Whether to remove the punctuation or not.
        lowercase (bool, optional): Lowercase all the tokens.

    Returns:
        tokens (list[str]): List of tokens from the text.
        lemmas (list[str]): List of lemmas from the text.

    """
    # YOUR CODE STARTS HERE
    tokens = []
    lemmas = []

    tokens = [word_tokenize(s) for s in sent_tokenize(raw_text)]

    if lowercase:
        tmp_tokens=[]
        for token in tokens:
            lower_token = [word.lower() for word in token]
            tmp_tokens.append(lower_token)
        tokens=tmp_tokens
    if remove_stopwords:
        tmp_tokens=[]
        nltk_stopwords = set(stopwords.words('english'))
        for token in tokens:
            stop_token = [word for word in token if word not in nltk_stopwords]
            tmp_tokens.append(stop_token)
        tokens=tmp_tokens
    if remove_punctuation:
        tmp_tokens=[]
        for token in tokens:
            pun_token = [word for word in token if word not in punctuation]
            tmp_tokens.append(pun_token)
        tokens=tmp_tokens
    for word in pos_tag_sents(tokens):
        tmp_tokens=[]
        for token, tag in word:
            word_tag = tagset_map(tag)
            lemma = lemmatizer.lemmatize(token, pos=word_tag)
            lemmas.append(lemma)
    # YOUR CODE ENDS HERE

    return tokens, lemmas


tokens, lemmas=preprocess(raw_text)
```

```
[nltk_data] Downloading package stopwords to /home/utlab/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Task 4. Splitting the text into chapters (1 point)

The Adventures of Sherlock Holmes has twelve adventures. If you look at the text (https://www.gutenberg.org/files/1661/1661-0.txt) each of them starts with at title, e.g. "I. A SCANDAL IN BOHEMIA" or "II. THE RED-HEADED LEAGUE".

Look through the text and come up with a regular expression that only captures the titles. Write all the titles in order using `re.findall`. Then, split the text into twelve adventures with `re.split`. Finally, join the titles with the corresponding texts in an ordered dict or a list of tuples.

Tip: https://regex101.com/ is a great website to test your regular expressions.

In [7]:

```
# YOUR CODE STARTS HERE
def get_book_dict(raw_text):
    pattern = r'([IVX]+\.\s(?:[A-Z]+(\s|\-|\’)){2,})'

    find_all_titles = re.findall(pattern, raw_text)
    tittles = [tittle[0] for tittle in find_all_titles]
    book_dict = {}

    tmp_raw_text = raw_text
    for tittle in reversed(tittles):
        split_sents = re.split(tittle, tmp_raw_text)
        tmp_raw_text = split_sents[0]
        book_dict[tittle] = split_sents[1]

    return dict(reversed(list(book_dict.items())))
# YOUR CODE ENDS HERE

print(len(get_book_dict(raw_text)))
```

```
12
```

Task 5. Statistics by chapter (0.5 points)

Using your `preprocess` function from the Task 3, for each adventure, print out the following information:

- Title
- Number of tokens
- Number of unique words
- Number of unique lemmas
- Top 20 lemmas

In [8]:

```
# YOUR CODE STARTS HERE
for key, value in get_book_dict(raw_text).items():
    tokens, lemmas = preprocess(value)
    joint_token = []
    num_tokens = 0
    num_types = 0

    for token in tokens:
        joint_token += token

    freqs = Counter(lemmas)

    print('Title:', key.rstrip('\r'))
    print('Number of tokens:', len(joint_token))
    print('Number of unique words:', len(set(joint_token)))
    print('Number of unique lemmas:', len(lemmas))
    print('Top 20 lemmas:', freqs.most_common(20))
    print('---'*20)
# YOUR CODE ENDS HERE
```

```
Title: I. A SCANDAL IN BOHEMIA
Number of tokens: 4437
Number of unique words: 1909
Number of unique lemmas: 4437
Top 20 lemmas: [('“', 283), ('”', 268), ('’', 61), ('holmes', 46), ('say', 39), ('one', 27), ('upon', 25), ('co
uld', 25), ('come', 23), ('see', 22), ('know', 22), ('man', 21), ('may', 21), ('‘', 21), ('take', 19), ('woul
d', 19), ('make', 19), ('street', 18), ('king', 18), ('majesty', 18)]
------------------------------------------------------------
Title: II. THE RED-HEADED LEAGUE
Number of tokens: 4754
Number of unique words: 1880
Number of unique lemmas: 4754
Top 20 lemmas: [('“', 245), ('”', 190), ('’', 138), ('say', 66), ('‘', 62), ('mr.', 54), ('holmes', 53), ('upo
n', 49), ('would', 35), ('come', 31), ('one', 30), ('well', 27), ('see', 26), ('little', 25), ('man', 25), ('co
uld', 23), ('wilson', 22), ('go', 22), ('time', 19), ('take', 19)]
------------------------------------------------------------
Title: III. A CASE OF IDENTITY
Number of tokens: 3568
Number of unique words: 1523
Number of unique lemmas: 3568
Top 20 lemmas: [('“', 164), ('”', 164), ('’', 61), ('say', 56), ('mr.', 47), ('holmes', 46), ('would', 39), ('u
pon', 35), ('come', 33), ('little', 28), ('could', 24), ('hosmer', 22), ('see', 20), ('one', 19), ('windibank',
19), ('man', 18), ('go', 18), ('father', 17), ('make', 16), ('never', 16)]
------------------------------------------------------------
Title: IV. THE BOSCOMBE VALLEY MYSTERY
Number of tokens: 4779
Number of unique words: 1877
Number of unique lemmas: 4779
Top 20 lemmas: [('“', 247), ('”', 206), ('’', 66), ('say', 52), ('holmes', 46), ('man', 43), ('upon', 42), ('se
e', 39), ('know', 38), ('mccarthy', 37), ('father', 33), ('one', 31), ('could', 29), ('come', 27), ('would', 2
7), ('son', 27), ('go', 26), ('little', 25), ('shall', 24), ('young', 24)]
------------------------------------------------------------
Title: V. THE FIVE ORANGE PIPS
Number of tokens: 3748
Number of unique words: 1727
Number of unique lemmas: 3748
Top 20 lemmas: [('“', 195), ('”', 160), ('’', 65), ('upon', 47), ('say', 36), ('‘', 36), ('come', 33), ('one',
29), ('paper', 26), ('holmes', 25), ('take', 25), ('may', 24), ('would', 20), ('man', 20), ('see', 20), ('day',
18), ('must', 18), ('time', 18), ('letter', 18), ('however', 16)]
------------------------------------------------------------
Title: VI. THE MAN WITH THE TWISTED LIP
Number of tokens: 4698
Number of unique words: 1944
Number of unique lemmas: 4698
Top 20 lemmas: [('“', 236), ('”', 223), ('’', 60), ('upon', 54), ('one', 35), ('say', 34), ('see', 30), ('man',
30), ('come', 29), ('st.', 28), ('holmes', 28), ('make', 27), ('could', 27), ('find', 27), ('clair', 27), ('kno
w', 26), ('would', 25), ('little', 21), ('room', 21), ('take', 21)]
------------------------------------------------------------
Title: VII. THE ADVENTURE OF THE BLUE CARBUNCLE
Number of tokens: 4128
Number of unique words: 1723
Number of unique lemmas: 4128
Top 20 lemmas: [('“', 236), ('”', 217), ('’', 108), ('say', 51), ('one', 39), ('man', 39), ('upon', 38), ('holm
es', 37), ('well', 31), ('see', 31), ('‘', 31), ('know', 28), ('hat', 25), ('goose', 25), ('little', 24), ('bir
d', 22), ('would', 22), ('go', 21), ('stone', 20), ('baker', 18)]
------------------------------------------------------------
Title: VIII. THE ADVENTURE OF THE SPECKLED BAND
Number of tokens: 5056
Number of unique words: 2028
Number of unique lemmas: 5056
Top 20 lemmas: [('“', 247), ('”', 233), ('’', 67), ('holmes', 56), ('say', 54), ('upon', 41), ('one', 32), ('se
e', 32), ('room', 30), ('would', 28), ('shall', 25), ('come', 23), ('could', 23), ('bed', 22), ('think', 22),
('sister', 22), ('light', 21), ('roylott', 21), ('dr.', 20), ('must', 20)]
------------------------------------------------------------
Title: IX. THE ADVENTURE OF THE ENGINEER’S THUMB
Number of tokens: 4207
Number of unique words: 1750
Number of unique lemmas: 4207
Top 20 lemmas: [('“', 192), ('’', 126), ('”', 111), ('‘', 86), ('say', 56), ('come', 41), ('upon', 38), ('one',
35), ('time', 25), ('little', 25), ('go', 25), ('could', 24), ('would', 23), ('see', 22), ('colonel', 21), ('do
or', 20), ('think', 20), ('u', 19), ('hand', 19), ('shall', 19)]
------------------------------------------------------------
Title: X. THE ADVENTURE OF THE NOBLE BACHELOR
Number of tokens: 4256
Number of unique words: 1746
Number of unique lemmas: 4256
Top 20 lemmas: [('“', 255), ('”', 247), ('’', 69), ('say', 49), ('st.', 42), ('simon', 38), ('lord', 37), ('hol
mes', 33), ('one', 30), ('make', 30), ('upon', 29), ('come', 28), ('think', 26), ('little', 25), ('see', 24),
('lady', 24), ('hand', 21), ('would', 20), ('take', 20), ('frank', 19)]
------------------------------------------------------------
Title: XI. THE ADVENTURE OF THE BERYL CORONET
Number of tokens: 4704
Number of unique words: 1785
Number of unique lemmas: 4704
Top 20 lemmas: [('“', 234), ('”', 163), ('’', 97), ('‘', 66), ('say', 60), ('could', 36), ('go', 35), ('upon',
33), ('think', 33), ('would', 32), ('one', 31), ('come', 29), ('see', 29), ('holmes', 27), ('man', 27), ('may',
25), ('coronet', 25), ('know', 24), ('tell', 23), ('hand', 22)]
------------------------------------------------------------
Title: XII. THE ADVENTURE OF THE COPPER BEECHES
Number of tokens: 5033
Number of unique words: 1889
Number of unique lemmas: 5033
Top 20 lemmas: [('“', 230), ('”', 143), ('’', 131), ('‘', 76), ('say', 51), ('holmes', 43), ('mr.', 43), ('litt
le', 37), ('rucastle', 37), ('miss', 36), ('could', 36), ('would', 36), ('man', 34), ('one', 34), ('upon', 33),
('come', 32), ('door', 29), ('look', 28), ('go', 28), ('think', 26)]
------------------------------------------------------------
```