Second Year Project:
Natural Language Processing and Deep Learning
Week 1

Barbara Plank, Rob van der Goot, Frey Alfredsson, Marija Stepanović

January 28, 2020

This is your assignment for the first week of the course *Second Year Project*. You should start working on exercises 1–3 from Tuesday, while exercises 4–9 should be done after the Friday lecture.

After completing the whole assignment, you should be:

- familiar with the concept of regular expressions
- able to discuss issues that arise in tokenization and segmentation and implement them in `Python`
- able to log on to the HPC server and submit a job to a GPU node
- able to use the Unix command-line tools for search (`grep`), count (`wc`), and basic text processing (`sed` for substitution), as well as the pipe (`|`), e.g., to count word types or extract a simple word frequency list
- able to copy files to and from a server

Requested reading: Chapter 2 (up to and including 2.4.2) from *Speech and Language Processing* by Jurafsky and Martin [1].

1 Regular Expressions (pen and paper)

For this section, it might be handy to use the website `https://regex101.com/` to test your solutions.

Note: By *word*, we mean any alphabetic string separated from other words by whitespace, any relevant punctuation, line breaks, etc., as defined in [1]. If we do not specify *word*, any substring match might be sufficient.

- Write a regular expression (regex or pattern) that matches any of the following words: 'cat', 'sat', 'mat'. (Bonus: What is a possible long solution? Can you find a shorter solution?)
- Write a regular expression that matches two consecutive repeated words (such as 'the the', 'Hubert Hubert', and so forth).
- Write a regular expression that matches Danish prices indications, e.g., 1,000 kr or 39.95 DKK or 19.95.

2 Tokenization (Jupyter notebook)

Solve the tokenization exercises provided in the notebook `tokenization.ipynb`.

3 Segmentation (Jupyter notebook)

Solve the segmentation exercise provided in the notebook `segmentation.ipynb`.

4 Connecting to the Server

In this exercise you will learn how to connect to the ITU's HPC cluster, which is a collection of servers where ITU researchers and students can schedule resource-intensive tasks. You can connect to the cluster simply by running `ssh ITU-username@hpc.itu.dk` in your preferred command-line shell (e.g. `bash`, `zsh`, or `powershell`) and entering your ITU password when prompted, as shown bellow (make sure to replace `ITU-username` with your own ITU username).

Note: The first time you try to log on, it might take a bit longer to connect because the server needs to initialize a new user profile. Also note that you have to be connected to the internet through an ITU network in order to log on to the HPC; you cannot access the server from your home network.

```
[yourname@YourPC ~]$ ssh ITU-username@hpc.itu.dk
WARNING:  Unauthorized access to this system is forbidden and will be
          prosecuted by law. By accessing this system, you agree that your
          actions may be monitored if unauthorized usage is suspected.

ITU-username@hpc.itu.dk's password:
```

Once connected, your home directory will look like this:

```
[ITU-username@front ~]$ pwd
/home/ITU-username
```

You can solve the following four exercises (5–8) and run other non-intensive tasks on the HPC's login node, but **never** use the login node for resource-intensive jobs, or the server will automatically kick you out and black-list you. All resource-intensive jobs need to be submitted to one of the computing nodes. You will learn how to submit a job as part of exercise 9. For more information on the HPC cluster, please visit `http://hpc.itu.dk`.

Tip: If you are used to coding in a text editor with a modern GUI, we recommend *Visual Studio Code* (available on all major operating systems). The Remote Development extension pack for VS Code allows you to open any folder on a remote server, as well as edit and execute the code stored there, all from your local machine. You can find the installation for the extension pack here and a tutorial on how to use it to connect to a remote server here.

5 Exercises with `grep`

Download the book *The Adventures of Sherlock Holmes by Arthur Conan Doyle* from Project Gutenberg, e.g., using:

```
wget http://www.gutenberg.org/cache/epub/1661/pg1661.txt
```

and use the Linux utility `grep` to solve the following exercises. Use `man grep` to find out more options about the command line tool, which generally works as follows:

```
grep OPTIONS PATTERN FILES
```

For example:

```
grep start pg1661.txt
grep "start" pg1661.txt
```

Note: Since your search term (`PATTERN`) does not contain any spaces or special characters, omitting the quotes works here; it is generally safer to use quotes though.

- Search for lines that contain the word "miss".
- Make sure you include both spelling options (lower and uppercase). Which option of `grep` can you use for that?
- What happens if you add the option `-w`?

6 Gluing Commands Together with the Pipe

The real power of Linux command line tools comes from *combining* commands with the pipe (`|`), i.e., the *output* of the former command is forwarded as *input* to the next. For example, this is handy if we want to quickly count matches in a text. For this, we forward the output of `grep` and use the handy command `wc`, like this:

```
grep -iw quick pg1661.txt | wc -l
```

With this command we can count how many *lines* of the file contain the word "quick" (make sure you understand why we use both options `-i` and `-w`, or in short `-iw`). However, look at the output of the grep search without *piping* it to `wc`. What if we were to count all *occurences* of the word "quick"?

A solution to this is to ask grep to *extract* all matches. We can do so with the option *-o*. Try it out to solve the following questions:

- On how many lines of the file `pg1661.txt` does the word "quick" appear?
- How many *times* does the word "quick" appear in the book?
- Use `grep` to extract all words that start with an uppercase letter and save them in a file. **Hint**: To store the output of a command in a file, we use the > symbol (means: redirect to file), e.g., `grep PATTERN FILE > output.txt`.

7 More Advanced Usage of Unix Tools: Creating a word frequency list, finding function words

Let us now create a simple word frequency list from the book above using Unix tools to answer the following question: Which four *function words* are the most frequent in *The Adventures of Sherlock Holmes by Arthur Conan Doyle*?

- The first step is to split the text into separate words. Here, we will use the command `sed` to replace all spaces with a newline:

  ```
  sed 's/ /\n/g' FILE
  ```

  Note: Remember the flag `g`, which stands for *global*, replaces *all* occurrences of a space on a line. As you will see in the next exercise, this is a very crude way of tokenization.

- **Hint**: It is handy to forward this command to a tool called `less`, which lets you browse through the result (type 'q' to quit).

  ```
  sed 's/ /\n/g' FILE | less
  ```

- Now we can sort the list of tokens and count unique words:

  ```
  sed 's/ /\n/g' FILE  | sort | uniq -c
  ```

- To create the most frequent words first, sort again in reverse numeric order (find the options of `sort` to do so, e.g. check `man sort`).

Note: Here we used `sed`, our textbook shows an alternative with `tr` instead.

8 Copying Files to and from the Server

The Umbrella corporation has recently found out that the access card belonging to one of their employees had been copied and the intruder had been using the card to access the main building after work hours. Given the large stock of Unobtanium stored in the building, the intruder must be neutralized as soon as possible. The access card belongs to the employee Lily Cameron whose username is `lily.cameron`. The director of your department has assigned you the task of drilling through the access records and finding the breach entries.

The building access log is stored in the file `building access.txt`, which can be found on LearnIT. Each line in the file describes what access card was used to open a door in the building. Every time a card is used after work hours, the entry is marked with a `*WARNING*`.

1. Download the log file `building access.txt` from LearnIT and copy it to your home directory on the HPC login node using the command `scp` (use `man scp` to learn more about this command).
2. Find when Lily Cameron's stolen access card was used after work hours and save the results in chronological order in the file `lily_cameron.txt`.
3. Use `scp` to copy the file `lily_cameron.txt` from the server to your own computer.

9 Submitting a Job to the HPC Cluster

In this exercise, we are going to run a short test job on one of the computing nodes of the HPC cluster. You are supposed to run the command `nvidia-smi`, which provides monitoring and management capabilities for common NVIDIA GPU's (see also `man nvidia-smi`). The output of `nvidia-smi` commonly looks like:

```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 430.50       Driver Version: 430.50       CUDA Version: 10.1     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  GeForce GTX 108...  Off  | 00000000:03:00.0 Off |                  N/A |
| 20%   44C    P2    84W / 250W |   6383MiB / 11178MiB |     44%      Default |
+-------------------------------+----------------------+----------------------+
|   1  GeForce GTX 108...  Off  | 00000000:82:00.0 Off |                  N/A |
| 20%   44C    P2   108W / 250W |   6371MiB / 11178MiB |     53%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|    0     16580      C   python3                                    6373MiB |
|    1     17012      C   python3                                    6361MiB |
+-----------------------------------------------------------------------------+
```

This output shows that there are two GPU's available, and they are both currently used by one python3 process each. On the top of the output, the driver version is shown (in this case 430.50). Report the driver version that is found on the compute nodes. To do this you have to write a job-script, submit it to the cluster, and read its output. See the lecture slides for more information.

References

[1] Jurfasky and Martin, In Preparation. *Speech and Language Processing (3rd ed. draft)*. Available at https://web.stanford.edu/~jurafsky/slp3/