Natural Language Processing and Deep Learning
Lab 0 (pre-class homework)

Barbara Plank

March 21, 2019

After this exercise you should be:

- familiar with the concept of regular expressions
- able to use the Unix command line tools for search (`grep`), count (`wc`) and basic text processing (`sed` for substitution) including the concept of the pipe (`|`), e.g., to count word types or extract simple frequency word list
- able to discuss issues that arise with tokenization and implement them in `Python`

Requested reading: Chapter 2 of [1] (up to and including 2.4.2)

1 Pen and paper exercise

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

Note: By "**word**", following [1], we mean any alphabetic string separated from other words by whitespace, any relevant punctuation, line breaks, and so forth. 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 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`?

3 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`.

4 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.

5 Tokenization (Python notebook)

Solve the tokenization exercises provided in the tokenization notebook.

References

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