Natural Language Processing

Lab 1: Whitespace Tokenization

---

1 Introduction

In many languages, word boundaries are marked by spaces. However, relying on whitespace alone seldom gives an adequate tokenization. In this lab, you will inspect the output of a tokenizer that only segments tokens separated by whitespace, discuss the shortcomings of this approach, and start to think about how they can be overcome. In the process, you will get acquainted with the Python programming language (if you haven't used it before). All the files needed for the lab are in `/local/kurs/nlp/basic1/`. Start by copying them into your home directory:

```
cd
mkdir NLP_Lab_1
cp -R /local/kurs/nlp/basic1/* NLP_Lab_1
cd NLP_Lab_1
```

2 Tokenize some text

The file `tokenizer0.py` contains the simplest possible tokenizer:

```
import re, sys

for line in sys.stdin:
    for token in re.split("\s+", line.strip()):
        print(token)
```

The first line imports the `re` module (for regular expressions) and the `sys` module (for reading from stdin). The main program consists of two nested `for` loops: For each line in the input file, split the line into tokens at (one or more) whitespace characters (`\s+`), and print each token. To run the tokenizer on the file `dev1-raw.txt` and print the result to the (new) file `dev1-tok.txt`, run the following command:

```
python tokenizer0.py < dev1-raw.txt > dev1-tok.txt
```

Check that `dev1-tok.txt` contains a sequence of tokens, one on each line, and you're good to go.

3 Analyze the result

Go through the output file and try to identify problems in the tokenization. Use your intuition about what units are convenient for further linguistic analysis and try to find arguments for why a particular segmentation may lead to problems, for example, when we perform a grammatical analysis or try to look up words in a lexicon. (We are going to make this more precise next time, but it is useful for you to create your own opinion first.) Most problems you encounter will be either cases of *oversplitting*, where the tokenizer has split what should have been one token into several tokens, or *undersplitting*, where the tokenizer has failed to split a string into multiple tokens. Try to categorize different cases of oversplitting and undersplitting and think about how they could be eliminated.

4 Correct the tokenization

When you have identified the tokenization errors, you should correct them manually using a text editor. You should correct at least the first ten lines of the text in `dev1-raw.txt` (ending just before the word "Pacific"). Compare with some of your classmates: do you agree on how to tokenize the text?