

1. Exercise Sheet

Statistical Methods in Natural Language Processing

The solutions to the problems may be submitted until **Tuesday, April 28, 2015** before the exercise lesson. The condition for participating at the exam for “Statistical Methods in Natural Language Processing” is the successful solution of 50% of the problems and the presentation of the solutions of at least **two** problems in the exercise lessons.

The solutions to the problems can be submitted in groups of up to **two** students.

For programming exercises:

- The implementation has to be done in C/C++, but you can use additional standard Unix tools (tr, sed, awk...) for the preprocessing.
- Implementations have to be uploaded to L2P as well as printed out and submitted before the exercise lesson.
- Upload the programs as a .tgz or a .zip compressed directory to L2P. It must include a **Makefile** and must compile on Linux using simply the **make** command.
- Include a *short* description of the main data structures and algorithms used in your solution sheet.

1. [1 Point] Show that the following definitions of stochastic independence are equivalent:

$$\forall a, b \quad p(a|b) = p(a) \quad \text{or} \quad p(b|a) = p(b) \quad \text{or} \quad p(a, b) = p(a)p(b)$$

2. [2 Points] Suppose we wish to calculate $p(h|e_1, e_2)$, and we know following quantities:

- (i) $p(e_1, e_2), p(h), p(e_1|h), p(e_2|h)$
- (ii) $p(e_1, e_2), p(h), p(e_1, e_2|h)$
- (iii) $p(h), p(e_1|h), p(e_2|h)$

Which of these sets of quantities are sufficient for the calculation...

- a) ... if no additional independence information is given?
 - b) ... if we know that $p(e_1|h, e_2) = p(e_1|h)$?
3. [3 Points] Let X and Y be two independent random variables with Poisson distributions with parameters λ_X and λ_Y . Show that the sum $X + Y$ is Poisson distributed with parameter $\lambda_X + \lambda_Y$.

(**Reminder:** Poisson distribution with parameter λ : $p_\lambda(x) = \frac{e^{-\lambda}\lambda^x}{x!}$ for $x \in \mathbb{N}_0$)

4. [3 Points] Implement a dictionary, i.e. a data structure for indexing words. The reverse access direction should also be included, i.e. given a word it is possible to get its index, and given an index the corresponding word can also be retrieved.

Use this data structure to implement a program which reads a text, selects the n most frequent words, and writes the text again with the other words substituted with the special <UNK> token (for “unknown”). Do this in an *efficient* way, especially:

- Read the text from disk *only once*.
- Do not store the text as words, use the dictionary.

Do this in a modular way, so you can reuse your implementation for future problems. In particular the dictionary and the routines for reading texts and splitting it into words will be used in further exercises.

You can try it on the text “Alice’s adventures in Wonderland” available from our website. Do not take case information into account and treat punctuation marks (‘ ’ _ , ; : ! ? . " () [] *) as separate words. (**Hint:** This preprocessing is best done as a separate step using scripts.)

An example Makefile is shown in the appendix. You can use this example as a template for your own implementation.

Appendix

Makefile

```
1  # Sample Makefile for the first exercise of the course "Statistical
2  # Methods in Natural Language Processing"
3  #
4  # Author:
5  # David Vilar, Lehrstuhl für Informatik 6
6
7  # Compiler binary
8  CXX=g++
9  # Compiler options
10 CXXFLAGS=-Wall -O2
11
12 # The first target will be the default when make is called without arguments
13 all: exercise01.4
14
15 exercise01.4: dictionary.o exercise01.4.cc
16 dictionary.o: dictionary.cc dictionary.hh
17
18 # These are only for convenience and are not strictly necessary
19 .PHONY: clean veryclean
20 clean:
21     ↪      rm -f *.o
22
23 veryclean: clean
24     ↪      rm -f *~
```