Universität Bielefeld Semantic Computing Group

Exercise Sheet

Statistical Natural Language Processing

Exercise Sheet 2

Due Date: November 28, 8 pm

Note on Submission

All solutions have to be uploaded together as a single zip file to LernraumPlus. Solve the exercises by implementing the functions in the file `exercise_sheet2.py`.

Exercise 1 [5 points]

In this exercise sheet we will solve the task of named-entity recognition by using HMMs. The file `corpus_ner.txt.zip` contains a corpus annotated in the IOB schema. Each line in this file contains two columns separated by an ASCII space character, the first column contains the token and the second one the label of the token. Sentences are separated by empty lines. The set of states S contains all labels contained in the provided corpus and the set of emission symbols K contains all tokens contained in the corpus. Add the special token `<unknown>` for unknown tokens to the set of all tokens. Treat tokens occurring only once in the corpus as unknown, i.e., replace them by the token `<unknown>`. You may use the Python function `import_corpus` for importing the corpus.

a) Implement the Python functions `initial_state_probabilities`, `transition_probabilities` and `emission_probabilities` representing the parameters $P_\pi(\cdot)$, P_A and P_B, respectively, of an HMM. Each of these functions takes one argument (among others) named `internal_representation`. This argument is a data structure of an internal representation of the parameters $P_\pi(\cdot)$, P_A and P_B, respectively. It is up to you which data structure you use. For example, you can use a Python dictionary for representing the parameter $P_\pi(\cdot)$. We denote the data structures for the parameters $P_\pi(\cdot)$, P_A and P_B by D_π, D_A and D_B, respectively.

Next, implement the Python functions `estimate_initial_state_probabilities`, `estimate_transition_probabilities` and `estimate_emission_probabilities` for learning the parameters of the data structures D_π, D_A and D_B, respectively.

See `exercise_sheet2.py` for further information.

P. Cimiano · C. Witte

Exercise 2 [15 points]

Implement the Viterbi algorithm by defining the Python function `most_likely_state_sequence`. Instead of maximizing $P(X|O,\mu)$, maximize $\log P(X|O,\mu)$, i.e., replace multiplication by summation and probabilities by log probabilities. Make use of the Python functions you implemented in exercise 1.