GWV – Grundlagen der Wissensverarbeitung

Tutorial 10 : Hidden Markov Models

Class Exercise 10.1 : (HMM decoding)

Suppose you have some magical device which can tell you on which street you are based on your surrounding (the Geolocation based on Photographs System). Our world consists of two streets (A-Street and B-Street), as depicted below. The GPS takes photos of the surrounding and guesses which street this photo comes from.

The sensor has the following characteristics: On A-Street and B-Street, 80% of the time it will yield the correct street. Otherwise it will yield the parallel street.

Use the forward-, forward-backward-, and viterbi algorithm to decode where someone has been at each step for the following observations:

A-Street, B-Street, B-Street

You know that the person could only have stayed in the same street. The person either started in A-Street or B-Street, but you don't know in which one.

"A-Street"

"B-Street"

Exercise 10.2: (A simple PoS-Tagger)

A Part-of-Speech(PoS) tagger is a program that assigns word classes (i.e. PoS) to words of a text. For example, for the input

> He left the book at the library.
>
> It was in the left room.

the corresponding output should be the following[1]:

> He\PRP left\VERB the\DET book\NOUN at\ADP the\DET library\NOUN.
>
> It\PRP was\VERB in\ADP the\DET left\ADJ room\NOUN.

As always, you can find the relevant data at our wiki. The data we use will be one word per line, with each line consisting of word [tab] PoS-tag.

Let

$Word_i$ be the variable denoting the ith Word

Tag_i be the variable denoting the PoS-tag for the ith word

Implement a hidden markov model where the probability distribution of $Word_i$ only depends on the state of Tag_i and Tag_i only depends on Tag_{i-1}. The tagger model (i.e. the transition, emission and prior probabilies) should be trained from the file provided in the wiki. (4 Pt.)

Create a function that takes a list of words (possibly from the command line) and uses filtering to produce a corresponding list of PoS tags. (2 Pt.)

Make sure that your tagger can cope with input that includes words that are not in the training data. (2 Pt.)

Implement either the forward-backward algorithm to output several PoS tags per word or the viterbi algorithm to output the most probable tag sequence. (4 Pt.)

Hint: You can assess how good your tagger is by tagging the test file and comparing the output produced by the tagger with the original file, e.g. with
`diff -u original_file generated_file | grep '^+' | wc -l`

Provide example output where applicable! See also the ***Formal Requirements for Programming Assignments*** *from Tutorial 3.*

Version: December 13, 2019
Achievable score on this sheet: 12

[1] The tags for this example do not correspond to the ones in the data file (because the data we work on is German). The German PoS tags are explained here: `https://www.ims.uni-stuttgart.de/forschung/ressourcen/lexika/germantagsets/#id-cfcbf0a7-0`