Text segmentation using Hidden Markov Models [1]

Expected Lab report

The report of the TP is to be deposited on the educational site (rubric "Rendus TP HMM "). It will consist on a notebook (versions .pynb **and** .pdf) including both :

— the answers to the questions,

— the code and the results

— a discussion on the method and on the results obtained,

— and all what you think would be useful from a scientific point of view.

Task : automatic segmentation of mails, problem statement

This Lab aims to build an email segmentation tool, dedicated to separate the email header from its body. It is proposed to perform this task by learning a HMM (A, B, π) with two states, one (state 1) for the header, the other (state 2) for the body. In this model, it is assumed that each mail actually contains a header : the decoding necessarily begins in the state 1.

— **Q1 : Give the value of the π vector of the initial probabilities**

Knowing that each mail contains exactly one header and one body, each mail follows once the transition from 1 to 2. The transition matrix $(A(i,j) = P(j|i))$ estimated on a labeled small corpus has thus the following form :

$$A = \begin{pmatrix} 0.999218078035812 & 0.000781921964187974 \\ 0 & 1 \end{pmatrix}$$

— **Q2 : What is the probability to move from state 1 to state 2 ? What is the probability to remain in state 2 ? What is the lower/higher probability ? Try to explain why**

A mail is represented by a sequence of characters. Let N be the number of different characters. Each part of the mail is characterized by a discrete probability distribution on the characters $P(c|s)$, with $s = 1$ or $s = 2$.

— **Q3 : What is the size of B ?**

Material

Coding/decoding mails

Emails are represented as ASCII character vectors.

In dat.zip, mail.txt can be transformed into a vector of numbers (between 0 and 255) (one vector per line) in a text ;

Files of the form `dat/*.dat` contain the already encoded versions of the corresponding mails. The list is in `mail.lst`.

Use the command `np.loadtxt` to load the dat files.

1. TP d'origine défini par François Yvon http://perso.limsi.fr/yvon/mysite/mysite.php?n=Main.HomePage

Distribution files

For the first part of the Lab, we work with the distributions that are provided in the P.text file.

Each of the columns of this file contains the distribution of the probabilities of occurrence of each character. of the ASCII codes respectively in the header and in the body. These distributions were learned on a small corpus labeled with 10 emails ; there are obvious differences, especially in areas where ASCII codes correspond to alphabetic characters, as you can see by viewing these distributions.

To implement

All the work is to be done under Python.

— implement the Viterbi algorithm. Concretely, it comes to coding a function which takes as argument a vector of observations and the parameters of the model, and returns a vector of states representing the most probable sequence.

— test it on some mails that are given in the dat directory (especially `mail11.txt` to `mail30.txt`).

Visualizing segmentation

Finally, the utility `segment.pl` allows to visualize a segmentation produced by your segmenter in the form of the best path found by the Viterbi algorithm (in a vector of 1 and 2). It produces a file `path.txt` where the segmentation is visualized. It calls `coder.pl` that encodes the mail in ascii. To use it :

```
perl segment.pl mail.txt path.txt
```

```
"===== cut here".
```

— **Q4** : print the track and present and discuss the results obtained on `mail11.txt` to `mail30.txt`

Further questions

— **Q5** : How would you model the problem if you had to segment the mails in more than two parts (for example : header, body, signature) ?

— **Q6** : How would you model the problem of separating the portions of mail included, knowing that they always start with the character ">".

BONUS : unsupervised learning

From this point, the answers are not requeted for the lab evaluation. Nevertheless, students that answer to this question could collect bonus point.

We now propose to implement the necessary bricks for the unsupervised learning of the parameters of the model.

1. Program the alpha calculation. You can, if you feel the need, use the logsum function, which calculates sums of log. If $x = log(y)$ and $x' = log(y')$, $logsum(x, x') = log(exp(x) + exp(x'))$.
2. Program the beta calculation. It is recalled that $beta(t, i) = P(X_{t+1}...X_T|S_t = i)$.
3. Using the reestimation formulas studied in progress, complete programming a step of the EM algorithm. This requires calculating the gamma vector ($gamma(t, i)P(S_t = i|X_1...X_T)$), then using this vector to update matrices A and P.

Start learning by initializing the parameters randomly and then initializing with distributions of question 1.

Q7Bonus : Present the pseudo code of the algorithm and discuss the results.

It is actually possible to do an "approximate" learning by replacing the calculation of alphas / betas with a call to viterbi. The gamma vector of the probabilities of state occupation is in this case deterministic. Without taking account this difference, the updating of the parameters remains the same. Implement this variant and compare the results obtained with the previous one.

Q8 bonus : Present the pseudo code of the algorithm and discuss the results.