SNLP 2016
Exercise 9

Submission date: 3.07.2016, 23:59

Part-of-Speech Tagging

In this exercise you will experiment with the problem of part-of-speech tagging using Hidden Markov Models (HMMs).
For the needs of this exercise you are given a part of the Brown corpus split into 3 parts, namely *training*, *development* and *test* set. For your ease a framework has been provided, which you may feel free to use or come up with your own implementation.

1. (4 points) In the following you will use the training set to learn (by counting) the parameters of a HMM model. The training phase should be performed once per training corpus and provide all data necessary to later (during prediction) compute probability estimates.

   (a) (1 point) Implement a function *count_transitions* to count tag-to-tag transitions. This function will be invoked once on the training data set and compute all data necessary to later compute transition probabilities for use in a bigram[1] tagger, to compute Maximum Likelihood estimates.

   (b) (1 point) Implement a function *count_emissions* to count tag-to-word emissions $C(w_i, t_i)$ and total tag counts $C(t_i)$ for each combination of word and POS-tag that occurs in the training corpus. Likewise, this function should be called once per corpus.

   (c) (1 point) Describe an appropriate way to compute the initial state probabilities. Now implement it.

   (d) (1 point) Implement a function *get_transition_probability* to compute Maximum Likelihood Estimates for the transition probabilities. Since this function may be called many times during prediction, you may use as an argument the data you computed in part (a). Remember that the Maximum Likelihood Estimate is computed by

   $$P(t_i|t_{i-1}) = \frac{C(t_i, t_{i-1})}{C(t_{i-1})} \ , \tag{1}$$

   where $C(t_i, t_{i-q})$ is the number of times that a tag t_i follows the tag t_{i-1} in the training set.

2. (6 points) Your goal is to tag the *test* set and in this attempt an important issue that arises is the treatment of Out-Of-Vocabulary[2] (OOV) words. One naive approach to tackle this problem is to use smoothing[3].

   (a) (1 point) *Non-Framework users* Implement a function that uses the counts you computed in part (b) of question 1 and computes the Lidstone[4] Smoothed estimate

   $$P(w_i|t_i) = \frac{C(w_i, t_i) + \epsilon}{C(t_i) + \epsilon V} \ , \tag{2}$$

[1] This makes the first order Markov Assumptions: $P(t_i|t_1, \ldots, t_{i-1}) \approx P(t_i|t_{i-1})$, as in Slide 33 of the Lecture Notes.

[2] Those are the words that appear in the test set for the first time; therefore no emission information can explicitly be drawn from the training set.

[3] Practically we modify the ML Estimate by assigning some probability mass to the unseen tokens, to avoid the "steep" zeros.

[4] The Lidstone smoothing is also known as *add-ϵ* smoothing due to the characteristic Equation (2). More information on the intuition behind it can be found in the relevant wiki page.

where V is the number of distinct tokens[5] in the union of the test and train set, and ϵ the smoothing parameter to be specified.
Note that the estimate is only computable after the parameters ϵ,V are specified, which happens in consecutive stages and is independent of the counting phase of Question .

Framework users

In your framework the function *count_emissions* has been implemented as per Equation (2). To get some intuition of the Lidstone smoothing method, show[6] that Equation (2) can be rewritten as

$$\frac{C(w_i, t_i) + \epsilon}{C(t_i) + \epsilon V} = \mu \frac{C(w_i, t_i)}{C(t_i)}) + (1 - \mu)\frac{1}{V} , \tag{3}$$

for some μ with $0 \leq \mu \leq 1$.
Can you now interpret each fraction in Equation (3)?

(b) (2 points) Use the provided framework (or any other implementation) of the HMM Viterbi algorithm, to predict labels for the test set, using your probability estimates.
To be able to compute the emission probabilities you will need to specify an ϵ parameter and you are welcome to experiment with different values, e.g: 1, 10^{-4}, 10^{-5}, 0, etc. Report missclassification rate using the function `evaluate`[7] of the framework.

(c) (1 point) Propose a way to find an appropriate value value for the ϵ parameter.
Hint: You may and should use the *development* set.

(d) (2 points) Implement your idea of part (c) from this question. Report the optimal ϵ^* parameter you computed.

Bonus (2 points)

The Piosson distribution is important for many processes in daily life (`https://en.wikipedia.org/wiki/Poisson_distribution`).

(a) (1 point) Calculate mean for the Piosson distribution.

(b) (1 point) Now calculate the standard deviation.

1 Submission Instructions: Read carefully

- You can form groups of maximum 3 people.
- Submit only 1 archive file in the ZIP format with name containing the MN of all the team members, e.g.:

Exercise_01_MatriculationNumber1_MatriculationNumber2_MatriculationNumber3.zip

- Provide in the archive:
  - your code, accompanied with sufficient comments,
  - a PDF report with answers, solutions, plots and brief instructions on executing your code,
  - a README file with the group member names, matriculation numbers and emails,
  - Data necessary to reproduce your results [8]
- The subject of your submission mail must contain the string [SNLP] (including the braces) and explicitly denoting that it is an exercise submission, e.g:

[5] i.e. the number of different words, ignoring multiple occurrences, in both the test and the training set.
[6] If you want to avoid writing the full derivation, it suffices to report the formula for μ and show that $0 \leq \mu \leq 1$.
[7] If you want to create your own, you simply have to ignore the successful full-stop ('.') detections.
[8] If you feel that these files are beyond reasonable size for an email submission and also reasonably convenient, please provide a means for us to access them online

[SNLP] Exercise Submission 09

- Depending on your tutorial group, send your assignment to the corresponding tutor:
  - Sedigheh Eslami: *eslami@mpi-inf.mpg.de*
  - Naszdi Kata: *b.naszadi@gmail.com*
  - Stephanie Lund: *stflund@gmail.com*

2 General Information

- In your mails to us regarding the tutorial please add the tag [SNLP] in the subject accompanied by an appropriate subject briefly describing the contents.
- Feel free to use any programming language of your liking. However we strongly advise in favor of Python, due to the abundance of available tools (also note that Python3 comes with an excellent native support of UTF8 strings).
- Avoid using libraries that solve what we ask you to do (unless otherwise noted).
- Avoid building complex systems. The exercises are simple enough.
- Do not include any executable files in your submission, as this may cause the e-mail server to reject it.
- **In case of copying, all the participants (including the original solution) will get 0 points for the whole assignment. Note: it is rather easy to identify a copied solution. Plagiarism is also not tolerated.**
- **Missing the deadline even for a few minutes, will result in 50% point reduction. Submission past the next tutorial, is not corrected, as the solutions will already be discussed.**
- **Please submit in your solutions necessary to support your claims. Failure to do so, might results in reduction of points in the relevant questions.**
- Each assignment has 10 points and perhaps some bonus points (usually 2 or 3). In order to qualify for the exams, you need to have 2/3 of the total points. For example, in case there are 12 assignments, you need to collect at least 80 out of the 120 points to be eligible for the exams. A person that gets 10 plus 2 bonus points in every exercise, needs to deliver only 7 assignments in order to be eligible for the exams, since 7*12=84.
- Attending the tutorial gives 2 points increase for the corresponding assignment.
- Exercise points (including any bonuses) guarantee only the admittance to the exam, however have no further effect on the final exam grade.