SNLP 2017
Exercise 3

Submission date: 19.05.2016, 23:59

Entropy

1) (5 points) For this exercise, you should select three texts, from a source such as NLTK or Project Gutenberg. Choose two in the same language (for example, English), and one in a different language (for example, German). Pre-process (lowercase, remove puncutation) and tokenize the text. You may use any functions from NLTK for this task.

- (2 points) Implement a function that takes two texts as arguments, and calculates the KL divergence $D(p||q)$ between them. For this task, you will need to use the probabilities after applying Lidstone smoothing. Calculate and report the KL divergence between the two texts of the same language, and between two of the texts in different languages. Comment on any difference in the results. (consider only unigram distribution, using the maximum likelihood estimation with Lidstone smoothing*)
- (1 point) In Exercise 2 we have proved $E[-logP(X)]-E[-logP(X|Y)] = E[-logP(Y)]-E[-logP(Y|X)]$. This value is defined as $I(X,Y)$. $I(X,Y)$ can be defined as KL-divergence of two events. Find these two events and prove it.
- (2 points) Provide an intuitive explanation of what $E[-logP(X|Y)]$ and $I(X,Y)$ measures respectively. For each of the three texts, select two pairs of words, one with a high $I(X,Y)$ and the other with a low $I(X,Y)$. Calculate $I(X,Y)$ for all the word pairs. Compute $P(X|Y)$ with bigram frequency.

*Smoothing is a technique used in language modeling to account for words that are missing from the training sample, so that they don't have a probability of 0. It works by taking a small amount of probability mass from other words, and giving them to new, unseen words. The formula for one method, Lidstone smoothing, is given below:

$$P_{lidstone}(w) = \frac{count(w) + \alpha}{N + \alpha V} \tag{1}$$

where N is the total number of tokens, and V the size of the vocabulary. We'll use $\alpha = 0.1$ for this example.

Text Compression

2) (5 points) Suppose the event X with probability $X = i$ as p_i, $i = 1, 2, ..., m$. Let li be the number of bits used to encode event $X = i$. We assume there is an associated cost c_i per bit when encoding event X_i. The average cost C of encoding event X would be $C = \sum_{i=1}^{m} p_i c_i l_i$

- (2 points) Devise a binary code that would minimize C and derive the minimum value C^*. Ignore any implied integer constraints on l_i. Use the theorems and conclusions drawn in the slide Chapter 4.
- (1 point) Now consider the integer constraint. Let $i = 5$ and $p_i = \frac{1}{5}$ for every i. $c_i = 5, c_2 = c_3 = 3, c_4 = c_5 = 2$. Use Huffman code procedure (explained here: `https://en.wikipedia.org/wiki/Huffman_coding#Informal_description`) to minimize C. Find the corresponding Huffman code for every event.

- (2 points) Following the previous setting, let $C_i = 3$ for every i, prove the Huffman code you draw in the last question is also optimal in this condition.

1 Submission Instructions: Read carefully

- You can form groups of maximum 2 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.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 [1]
- 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:

[SNLP] Exercise Submission 01

- Depending on your tutorial group, send your assignment to the corresponding tutor:
  - David Adelani: *s8daadel@stud.uni-saarland.de*
  - Rajarshi Biswas: *rbisw17@gmail.com*
  - Xiaoyu Shen: *s8xishen@stud.uni-saarland.de*

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 will result in 0 point for the whole assignment. So make sure to submit on time.**

[1] 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

- **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 1 point increase for the corresponding assignment.
- Presenting a solution will give bonus equal to the point of the presented question (Up to 4 poins). The bonus is divided evenly in the group member. Each group can present up to 3 times.
- Exercise points (including any bonuses) guarantee only the admittance to the exam, however have no further effect on the final exam grade.