Advanced Information Retrieval

Assignment #1

August 21, 2017

This assignment consists of two parts. In **Part 1**, you will practice calculating the entropy based on empirical probabilities estimated from a training corpus. In **Part 2**, you will implement techniques for automatic text categorization. **Part 1** is accompanied by the code template that familiarizes you with the general workflow of computing with Python. In **Part 2**, you are expected to single-handedly implement methods required for data preprocessing, text categorization, and evaluation.

You are provided with the data set of resumes (CV). Each CV is stored as a PDF. In the folder `data` you will find two subfolders. The folder `LinkedIn` stores standardized resumes that have a distinct and clear structure. The folder `other` contains resumes that were designed in a non-standardized way. You are going to use these files to finish both parts of this assignment.

Summarize the results of this assignment into a **report** and submit it together with your code through Moodle before the due date. **Late submissions receive the score of 0**.

Part 1

This part of the assignment is accompanied by a code template written in Python. Make sure you have required libraries installed. The main file is `assignment1.py` that executes entirely only when all auxiliary functions are implemented. To finish this part, you need to modify the following files:

`q2_entropy.py`
`q3_conditional_entropy.py`
`q4_joint_entropy.py`
`q5_mutual_information.py`
`q6_min_max_entropy.py`

To finish this part you need:

1. Study the way probabilities of terms are estimated from the data set. The vocabulary and the probabilities are obtained by calculating frequencies of terms. Use the file `q1_create_n_gramm.py` as a reference. Understand how the n_gram data structure is organized.

2. Implement the function `entropy()` to calculate the entropy of a random variable (RV) based on a sample from its distribution. Assume that the structure that stored empirical probabilities is a dictionary with terms as keys.

3. Implement the function `conditional_entropy()` to calculate the conditional entropy of a 2-word phrase 'X Y' $\mathbf{H}(Y|X)$

4. Implement the function `joint_entropy()` to calculate the joint entropy of two words in a 2-word phrase 'X Y' $\mathbf{H}(Y|X)$

5. Implement the function `mutual_information()` to calculate mutual information of words in 2-word phrase 'X Y' $\mathbf{I}(X, Y)$

6. Implement the function to compute minimum and maximum theoretical value of terms entropy based on the vocabulary obtained from the corpus

7. Run the file `assignment1.py` to receive the values for CV data set using the functions implemented earlier. Place the results into your report.

8. What is better, high or low entropy value? Why?

9. You have noticed that the calculated entropy is less than the maximum entropy. Why?

10. Based on CV data set, what is larger: entropy or conditional entropy? Can it be otherwise? Why?

11. What are the upper and lower bounds for mutual information? Explain intuition behind these bounds. What does the value of mutual information you have obtained mean? Can you relate this to bits that are required to encode one term?

Part 2

In this part, you are asked to write your own code and implement the algorithm for classifying segments of a document into different categories. Consider a resume from the provided LinkedIn data set. It has three distinct sections:

1. Summary
2. Work Experience
3. Education

You need to implement the algorithm that parses a particular resume and labels segments of the document with different categories using probabilistic (Bayesian) approach.

In the folder `./data/LinkedIn/` you will find the script `extract.py` that allows you to easily split the LinkedIn resumes into sections. This split is not perfect and may cause errors. For better result think of other ways for preparing training data. Training data is to be used for calculating conditional probabilities of terms. Let s denote the type of section (e.g. $s \in \{workexperience, education, summary\}$). You can estimate the conditional probabilities $p(x_i|s)$ by counting terms in different sections. This gives you the opportunity to categorize segments of a particular resume into one of the section types. One of the simplest approaches is to use Maximum Likelihood Estimator (MLE)

$$\hat{s} = \text{argmax}_s \left(p(s|\mathbf{x})\right) = \text{argmax}_s \left(\frac{\prod_{i=1}^{N} p(s, x_i)}{\prod_{i=1}^{N} p(x_i)} \right) \tag{1}$$

where $\mathbf{x} \in \mathbf{R}^N, N$ =size of vocabulary, $x_i \in \{0, 1\}$ indicates whether the term i is present in the segment to label. The term occurrences are considered to be independent. You can find more details about Bayesian classification in the literature. Hint: to accelerate calculations consider maximizing $\log\left(p(s|\mathbf{x})\right)$.

To complete the current part of the assignment you need:

1. Split LinkedIn resumes into training and testing data set. Process the document of the training data set into a form suitable for further work, e.g. categorize the parts of the resumes for estimating conditional probabilities.
2. Choose how you are going to score the quality of your classification algorithm. Recall what is **Precision, Recall, F1-score**. Understand how

you are going to label sections in resumes (by sentences, by lines, by word groups). Which scores are the most suitable for this task given this data set and your labeling scheme? Prepare the testing data set for calculating your performance score. Approach this task very carefully.

3. Implement the labeling algorithm described in (1). Give a thought to the best data structure to store the probabilities of terms (dictionary, table, etc.). Take the usability of your data structure into consideration. The highest performance is achieved with vectorized solution, but does it achieve good memory usage? Employ your testing data set to compare estimated label and the true label. Record your best score.

4. Test your algorithm on several resumes from the folder `other`. You **do not** need to calculate the performance score for this task. How well does your algorithm work when presented with resumes of different structure? How can you improve the performance in this case?

5. Do you know any other IR methods to perform the labeling of document sections? What are their advantages/disadvantages?

6. (Optional) Implement another labeling algorithm using another IR method (the one you believe is better). It is better if your new method is compatible with your training/testing data. How does this new method score comparing to the previous one?