**CMPE 561 NATURAL LANGUAGE PROCESSING** 04.04.2017

###### **APPLICATION PROJECT #2**

In this project, you will implement an author identification system using both a generative model (Naïve Bayes) and a discriminative model (Support Vector Machine). You will use the “69 Authors” corpus that includes 910 articles (documents) from 69 different Turkish authors. The corpus is available on the course web site.

Divide the corpus into training and test sets “randomly” such that 60% of the articles will be in the training set and 40% will be in the test set. You should do this in such a way that 60%-40% ratio will be maintained for each author. (For instance, for Abbas Güçlü who has 15 articles, 9 of them will be assigned to training and 6 to test randomly.) Then you need to tokenize all the documents. The tokenizer must be your own implementation; a simple tokenizer will suffice.

**Part A. Generative Modeling**

The Naïve Bayes model for identifying the author of a given document will be as follows. For a given document *d*, the probability that the author is *a* is:

(1)

where *n* is the number of words in the document and *wi* denotes the ith word. We use the independence assumption, i.e. each word *wi* is independent of other words in the document. The author for which the above equation is maximized for the given document is identified as the author of the document.

values can be estimated as follows:

(2)

So, the overall process will be as follows:

1. Training: Calculate the word probabilities for each author using Eqn. (2). Use Laplace smoothing.
2. Testing: For each document in the test set, determine the predicted author using Eqn. (1). (You can assume that value is the same for each author, since we have a balanced data set.)

**Part B. Discriminative Modeling**

We will use the SVM (Support Vector Machine) learning method for author identification. We will use SVMmulticlass library that is available at <https://www.cs.cornell.edu/people/tj/svm_light/svm_multiclass.html>. Download the two files svm\_multiclass\_learn (for training) and svm\_multiclass\_classify (for test).

The process will be as follows:

1. Training: Execute the training program as

svm\_multiclass\_learn train\_file model\_file

where *train\_file* is input and *model\_file* is output. The system learns a model that best fits the training data and puts the learnt model into the model file.

1. Testing: Execute the test program as

svm\_multiclass\_classify test\_file model\_file output\_file

where *test\_file* and *model\_file* are input, and *output\_file* is output. The system applies the learnt model to the test data and writes the result into the output file.

Explanation of *train\_file* and *test\_file*: (Also explained on the web site) Each line corresponds to a document as follows:

author-id f1:v1 f2:v2 …

*author-id* (class-id) is the id of the author of the article. Give each author a unique id. *f1*, *f2*, … are the feature numbers and *v1*, *v2*, … are their weights. Each feature is a distinct token that appears in the document collection and each value is either 1 (that feature occurs in that document) or 0 (that feature does not occur in that document).

So, you should first form a vocabulary of all the tokens in the document collection and give an id to each token (feature).

For instance, suppose that a document that belongs to author 12 includes the tokens whose ids are 3, 47, 562, 3218 (and no other token). Then the line will be

12 3:1 47:1 562:1 3218:1

During training, the system learns which features are important (discriminative) for each class (author).

During testing, the system determines which author is most likely for a document (without using the author-id value at the beginning of the line). Then, by using the given author-id values, the system measures the success rate of its predictions.

**Part C. Evaluation**

Both for Part A and Part B, evaluate the system.

You will evaluate the results in terms of three metrics:

1. Accuracy: This is equal to the number of correct predictions divided by the number of documents. Calculate this for each author and for the entire test set.

For metrics (b) and (c), first do the following:

For each author A, form the following table:

|  |  |  |
| --- | --- | --- |
|  | True author is (A) | True author is (not A) |
| Predicted as author (A) | TP | FP |
| Predicted as author (not A) | FN | TN |

TP (True Positive): Number of documents whose author is A and predicted as A

FN (False Negative): Number of documents whose author is A but predicted as another author

FP (False Positive): Number of documents whose author is another author but predicted as A

First, determine TPi, FPi, FNi for each author *i*, 1≤i≤M (M is the number of authors). Then, calculate metrics (b) and (c) below.

(We will use precision (π), recall (ρ), and f-measure (F) metrics. Their general definitions are as follows: , , )

1. Micro-averaged Precision, Recall, F-measure:

(Micro-averaged F-measure gives equal weight to each document and is therefore considered as an average over all the document/class pairs.)

1. Macro-averaged Precision, Recall, F-measure:

(Macro-averaged F-measure gives equal weight to each class, regardless of number of documents in that class.)

**Part D. Bonus Part (Optional)**

For Part B, in addition to features formed of tokens (words), you can use more features that are discriminative in determining the author of an article. For instance, the length of an article, the average sentence length, the number of punctuation marks in an article, the number of “…”s (three dots) in an article (some authors like that style), and so on may be good features in this task.

Prepare a design and implementation document which clearly explains the system. Follow the structure given in <http://www.cmpe.boun.edu.tr/~gungort/informationstudents.htm> - Programming Project Documentation. Explain the modules, the data structures used, the logic of the algorithms, etc. Show how the system learns the model parameters, the training phase, and the executions on the test data. The document will be an important part of the project. The suggested size of the document is about 10-15 pages. Submit the document and the source code (as an appendix within the document) on 17.05.2017 (both as hard-copy and via e-mail).

You will do a demonstration of the project between 22.05.2017 and 26.05.2017 (which is subject to change due to the date of the final exam of the course). We will arrange for each group a date and hour for the demo.

*Notes:*

* The project will be done by groups of two students.
* Each deadline indicated above must be followed. There will be a grade reduction for each in the case of being late.
* ***The program codes must completely be written by you.***
* You can use any programming language you wish.