Homework 1

NLP/ML/Web - Fall 2013 - Andrew Rosenberg

Due Monday, March 9 at 11:59pm

Problem 1) Web Page Classification
Your task is to write a program that classifies web pages.

1a. Identify at least 5 categories for web documents. Include clear definitions of each of these. Possible categories include "author", "date published", "product type based on reviews", "politics vs. sports vs. etc.".

1b. Download at least 150 instances of each category.
Identify 100 as training, and 50 as testing. Generate labels for these documents these documents for each category. These labels may be generated by hand, or automatically (i.e. all web pages that contain "by Adam Platt" are categorized as "Adam Platt". NB: if you are doing something like this, make sure that you remove this identifier before the classification routine.)

Details:

- Store each of these webpages in a single directory.
- Create two plain text files one for training, train_labels.txt and one for testing, test_labels.txt.
- These text files should contain the filenames and associated label for all of the downloaded pages. One per line, separated by a comma.

E.g.

```
http:__cheezburger.com_7790385920,CAT
http:__www.dailypuppy.com_puppies_spud-the-labrador-retriever_2013-09-16,DOG
http:__en.wikipedia.org_wiki_Giant_panda,PANDA
```

1c. Write a training program that, given these web pages and labels produces a classifier.

You may use sklearn or weka or other classification tools. You may also write your own classifier.

Most important component here is that you write feature extraction routines to generate a **document representation**.

Details:

- This program should be called with (minimally) three command line parameters in this order: 1) the training label file, train_labels.txt, and 2) the data directory, something like '/home/andrew/nlpmlweb/hw1/data/' and 3) a filename for the generated classifier. If additional files need to be read, they should be included in the command line after these three. A description of these files must be included in the README.

1d. Write a testing program that evaluates a set of testing files using on a trained classifier.

This program should generate two outputs. 1. A measure (or measures) of overall classification performance. Accuracy is an excellent choice here. 2. A listing of predictions for each data point.

Details:

- This program should be called with (minimally) four command line parameters in this order: 1) the testing label file, test_labels.txt, and 2) the data directory, something like '/home/andrew/nlpmlweb/hw1/data/' and 3) a filename for the trained classifier, and 4) a filename to write predictions to with an indication of if it is correct or not. If additional files need to be read, they should be included in the command line after these four. A description of these files must be included in the README.
- The overall classification performance should be written to the command line.
- The format of the prediction file should be a comma separated value file containing, the file stem and true label (from the labels file), the predicted label, and an indicator to indicate whether it was correct or not ('+' for correct, '-' for incorrect).

E.g.

```
http:__cheezburger.com_7790385920,CAT,DOG,-
http:__www.dailypuppy.com_puppies_spud-the-labrador-retriever_2013-09-16,DOG,DOG,+
http:__en.wikipedia.org_wiki_Giant_panda,PANDA,DOG,-
```

1e. **Computer Science Oriented Requirement:** Use at least 3 document representations. Write your code such that introducing and selecting a new word representation is trivial. I.e. setting a configuration variable to an appropriate value.

1f. **Linguistics Oriented Requirement:** 1-3 pages: Write a response to word/document representation. You may respond to any quality of these you like. Linguistic underpinnings

and implications. Feasibility. Computational Complexity. Expected efficacy (i.e. which types of tasks would each be good at). Limitations and advantages.

1-2 pages: Describe an ideal word representation for your categorization task. Barring technical ability and time, are there any limitations to implementing this representation? Why is this better than any existing approach?

Deliverables

- All source code and libraries (or pointers to download) required for your project.
- A README/Report file whose contents are described below.
- A response to the Linguistics Oriented Requirement.
- All training and testing data.
- All required configuration files (including training_labels.txt and testing_labels.txt).

The README can be in any electronic format (txt, pdf, doc, google docs, open office) and should minimally include the following

- A list and description of every file included in the submission – including which matrix library (if any) you use.
- A description of how your code is compiled (if it is compiled)
- A description of how to run your code.
- A high level description of the task – here: A description of the data and the types of document representations you are exploring. This should be just a few paragraphs long, and doesn't necessarily need a lot of mathematical notation. It should be understandable to a reasonably informed reader.
- A report of the experiments. How do the document representations compare against each other. Is one better than the others? Do they have different strengths that are observable by its performance on particular files? Graphs may be helpful for this assignment
- Any other points of interest – running time, complexity, unique qualities of your implementation, etc.

Grading:

- **15 points - Compilation** Each file must compile without error or warning into an executable as described above. (Note: for python, no points are awarded for compilation, but execution is worth 30 points.)

- **15 points - Execution** Each executable must run without error or warning on valid input using the command line parameters described above. (Note: for python, no points are awarded for compilation, but execution is worth 30 points.)
- **10 points - README** Does your README documentation completely and accurately describe the task and approach taken? Does it satisfy the content requirements (i.e. how to compile and run your project, file listing, etc.)
- **20 points - Linguistics Oriented Requirement** Have you given a thoughtful response to the word/document representation problem? Is the response coherent and well-written? Does the response represent creative thinking about a common problem?
- **7 points - Within Code Documentation** Every function should include a comment minimally describing 1) what it does, 2) what its inputs are and 3) what its output is. Are there effective other comments throughout the code? You may use a javadocs, or pydoc, or other standard. For a good read check out the google style guides: `https://code.google.com/p/google-styleguide/`.
- **8 points - Style** Is the structure of your program clear and coherent? Are functions and variables given self-explanatory names? Are functions used to aid intelligibility of the code? Are functions used to reduce repeated blocks of code? Is indentation, spacing, use of parentheses, use of braces consistent, and sensible? For example, if you use brackets on the same line at the start of a block, always do so. If you place a brace on the line following the start of the brace, always do so. If you put a space between variables and operators, e.g. `if (i == j)`, always do so. So, `if (i == j) i = j+k;` is bad. It should be `if (i == j) i = j + k;` or if you prefer `if (i==j) i=j+k;`. You will be graded on consistency in these decisions, not on any particular style.
- **20 points - Correctness** Is/Are the algorithm(s) implemented correctly? Have an appropriate number of word/document representations been used and used correctly?
- **5 points - Instructor's Discretion** Has this assignment gone beyond the minimal requirements in a substantive way? Is it especially clear? Is the code especially well written? Is the response particularly thoughtful or insightful? Have non-trivial representations been examined?