HOMEWORK 2
MOVIE REVIEW CLASSIFICATION
CS 4705: NATURAL LANGUAGE PROCESSING
DUE: OCT 29, 2010 BY 11:59PM

Table of Contents

PROBLEM DEFINED

Do you read the reviews before seeing a movie? Have you ever changed your mind about watching a movie because you read bad reviews? Imagine that you read several movie reviews with star ratings given by different reviewers. Now you read an anonymous new review of *Avatar* in the newspaper that has no stars. Just from the review itself, can you guess who wrote it and how many stars they would give?

This is what this homework assignment is about: You will be running a number of machine learning experiments on a set of movie reviews. The tasks involve classifying movie reviews into a 4-star Rating; classifying the same reviews into a binary (Positive/Negative) rating; and identifying the reviewer who wrote the review. You will be turning in a total of five classifiers for this assignment.

THE DATA

You will be given an annotated data corpus to train your classifiers on. In this corpus there are 5006 reviews, one review per line. Each review/line consists of 4 fields: a review id; a reviewer ID (A,B,C, or D); a star rating(1 to 4/worst to best); and the text of the review itself. This corpus can be found at: /home/cs4705/HW2/movie-corpus.txt. The corpus will look like:

```
<id>1</id><reviewer>A</reviewer><star>3</star><review>It is an interesting comedy…</review>
<id>2</id><reviewer>C</reviewer><star>1</star><review>I am so glad that I saw it… </review>
<id>3</id><reviewer>B</reviewer><star>2</star><review>I like the story, but…</review>
```

THE CLASSIFICATION TASKS

You will need to classify movie reviews in the following separate experiments:

1. 4-STAR RATING CLASSIFIER TASK

In this task, you will be given a test set of movie reviews that you have not seen before (not in the training corpus). You should build a classifier to assign ratings to these reviews, using a 4-start rating scheme (1 to 4 / worst to best).

There will be two test sets for this classification task:

A. In the first test set, the reviews were written by the same four reviewers (A, B, C and D) who wrote the reviews in the given training corpus
B. In the second test set, the reviews were written by other reviewers, whose reviews do not appear in the training corpus

You must submit two classifiers for this classification task. ***Note***: Even If you decide to use the same classifier for both tasks A and B, you must still submit two classifiers and appropriate documentation for each.

2. BINARY (POSITIVE/NEGATIVE) RATING CLASSIFIER TASK

In this task, you are required to build a classifier that will simply classify a movie review as either positive or negative. For training, you should collapse the 3 and 4 star ratings to form the "positive" class and the 1 and 2 star ratings to form the "negative" class.

As in the 4-star classification in Task 1, there will be two tests for this binary classification, one on reviewers seen in the training data and one on unseen reviewers. Again, you must submit two classifiers for this task. Note: Again, if you decide to use the same classifier, you must still submit two classifiers, even though they are identical.

Hints: There are two strategies that you might adopt:

A. You might train classifiers for the Positive Negative rating separately
B. You might simply use your 4-star classifiers to classify reviews and then transform the results into a binary classification. Note that this approach may not give you the same results as (A)

Whichever strategy you adopt, please be sure that you turn in 2 classifiers for this task.

3. REVIEWER CLASSIFIER TASK

For this task, you will be required to classify reviews by reviewer; that is, you must identify which of the four reviewers whose reviews you have seen in the training data wrote a review. The test set will come from other reviews written by these four reviewers that are not in the training corpus you have. You must submit 1 classifier for this task.

TESTING

The test sets we will use to test all of your classifiers will have the same format. Each line will have a review id and the review content, as follows:

```
<id>5007</id><review>Excellent acting…</review>
<id>5008</id><review>Maybe young people like the movie, but for me…</review>
<id>5009</id><review>I don't understand why the director…</review>
```

For Task 1, the classification by 4-star Rating task, your classification results should be in the following format:

```
<id>5007</id><star>4</star><review>Excellent acting…</review>
<id>5008</id><star>2</star><review>Maybe young people like the movie, but for me…</review>
<id>5009</id><star>1</star><review>I don't understand why the director…</review>
```

For Task 2, the classification by Positive/Negative Rating task, your classification results should be in the following format:

```
<id>5007</id><PN>P</PN><review>Excellent acting…</review>
<id>5008</id><PN>N</PN><review>Maybe young people like the movie, but for me…</review>
<id>5009</id><PN>P</PN><review>I don't understand why the director…</review>
```

For Task 3, the classification by reviewer task, your classification results should be in the following format:

```
<id>5007</id><reviewer>A</reviewer><review>Excellent acting…</review>
<id>5008</id><reviewer>B</reviewer><review>Maybe young people like the movie, but for me…</review>
<id>5009</id><reviewer>A</reviewer><review>I don’t understand why the director…</review>
```

MACHINE LEARNING TOOLKIT

You are to use the machine learning toolkit ***weka*** in order to run your classification experiments. To this end, one part of your submission will be a program that generates ***weka.arff*** formatted files. As discussed in class and as in the weka documentation, these files describe your data set as a series of class-labeled feature vectors. You can find a weka tutorial on http://weka.wikispaces.com/Primer and you can download it from http://www.cs.waikato.ac.nz/ml/weka/ .

Your program should read the data set and produce one ***.arff*** file for each classification, for a total of 5 files. The feature set that you extract for use in these classification experiments is completely up to you; however, obviously, you must not use any of the review labels *(<id>, <reviewer>, <PN>, <star>)* as features in your feature vector, since these will not be available to you in the test set or, in the case of <id>, at all useful. You may extract different features for different classification tasks, but you are not required to do so. You should try at least three different classification algorithms for each task so you can see how they operate on different tasks. For these classification experiments you should use cross-validation; we recommend 10-fold cross-validation for these experiments but the number of folds is your choice. Note that you may use weka’s cross-validation options or you may choose to divide the data up yourself. It is essential that you use the weka package found at ***/home/cs4705/bin/weka.jar*** to run your experiments. If you do not, there is no guarantee that it will be possible to evaluate your final models. NB: You

You must also export the ***model*** that yielded the best results for each classification task, and submit it along with your feature extractor code – if you do not, evaluating your submission will be impossible. Also, it is essential that you indicate the classifier and parameters that generated the submitted model.

Tip: You may find that you want to use features that are calculated relative to the entire data set. For example, “does this review have more or fewer words that the average review in the training data?” These types of features can be very useful. However, you need to be careful when using them in a cross-validation scenario, since features that you calculate on the whole training corpus will include all of the data weka uses in the cross-validation folds. To avoid this problem, you can run the cross validation evaluation manually, by randomly dividing the training corpus into training and testing sets for feature extraction and (development) evaluation. However, for your submission you may build a model on your entire training set.

GRADING

The homework will be evaluated as follows:

FUNCTIONALITY (25PTS)

- Does the feature extractor compile?
- Does the feature extractor produce well-formed arff files?
- Did you include trained model files for each classification task?
- Did you submit all supporting scripts?
- How much do they limit the required human interaction?

RESULTS (35PTS)

- How well does the submission classify the supplied training data?
- How well does the submission classify the unseen testing data?

WRITE-UP (25PTS)

- Did you include the cross-validation accuracy for each experiment?
- Which classifiers did you use on each of the tasks? Why?
- Which were the fastest? Most accurate? Easiest to use?
- Which do you prefer and why?
- Which classification task was the easiest/hardest? Why?
- Within tasks, were some classes easier to classify than others? (NB: Examine the weka output for this information.) Why?
- Which classifications were the most similar/most different? Why?
- What features did you use? Why?
- Did they perform better or worse than expected?
- Did early experiments guide your thinking for your final submission? How?
- Which features were the most/least useful? Why?
- If you used any external resources, which did you use? How did they contribute to the success of your submission?

DOCUMENTATION (5PTS)

- Within-Code Documentation
- Every method should be documented.

CODING PRACTICES (5PTS)

- Informative method/variable names
- Efficient implementation, Memory, and Processor efficiency – don't sacrifice one unless another is improved

README FILE (5PTS)

This must include the following information:

- How to compile the feature extractor (if necessary).
- How to run the feature extractor.
- Which submitted model corresponds to which classification task.
- Which machine-learning algorithm (and parameters) generated the model.
- Any particular successes or limitations of the submission should be highlighted here.

Extra Credit may be awarded for particularly inventive or successful approaches to the assignment.

SUBMISSION

Your submission should require as little effort as possible to test; therefore you MUST follow these instructions:

In your submission you must include the following files generated by your system:

A. starRatingSameUsersTrain.arff and starRatingSameUsers.model
B. starRatingDiffUsersTrain.arff and starRatingDiffUsers.model
C. binaryRatingSameUsersTrain.arff and binaryRatingSameUsers.model
D. binaryRatingDiffUsersTrain.arff and binaryRatingDiffUsers.model
E. authorTrain.arff and author.model

The following are crucial scripts:

1. Submit one script to compile your code (if needed): ***make.sh***. This script should set all necessary environmental variables and should be tested on the clic.cs.columbia.edu machines before you submit your homework.
2. Submit ***five*** additional scripts, one for each classification task. Each of these scripts generates an arff file and runs weka on a given test file. These scripts will be used to test your models on unseen data, for example:

   ./starRatingSameUsers.sh starRatingSameUsers.model ./testfile1

   It will extract features from the test file in *testfile1* and generates ***starRatingSameUserTests.arff*** file. This script will also run weka using ***starRatingSameUsers.model*** and the generated ***starRatingSameUserTests.arff***.

Homework will be submitted via Courseworks:

1. Place all of your files in a directory using the naming convention: ***uni_hw2***
2. Archive and zip your folder:
   ```
   $ tar -cvfz uni_hw2.tar.gz uni_hw2
   ```
3. Upload the folder to Courseworks → Shared Files → HW2

ACADEMIC INTEGRITY

Copying or paraphrasing someone's work (code included), or permitting your own work to be copied or paraphrased, even if only in part, is not allowed, and will result in an automatic grade of 0 for the entire assignment or exam in which the copying or paraphrasing was done. Your grade should reflect your own work. If you believe you are going to have trouble completing an assignment, please talk to the instructor or TA in advance of the due date.

FAQ

When should I start doing homework 2?

You should start working on homework 2 right away. Homework 2 will need more time than homework 1. Some classifiers might take longer time in training than others. You will also run several experiments to find the best classifiers and feature sets. In addition, starting early will give you the advantage of working on the servers while they are not overloaded by everyone trying to meet the deadline.

Can I install weka on my machine and work locally?

Yes, you can work on your machine. Weka is available for Windows, Linux and Mac. Please make sure to install the same version that we will be testing on (weka-3-6-3). Also, plan to have some time to test your submission on the server to make sure that everything runs smooth without problems.

I'm trying to get started on homework 2, but I'm really not sure where to start. I understand that we're supposed to generate ARFF files, but I have no idea what we need to do to generate them or what attributes need to be defined in the files. Can you give any hints?

Try to extract features (integer values or strings) from the documents that distinguishes one class from others, these features should be general enough to describe the documents in the same class, not too specific otherwise your model will not generalize for the test data.

For example:

1. If you want to classify the movie review based on reviewer: the review length and/or punctuation could be good features.
2. You could try the following: take two reviews randomly from the training corpus set without looking at the tags. Try to guess the ratings of these reviews by reading them. Think about the reasons that led you to that decision – what features you were paying attention to. Try extracting these features and including them the .arff file. Note that each classifier may have its own set of features.

Are there any general techniques you can recommend to collect features? Is there anything specific that we were taught in class that would be helpful?

You can compute some statistics from the reviews and add them as features: you can extract lexical features, syntactic features (maybe the usage of some syntactic constituents differ from one class to other), and n-gram features – think about a way to represent them. There are more, just try to be creative as much as you can. NB: for each feature, you will need to explain your intuition about why that feature might be useful in your write-up; e.g., do not use a bi-gram feature if you do not have a supporting intuition or theory.

"You may extract different features for different classification tasks, but you are not required to". Even if we're not required to, are we still supposed to use different features? It doesn't seem like in most cases the same features that are needed to classify one category would be useful for another.

If you think that you found a magic set of features good enough to classify all types, just use them in all tasks. Actually, we recommend that you do this first, so you will have something to turn in in the worst case, and then begin to explore complicated and more task dependent features for the tasks that the initial set of features do not perform so well on.

Do you recommend using any other APIs besides weka for this assignment? Something that might help with feature extraction?

Weka is a set of machine learning (ML) algorithms, you will use weka only to train your classification models and run the cross-validation experiments -- not to extract any feature. You can use any library you like (obviously, except a document classifier or another ML toolkit). Your program should extract the features and produce the arff files...(then use them in weka).

You said we can use the output of a previous classification to help with another classification. It seems that to accomplish this, I would need to input a single feature vector to weka and have weka return a classification type. Is this what would I need to do? If so, can you give any hints on how to do this?

What you said is correct! To do that, for example let's assume that we want to use the reviewer classification to do the 4-star classification on a review X.

1. Extract one feature vector from review X (using the feature extractor for the reviewer classification)
2. Build an arrf file that includes this vector.
3. Run weka using the command line giving it the reviewer model and this arff file and weka will output the reviewer classification.
4. Run your feature extractor for the 4-star classification given the reviewer classification and the review X, it should build your new feature vector for review X.
5. Write it in your arff file.

Please refer to Weka command line help for the syntax.

Do you recommend any specific classification models? NaiveBayes was demoed in class, but other than that I have no idea how to differentiate any of them (other than trying each one manually and see which one gives the best results).

This is a good question; the answer to this question is in the Machine Learning course. NaiveBayes has its limitation because of the independent assumption; therefore it may not do well in this task, because your features may not be independent random variables. Try whatever is available in weka. Try J48 tree, JRip (rule based approach), SVM (SMO: this is a very strong classification algorithm, it's based on structural risk minimization – it should do well on unseen data, by principle) even though it might take a lot of time to train your model. Multilayer preceptor – using empirical risk minimization... In this class, you're required to experiment these algorithms but not to understand them deeply – you should read a bit about your classifier to support your choice in the write-up.