Homework 3 – Summarization and Machine Learning
Natural Language Processing
Due: Oct 14, 2008 at 2:00 p.m.

Assignment:

Your assignment is to run a number of machine learning experiments on a set of news data. For Homework 3, your task is to turn in the learned model. The TAs will test it on the held-out test data. For Homework 4, you will be asked to write a paper describing your experiments and finding. The task requires classifying each sentence in a set of news stories as to whether it should belong in a summary or not. The training and testing material consists of a corpus of paired summary/document sets where the summaries have been written by people. The task is to develop a learned algorithm that can generate a summary given an input set of related documents. This problem is called *multi-document summarization*.

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 in the *weka* documentation, these files describe your data set as a series of class-labeled feature vectors. Your program should read the data set and produce an .arff file for the training data. Each feature vector will represent the extracted features for a document sentence along with the class label (1 if it belongs in the summary and 0 otherwise).

A good part of your assignment will be determining the features for learning. The feature set that you extract for use in these classification experiments should use at least three different types of features. The choice of features is up to you. For example, some of the features that people have experimented with in the past include word frequency, other lexical metrics (e.g., TF*IDF, word probability), sentence position in the document, discourse cues (e.g., “In sum” “most importantly”), POS tags, etc. You should experiment with the impact of these different features on your accuracy by using each of the three feature types separately and then in combination. For example, suppose you decided to use: 1. all words, 2. the 50 most frequent bigrams, and 3. Sentence position. You would run the classifier four times, first with just words, second with just bigrams, third with position, and fourth with all features. You should describe the features you used. You may receive extra credit for additional features if they are well justified and if they increase your accuracy or for running and analyzing feature selection.

You should try two different classification algorithms for each task so you can see how they operate on different tasks. We suggest J48 and Naïve Bayes.

For these classification experiments you should use 10-fold cross-validation. It is essential that you use the *weka* package that can be 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. You must also export the model that yielded the best results for each 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.

You may find that you want to use features that are calculated relative to the entire data set. For example, “does this sentence have more “important” words or less than the average sentence in the training data?” These types of features can be useful. However, you need to be careful when using them in a cross-validation setting. These features should never be calculated using any testing material. This may force you to run the cross validation evaluation “manually”. That is, randomly dividing the training data into training and testing sets for feature extraction *and* evaluation. For your submission you may build a model on your entire training set.

Submission:

Submit your files together in a file named: <uni>HW3.tar.gz. Your submission should require as little human interaction as possible to test; therefore you MUST follow these instructions:

In your submission you must include the following files generated by your system:
Summarization.arff and Summarization.model

The following are crucial scripts:
1) Submit one script to compile your code: make.sh
2) Submit **one** additional scripts for the classification task. This script generates an arff file and runs weka on a given a directory that contains the input files.

These scripts will be used to test your models on unseen data, for example:

./runSummarization.sh Summarization.model /home/nlp/hw3-testfiles

- ⇨ It will extract features from all *.input files in /home/nlp/hw3-testfiles => generates SummarizationTest.arff file

This script will also run weka using Summarization.model and SummarizationTest.arff ==>
weka result report. To get these results from the command line you can use the following:

java -Xmx1G -cp /home/cs4705/bin/weka.jar weka.classifiers.trees.J48 –l Summarization.model -T Summarization.arff

(assuming that J48 algorithm was used when you built your model)

Materials:

You are provided with paired summary/document sets to develop your classifiers. This data set can be found at: **/home/cs4705/corpora/DUC/trainingdata.** The README describes the naming convention used for the manual summaries and documents. Note that to find a paired summary and document set you need to look at the first part of the file names which encode the document number. Note that in addition to extracting the features for each sentence, you will also need to extract the class by determining if the sentence occurred in the summary or not. In cases, where the summary is abstractive, you will need to develop a method for matching document sentences to summary sentences.

Requirements:
Note: same language restrictions hold as in HW 1.

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?
- Submit any supporting scripts.
- How much do they limit the required human interaction?

Feature set justification (10 points)

- Describe the features you used.
- Why did you choose this set?

Results (40pts)

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

Documentation (15pts)

- README file: This must include the following.
    - How to compile the feature extractor (if necessary)
    - How to run the feature extractor
    - What features are extracted? How, in broad strokes?
    - 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.

Within-Code Documentation

- Every method should be documented.

Coding Practices (10pts)

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

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

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.
(Appendices on the following pages)