Homework #4 (due on 4/3/17)

There will be four more homework assignments (including this one). I anticipate that the other three assignments will cover Web search, multilingual retrieval, and applications of NLP to IR. These assignments should involve less programming work, compared to the first three assignments, and this will give you more time to devote to your independent project. Also recall that you only need to hand in three of the remaining assignments; you can skip one HW assignment of your choice.

Problems (40 points)

(20 pts) Explain the following concepts and what they mean in terms of text classification: (1) *bias-variance tradeoff*; (2) *kernel trick*; (3) and *cross-validation*.

(20 pts) Compute how a naïve Bayes classifier classifies the following two 'documents' using the binomial (also known as Bernoulli) model. Show work. The two classes are 'Good' and 'Spam'. Recall in the binomial model estimates of P(word|class) are based on the percentage of documents of the class containing the word. Estimates of P(class) and P(word|class) are given for you in the tables below. Don't forget absence of a term is also important information.

Document 1: "cheap auto insurance"
Document 2: "auto repair manual"

Words	P(w|Good)	P(w|Spam)
auto	0.02	0.03
cheap	0.03	0.20
insurance	0.01	0.10
repair	0.04	0.01
manual	0.04	0.01

P(Good) = 0.75
P(Spam) = 0.25

Text Classification Lab (60 points)

For this assignment you will work with the ***Systematic Review*** dataset that I described in the video lectures. A chief goal is to build a binary classifier using the training data and to use the trained model to make predictions on the test data. You will have considerable flexibility in which methods you use for the task, but you will be asked to investigate a few experimental conditions. The data for this assignment are formatted differently from the first three programs, however you are welcome to reuse source code, if it makes any sense to do so.

Data

The dataset is available as three tab separated value (TSV) files which can be found on the course website. Each line of the file consists of 10 tab delimited fields which are described below. Any field may be empty, though there should be meaningful values for at least columns 1-3. The data are about 65 MB in size uncompressed and are encoded as UTF8, although the data should be nearly if not entirely English. The data are split into three portions: ***train***, ***dev*** and ***test***. Train should be used to build your models. Dev should be used for tuning (if needed). You will submit predictions for the Test partition.

Col.	Field	Description
1	Assessment	-1, 0, or 1; zero indicates unknown; 1=accept, -1=reject
2	Docid	Unique ID. Usually PubMed ID, or hashed ID
3	Title	Article title
4	Authors	List of authors (poss. separated with ';'). Unnormalized.
5	Journal	Journal title
6	ISSN	Numeric code for journal (possibly a list)
7	Year	Publication year
8	Language	Trigram for language code (e.g., "eng")
9	Abstract	Several sentence abstract from article
10	Keywords	List of keywords (formatting is highly variable) – should be ';' separated.

Download the data from the course web site and write a program that can process the data files and create feature vectors. You will then build a classifier from the training set. You can use any method for training (e.g., kNN, naïve Bayes, decision trees, SVMs, writing regular expression rules based on training samples, etc…). You may use, and in fact are strongly encouraged to use open source machine learning toolkits. Links to a number of such tools are available on my IR resource webpage. I suggest you consider using the SVMlight tool, which is available from http://svmlight.joachims.org/ (binaries are available for modern operating systems). I'll describe the process using the SVMlight tool below.

Baseline & Scoring (40 points)
Use features from the title and abstract sections (columns 3 and 9) of the data. Make predictions against the Test partition and report precision, recall and F_1 scores. For full credit show the work in your computation (i.e., the numerators and denominators for precision and recall). Recall is the percentage of +1s in the Test file that were correctly predicted to belong to the class; precision is the percentage of +1s in the output file (or that you predict are positive) which are indeed correct according to the Test file labels. $F_1 = 2*P*R/(P+R)$.

Choose-Your-Own Experiment (20 points)
Conduct another experiment, preferably something non-trivial. Here are a few examples to consider:

- Tokenization: use stemmed words instead of plain words
- Attributed fields: Have features be specific to the column they come from (i.e., Title:hospital is not the same as Abstract:hospital)
- Find other columns in the data that are helpful for prediction or determine that none are
- Try a different machine learning model than your baseline approach
- Sweep parameters for your learning method (experimenting on the Dev set), and find a setting that achieves the best results

You are not limited to the ideas above. Be sure to explain what your experimental condition is, and how the results compare to a baseline that uses "title + abstract" for features. Reported results should be on the Test partition as before.

No matter which classifier you use, clearly describe your methods (e.g., do you remove stopwords or perform stemming; do you use binary weights or TF/IDF weights); which tools you use with which parameter settings; and how features are determined. Also hand in any source code that you write for the assignment.

The 2 results that I find most interesting, compelling, and well described will receive between 2^1 and 2^2 extra credit points on the assignment.

SVMLight
SVMLight is one machine learner you may choose to use, but you are welcome to use others. To use SVMlight you should process the data files, and write out files of vectors, one document vector per line. For example:

```
+1 5:1 13:1 78:1 … 15008:1
+1 5:1 45:1 78:1 15000:1
-1 3:1 13:1 87:1 12000:1
```

"+1" in the leftmost column indicates that the vector is positive for the class and "-1" indicates it is negative. Each *termid:value* element in this example uses binary weights; '1' indicates the binary presence of a term in the document, and terms not in the document (i.e., the zeros) are not written out. You do not have to use binary weights. For each vector the termids must be non-decreasing from left to right. After you create vectors, build a classifier and use it to make predictions for the dev/test partitions. Note: it is critically important that the termids are consistent in the training and dev/test data (e.g., 'disease' gets termid=37 for both the phase1.train and phase1.dev documents).

The example above is in the format SVMlight expects. To train a model with SVMlight:

```
% svm_learn –j 10 phase1.train phase1.mod
```

The "–j 10" parameter is helpful due to class imbalance in this data. To run a test set against a trained model:

```
% svm_classify phase1.dev phase1.mod phase1.dev.out
```

The output file contains scores that are in line-by-line correspondence with the input vectors. Output scores (margins from the hyperplane) that are above zero indicate that the prediction is that the document belongs to the positive class. As a rough guideline, it is not too difficult to attain precision and recall over 40% for this dataset.