CPSC 67 Lab #4: Genre Classification

Due **Monday**, Feb. 24 (11:59pm)

The goal of this lab is to classify artists as belonging to a particular genre. To do this, you will implement both a k-nearest neighbors (kNN) classifier and a Naïve Bayes (NB) classifier.

You may need to make minor any updates to previous code in order to complete this lab (details below). As before, you will need to use the database and the files you have already downloaded. You will, however, spider for additional artist data in this lab. The wiki includes details on the artists and genres you will download. After downloading the additional data, you will have web pages for 80 artists: 30 Alternative, 20 Classic Rock, 15 Country Pop and 15 Hip Hop.

1 Content-Type

If you didn't already update your code to do so, modify your Spider class so that when pages are downloaded, the `doc_type` field is automatically updated in the database according to the Content-Type of the page you downloaded. See the writeup from Lab #3 for details about the Content-Type. This will allow you to keep your database content up-to-date as you download data for the 40 new artists.

2 k-Nearest Neighbors

The first classifier you will implement is the k-Nearest Neighbors (kNN) classifier. The standard algorithm for kNN is as follows:

Given a new, unlabeled document d, and a labeled collection of documents:

1. Compare d to each document in your collection.
2. Assign to d the *plurality* label of the k most similar documents.

You will report results for $k = 1$ and $k = 7$.

2.1 Document granularity

We aren't particularly concerned with labeling individual web pages associated with an artist. Rather, we would like to use all of the web pages associated with an artist as a way of predicting the genre classification for that artist. To this end, each "document" will actually be the concatenation of all the web pages associated with a particular artist. Since you have downloaded web pages for 80 artists, there are only 80 total "documents" in your collection.

2.2 Leave one out cross-validation

One problem with the above kNN algorithm is that we only have labeled data. That is, for all of the artists that we've already downloaded data web pages, we already know the genre classification for that artist. This means we have no "new, unlabeled document d". Instead we will use a procedure known as cross-validation

In cross-validation, we begin with a collection of documents, D, that we know the labels for. When then select a subset of the documents, C, that we will classify. We first remove the documents in C from the collection D. Next, we learn a classification function using the remaining documents in $D - C$. Next, using the learned classification algorithm, we classify each of the documents in C. Finally, we compare the results of our classification with the true labels associated with each of the documents in C. This is repeated until all of the documents in D have been labeled by this procedure.

Leave one out cross-validation is a special case of cross-validation where each subset C contains only a single document. In a collection of 80 documents, this means that you will have 80 classification functions, each time classifying only one of the 80 single-document subsets.

2.3 Comparison function

To compare your unlabeled document d to each document in your collection, you will use the vector retrieval tools from Lab #3 with a setting of TF-IDF (ntc.ntc). The unlabeled document d is the query, and each of the remaining 79 documents is your document collection.

2.4 Tying it all together

The pseudo-code for implementing all of the above pieces is as follows. For efficiency reasons, you may want to choose a slightly different implementation.

```
for each artist to classify, A:
    create a single Vector J containing all of A's web pages
    create an empty VectorCollection VC
    for each artist remaining in the collection, R:
        create a single Vector V containing all of R's web pages
        add V to VC
    using tf-idf, compare J to each document in VC
    label A with the plurality label of the top K most similar documents in VC
```

3 Naïve Bayes

The second classifier you will implement is the Naïve Bayes (NB) classifier. For each document d to be classified, the class label is chosen according to this formula:

$$\text{classification} = \underset{c \in C}{\operatorname{argmax}} P(c|d) = \underset{c \in C}{\operatorname{argmax}} P(c) \prod_{1 \le k \le n_d} P(t_k|c)$$

where C is the set of class labels, n_d is the number of tokens in d, $P(c)$ is the prior probability of the class label c, and $P(t_k|c)$ is the probability that term t_k occurs in class c.

We don't actually know the true probabilities $P(c)$ or $P(t|c)$, but we can estimate these from the data that we already have. We can estimate $P(c)$ as follows:

$$P(c) = \frac{N_c}{N}$$

where N_c is the number of documents with label c, and N is the total number of documents.

We will once again be using leave one out cross-validation, so let's assume that we're trying to figure out the genre classification for Eric Clapton. We remove Eric Clapton (a Classic Rock artist) from the document collection and estimate $P(c)$ by simply counting the number of times we see each class label, divided by the total number of documents in the collection. This leaves us with $P(\text{Alternative}) = \frac{30}{79}$, $P(\text{Classic Rock}) = \frac{19}{79}$, $P(\text{Pop Country}) = \frac{15}{79}$, and $P(\text{Hip Hop}) = \frac{15}{79}$. We have only 19 Classic Rock artists and only 79 total documents in the collection because we removed Eric Clapton.

To estimate $P(t|c)$, we simply count the number of times each term t occurs in documents labeled with c, divided by the total number of terms in all documents labeled with c:

$$P(t|c) = \frac{T_{ct}}{\sum_{t' \in V} T_{ct'}}$$

where T_{ct} is the number of times t occurs in documents labeled c. For example, if the word Clapton occurs 3 times in all Classic Rock documents and the Classic Rock documents contain 1 million words, then $P(\text{Clapton}|\text{Classic Rock}) = \frac{3}{1000000}$.

3.1 Smoothing

For many words, we will estimate $P(t|c)$ to be 0 since the word t will not appear at all in the documents labeled c. This is a problem since the NB formula multiplies together all of the term probabilities and multiplying by 0 will make the probability $P(c|d)$ be 0. Instead, we will use a very simple smoothing method called add-one smoothing (or Laplace smoothing). See page 240 for implementation details. (Or see your notes, we'll talk about it in class.)

$$P(t|c) = \frac{T_{ct} + 1}{\sum_{t' \in V} T_{ct'} + |V|}$$

where $|V|$ is the size of the vocabulary, excluding punctuation and stopwords.

As per our discussions in class, we want the vocabulary to include all the words in the training data and the test data. In other words, the vocabulary size never changes, regardless of which artist we are currently classifying.

3.2 Log Probabilities

Multiplying together many small probabilities can quickly lead to underflow. Instead of multiplying together lots of small numbers, we can add the logs of lots of small numbers. This will get rid of our underflow problem. See page 239 for details. The formulation for NB using logarithms is:

$$\text{classification} = \underset{c \in C}{\text{argmax}}\, P(c|d) = \underset{c \in C}{\text{argmax}}[\log P(c) + \sum_{1 \le k \le n_d} \log P(t_k|c)]$$

This works because $\log(xy) = \log(x) + \log(y)$.

3.3 Stopwords and punctuation

Unlike with kNN, which uses the vector retrieval model's TF-IDF weighting, the NB classifier treats all words equally, even those words that are unlikely to be of much benefit to classification. To overcome this problem, you will first exclude any token in a document unless it is a "word", ~~defined to be any string that passes the `isalpha()` test~~. To test if a string is a word, use the following test:

```
import re

def isword(word):
    rgx = re.compile('^[A-Za-z]+\$')
    if rgx.search(word): return True
    return False
```

In addition, from the remaining "words", you should create a stopword list – a list of words that won't be used for classification – by excluding the top 200 most frequent words across all documents. When calculating $P(t_k|c)$, you will exclude all tokens t_k that are in the stopword list.

3.4 Tying it all together

As with kNN, you will use leave one out cross-validation. In order to compute $P(c)$ and $P(t|c)$, you will need to estimate the probabilities from all the documents excluding the documents for one artist. Then, for each genre, you will calculate $P(c|d)$ and choose the genre with the highest probability.

```
create a stopword list from all documents in the collection

for each artist to classify, A:
    create a single document, d, out of all the documents associated with A
    for each artist remaining in the collection, R:
        estimate P(c) and P(t|c)
    for each genre, c:
        calculate P(c|d)
    classify A as belonging to the genre c that maximizes P(c|d)
```

4 Results

On the wiki, you will report the following:

1. Using each of the three methods (kNN with k = 1, kNN with k = 7, and NB), you will report the mean accuracy of each classification method (for all artists).

2. Using each of the three methods (kNN with k = 1, kNN with k = 7, and NB), you will report the confusion matrices for the 4 genres. Individual wiki pages have been set up for you to do this.

For your demo, you should be able to run these same queries on the common data set in `/scratch/cs67/data/` which, as of the end of demos on Friday, will contain 10 pages each for all 80 artists.