LIS590TX – Text Mining
Spring 2016
Assignment 2 – Feature Extraction and Text Classification

Motivation: Using decision trees is a good choice if you want an open box classifier where you can clearly see the features that are playing an important role in classification performance. With that said, decision tree algorithms can only split a given search space along lines that are parallel to the incoming attribute space, so the choice and transformation of attributes can play an important role in ensuring classification accuracy. The Naïve Bayes classifier is good choice when you have a sparse search space, which is often the case for text classifiers.

The purpose of this assignment is to provide you with hands-on practice building two supervised classification algorithms (decision trees and naïve bayes) that will predict the label assigned to an abstract of a scientific article. You will test the classifier on the collection, but the classifier will work on any abstract (of course we would expect that the classifier will work better for abstracts that are drawn from the same domain). Before starting this assignment you should review the slides from the two classification lectures and re-read the Yang and Pedersen paper as it has been a while since we covered this material.

Detailed Instructions:

Step 0) Make sure that you can log into oracle by following the class instructions.
Text mining and keyword search are different, you may want to watch this short utube video to get oriented: https://www.youtube.com/watch?v=40QIW9Sr6Io. The text classification task explored in this assignment it just one kind of text mining.

Step 1) Data Selection
I have prepared a dataset for you so that you can focus on the pre-processing and transformation stages of the knowledge discovery process. These two datasets will be available in the dmuser user. Select *users* at the bottom of the sql-developer, then select dmuser and then you will be able to see the tables. In your queries you will need to add the dmuser. Prefix. I first collected abstracts from two categories using the MeSH terms 'Heart Diseases' or 'Lung Neoplasms' from the 2014 Baseline Medline[1]. *I will use the terms class and category interchangeably throughout these instructions*. For the purposes of this assignment I wanted to make sure that we had an even distribution of abstracts so I ran the following query to see if the distributions were the same.

```
Select meshheading, count(*)          Result:
FROM ML330_SODA2014_MSH               Heart Diseases     54300
group by meshheading ;                Lung Neoplasms    155005
```

Clearly there are fewer abstracts on 'Heart Disease'. Some abstracts were assigned both class labels so I removed those abstracts from the collection and then sampled 25,000 abstracts from each category using the queries below. You don't need to run these, but I wanted to include them so that you can refresh you're your knowledge of sql:

[1] From the National Library of Medline http://www.nlm.nih.gov/bsd/licensee/2014_stats/baseline_doc.html

```
create table ml330_soda2014Sample
as select pmid from (select pmid
from ML330_SODA2014_MSH
where meshheading='Heart Diseases'
ORDER BY dbms_random.value)
where rownum<=25000;
```

```
insert into ml330_soda2014Sample
select pmid from (select pmid
from ML330_SODA2014_MSH
where meshheading='Lung Neoplasms'
ORDER BY dbms_random.value)
where rownum<=25000;
```

I used these tables to create the two tables that you will use in this assignment. The first table **ML330_SODA_NWD** contains the pubmed identifier (pmid), the unique sentence identifier (usentid), sentence type (H=heading and S=sentence), term identifier (termid) and the term. The first word in the sentence is typically been converted to lower case, but all other words are the same as they were in the original abstract case. The second table, **ML330_SODA_MSH** contains the pmid and the corresponding MeSH heading. You will need to join these tables on the pmid in order to count the number of terms and abstracts in each category. Note that both of these tables are in the dmuser's account. You don't need to copy these into your account, just prefix the table with dmuser, for example the following query will show you how many pmids are in the table

Select (distinct pmid) from dmuser.ML330_SODA_NWD;

Step 2) Preprocessing

If you were to use the entire collection as it stands i.e. without any pre-processing or feature selection, you would convert the NWD table to a matrix of size n x m, where n is the number of documents (rows) and m is the vocabulary (columns). The text mining algorithm must find patterns in this search space. Stemming (reducing the term to the base form) and converting all terms to lower case will drastically reduce the number of columns in this matrix and thus size of your search space. I strongly recommend that you use the lower case all the words (you can use the oracle function lower(term) to get the lower case form).

How many words are there with no vocabulary changes or pruning?

How many lower case words are there?

In addition to converting to lower case, you may want remove stop words, which are terms such as the, a, in, of, and, that don't contribute to the overall topic of an article. You will need to find a load a stopword list. These are medical texts, so a medical stopword list would be best, but any one will be fine. How many words are there after removing stopwords?

Terms that appear in every abstract and terms that appear in very few abstracts will be unable to accurately discern one category from another. Write SQL queries to identify and remove terms that appear in very few abstracts (say in 5 or fewer articles) or in most abstracts (say more than 95% of the articles). Feel free to explore variations of this pre-processing (i.e. set the lower bound higher or the upper bound lower) and include the details of your settings in your final report. Consider any other strategies that you can use that do not consider the category label, but result in better features and include details on how the vocabulary size changed.

What strategies did you use and how many words were in the final collection at this point?

Step 3) Transformations

The main focus of this assignment is to determine the features (in our case, words) that you will use in the classifier. Selecting the best words will play a critical role in classification performance, because reducing the vocabulary size helps to ensure that the classifier has the best chance of finding non-spurious associations. A large search space also causes the data mining algorithm to take a long

time to run. The number of terms after step 2 will be in the order of thousands, and we will reduce the number of terms to the order of hundreds.

3.1) Term Frequency (tf) * Inverse document frequency (idf)

This measure comes from information retrieval community and balances the number of times the term appears in a document (tf) against the number of documents in which the term appears (idf). The feature does not take into account the label. Because tf*idf can differ for the same word in two different abstracts (because the tf could be different) you will need to select the overall highest tf*idf scores for any abstract.

Term Frequency (TF): Write an sql statement that calculates the number of times that a term appears in each abstract. Hint you will need the aggregate function count, and the order by clause to identify terms that appear frequently. Save your results in a new table called TF.

Document frequency (DF) :Write an sql query that counts the number of articles in which a term appears and save your results in a table called DF.

With the TF and DF tables in place you are now ready to calculate tf*idf. You will end up with a table that includes the abstractID, the term and the tf*idf value for that term. The most common weighting scheme used is TF-IDF (again there are many variations on how to calculate tf*idf and you may use any you like), where

tij = TFjxIDFi
Where tij =weight assigned to term i in document j
TFj = number of times that a term appears in an article
IDF = log (N)-log(n)+1
Where N = number of documents
n = number of documents in which term n appears

Save the results of your query in a TFIDF table.

Although IDF is constant for all abstracts, TF can differ between abstracts. Thus the same word in different abstracts can have a different tf*idf weight (you can take a look at your TFIDF table to confirm this). You want to identify the 100 terms from the TFIDF table that have the highest values. Be careful as you write your query as the same term can have a different tf*idf score so make sure that you end up with 100 terms. There is an attribute in oracle called rownum that you can use to rank the terms. You may find the following helpful – which identifies the term and the maximum value. You may also use the average or minimum value by changing the aggregate function. You may also find the oracle reserved term rownum helpful.

```
Select term, max(tfidf) as maxtfidf
From TFIDF
Group by term;
```

3.2) Information Gain

The second feature selection strategy will consider the actual classification value. Before you start read the information gain notes from class and the Yang and Pedersen paper. You can certainly calculate information gain in a program, but if you have your data loaded into oracle you can also write a series of SQL statements that use count and some intermediate tables to calculate the terms that have the best information gain. I have provided a specific example below so that you can see how to create the first of these tables.

To achieve this goal you will need to write a series of SQL statements and intermediate tables. I have provided a specific example below so that you can see how to create the first of these tables. Consider the word "lung" which we would expect would be a good word to differentiate Lung Neoplasms (cancer) from heart disease. You need to write queries that will provide the data for the following contingency table. Note that because you know that there are 25,000 abstracts in each category you can just create a table that contains data in the first row and then subtract that amount from 25,000 to produce the second row.

	Cancer	NotCancer
Abstract has "lung"	17174	392
Abstract does not have "lung"	7826	24608
Total	25000	25000

Once you have the tables in place to provide the numbers for each term and MeSH combination you will need to review the information gain formula which is:

$$Information\ Gain\ (t) = -\sum_{i=1}^{m} \Pr(c_i) \log \Pr(c_i) + \Pr(t) \sum_{i=1}^{m} \Pr(c_i|t) \log \Pr(c_i|t) + \Pr(\bar{t})\ \sum_{i=1}^{m} Pr(c_i|\bar{t}) \log \Pr(c_i|\bar{t})$$

In this assignment we have an equal distribution of abstracts in each category, so you can ignore the first part of this equation $\Sigma Pr(c_i)\ logPr(c_i)$ component of this formula. The horizontal line above the t means that the term is negated. If we expand the formula we see that:

```
Pr(t)=P(term) = number of abstracts in which the term appears
                / total number of abstracts

Pr(ci|t)=P(class|term) = number of abstracts in class and term
                / total number of abstracts with term

Pr(ci|t̄)= P(class|not term)
          = number of abstracts in class without a term
          / total number of abstracts without the term
```

To calculate the information gain for the word *lung* you need to know

```
Pr(lung) = number of abstracts in which the word lung appears
                / total number of abstracts

P(Cancer|lung) = number of Cancer abstracts with the term lung
                / total number of abstracts with term lung

P(Heart Disease|lung)
          = number of Heart Disease abstracts with the term lung
                /total number of abstracts with term lung


Pr(not lung) = number of abstracts where the word lung does not appear
                / total number of abstracts

P(Cancer|not lung) = number of Cancer abstracts without the term lung
                / total number of abstracts without term lung

P(HeartDisease|not lung)
          = number of Heart Disease abstracts without the term lung
                / total number of abstracts without term lung
```

I recommend that you create intermediate tables that capture counts and probabilities and then combine them together to calculate the entire information gain score, that way you can check your

results at each step with words that you think would occur frequently with each of the different categories.

3.3) Create tables for classification

Once you have the terms ranked using either tf*idf or information gain write a query to identify the n terms with the best scores. Each of your top n terms will become a column in your table. You can use the pivot operator to convert rows to columns, but quotes produced when using the pivot operator can be a pain, so I suggest that you first create a new table, then add your data. For example if n = 5 we would create :

```
create table ML330_SODA_FEATURES1 (
      pmid integer,
      lung integer,
      cancer integer,
      heart integer,
      cardiac integer,
      cell integer);
```

Then populate your new table using the pivot query in oracle. Pivot allows us to convert the words that are initially stored in a table as row to a table where the words are columns.

```
Insert into ML330_SODA_FEATURES1
select *
from (select pmid, term
         from ML330_SODA_NWD)
         pivot
      (count(term)
            for term in ('lung', 'cancer','heart','cardiac','cell'));
```

You will replace the list of terms with an SQL query that identifies the top 100 terms that have the best information gain. A closer look at some example records in `ML330_SODA_FEATURES1` shows that each row corresponds to an abstract and each column corresponds to word. The decision tree will be easier to read if you can name your columns the actual words, but you will need to be careful as some words selected may be reserved words in the database (for example if you try to call your column name "select" you would get an error)

Lastly you will need to join the results of this query with the ML330_SODA_MSH table so that you can add the class label for each of the abstracts. Repeat the feature selection process to more times, so that you end up with two additional tables `ML330_SODA_FEATURES2` that has the 200 best information gain words and `ML330_SODA_FEATURES3` that has the best 500 information gain words.

At the end of step 3 you will have 6 tables that look similar to this, with 100, 200 and 500 terms for tf*idf and a second set of 100,200 and 500 terms using information gain.

PMID	LUNG	CANCER	HEART	CARDIAC	CELL	MESHHEADING
851	0	0	1	0	0	Heart Diseases
1240	0	0	0	0	0	Heart Diseases
1853	0	0	0	0	0	Heart Diseases
1921	0	0	0	0	0	Lung Neoplasms
2958	0	0	2	0	0	Heart Diseases
4857	0	0	0	1	0	Heart Diseases

Step 4) Data Mining
Use the oracle data miner to create two classifiers (a decision tree and naïve bayes) to predict the correct category of abstract (see the slides from class for instructions) from each of your feature sets that you created in step 3. Note that the default in oracle is to also create a support vector machine and a general linear model, which are not needed in this assignment.

Step 5) Interpretation
Write a report (at most 10 pages, single spaced) that outlines your experiments and any decisions that you made along the way. The introduction should include how you selected the vocabulary, the accuracy of each classifier (i.e. decision trees and naïve Bayes) at each of the three feature spaces (i.e. 100, 200 and 500 words) and two feature selection strategies (tf*idf and information gain). Discuss how the decision trees changed and include a confusion matrix, which you can find by looking at the decision tree model, and then selecting the performance tab.

Look at the actual decision tree and terms from the naïve Bayes classifier that played an important role. Are these the terms you would have expected?

Submission Instructions: Upload your report to moodle.

Grading: This assignment is worth 10% of your grade, but more importantly you will use the feature selection strategies you have implement in this assignment for your final project.