University of Regina

Department of Computer Science

Winter 2019

CS 824 - Information Retrieval

Assignment 2

Submitted to Dr. Yiyu Yao

By

Chao Zhang

Regina, February 21, 2019

1. Discussion of the main ideas of indexing.

Indexing plays a significant role in search engine. It provides fast, accurate processing for information retrieval by collecting, analyzing and storing data. The main idea of indexing in information retrieval is to analyze set of documents to reorganize the contents. It is a process from external representation of documents such as papers to internal presentation like a list of key words. By indexing, we can find useful contents within many sets of documents whatever which forms they are. Generally, there are two kinds of indexing: manual indexing and automatic indexing. In addition, for determining sets of documents, it has controlled vocabulary and uncontrolled vocabulary. There is a bridge from paper to user in indexing, in other word, it is from lists of terms to query terms. A basic assumption for indexing is that if a document contains a word, then it is about the word and the number of appearance of a word is related to its importance. So, based on the assumption, there are three steps in indexing: building a dictionary, combing several terms of high frequencies and putting a set of terms to one class as a new term. Finally, the result of indexing is a list of keywords with frequencies which can be classified into a term x frequencies matrix, shortly TF matrix.

2. Describe different term of weighting methods.

From TF matrix, we can know the number of appearance of terms in documents. However, sometimes we want to know the importance of terms in describing documents. So, we need some weighting methods to transfer TF matrix to a weighted matrix.

Method 1: Document frequency of a term t. The number of documents where t appears. $df(t) = \{TF(i, j) \neq 0 \mid i \leq j \leq n\}$.

Method 2: Row-wise normalization: change each document to a probability document.

Method 3: Column-wise normalization: term distribution of d_j in different documents.

Method 4: $TF * IDF$. Term frequency * Inverse document frequency.

Method 5: Information entropy of a term: $H_B(t_j) = df(t_j)(-\frac{1}{df(t_j)} * \log\frac{1}{df(t_j)})$

3. Describe the basic ideas of IDF and provide an explanation of IDF based on Shannon information theory.

Inverse document frequency(IDF) is a measure of specificity of a term based on B matrix. For some terms which have less important, they may appear frequently. However, some important terms are rare. So IDF can be used to determine how much information a term carries. Through IDF we can find these higher important information with lower probability terms and those higher probability terms which are not important.

Shannon information entropy is an expectation of information. The meaning of Shannon is a measure of information. It provides which level uncertainty information is. From TF matrix to B matrix to column-wise normalization of B, we have $H_B(t_j) = df(t_j) * \left(-\frac{1}{df(t_j)}\lg\frac{1}{df(t_j)}\right) = \log df(t_j)$. For IDF, We have $IDF(t) = \frac{1}{\log df(t)}$. Combining these two formulas, we can get $IDF = \frac{1}{H_B}$. Similarly, both IDF and Shannon information theory tell us how much information each specific term carries.

4. Describe system evaluation measures based on binary relevance judgments, namely, a document is either relevant or non-relevant.

The purpose of system evaluation is to measure how good is the match. We can use Boolean Model for evaluation. A document is either relevant or non-

relevant which means a document is either useful or not useful. We consider documents are relevant and non-relevant in R and $\overline{R}$. For the set of documents as D, we have $R \cup \overline{R} = D$ and $R \cap \overline{R} = \phi$. Similarly, for an information retrieval system, we have $RET \cup \overline{RET} = D$ and $RET \cap \overline{RET} = \phi$. System evaluation is to find measures which tell whether a system is good and if one system is better than another system. Based on binary relevance judgments, we have precision function $PRECISION = \frac{|R \cap RET|}{|RET|}$ and recall function $RECALL = \frac{|R \cap RET|}{|R|}$ where $0 \le RECALL \le 1$. There is a kind of inverse relationship between precision and recall. These qualities are also related to F measure which is harmonic mean of precision and recall. So we have $F = 2 * \frac{PRECISION * RECALL}{PRECISION + RECALL}$ where $0 \le F \le 1$.

5. Use an example to draw a recall-precision graph.

In this example, I generate 400 samples for original data and differentiate them into two different classes. Then I compute the average precision score and draw the recall-precision. Typically, precision-recall graph is used in binary classification to study the output of a classifier.

The example is implemented by python and details are covered below:

The source code can be download from:

https://github.com/XueGeChuiZi/CS-824

Import python package:

```
from sklearn import svm, datasets
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.metrics import average_precision_score
from sklearn.metrics import precision_recall_curve
import matplotlib.pyplot as plt
from sklearn.utils.fixes import signature
```

Create simple data:

```
iris = datasets.load_iris()
X = iris.data
y = iris.target

# Add noisy features
random_state = np.random.RandomState(0)
n_samples, n_features = X.shape
X = np.c_[X, random_state.randn(n_samples, 400 * n_features)]

# Limit to the two first classes, and split into training and test
X_train, X_test, y_train, y_test = train_test_split(X[y < 2], y[y < 2],test_size=.5,random_state=random_state)

# Create a simple classifier
classifier = svm.LinearSVC(random_state=random_state)
classifier.fit(X_train, y_train)
y_score = classifier.decision_function(X_test)
```

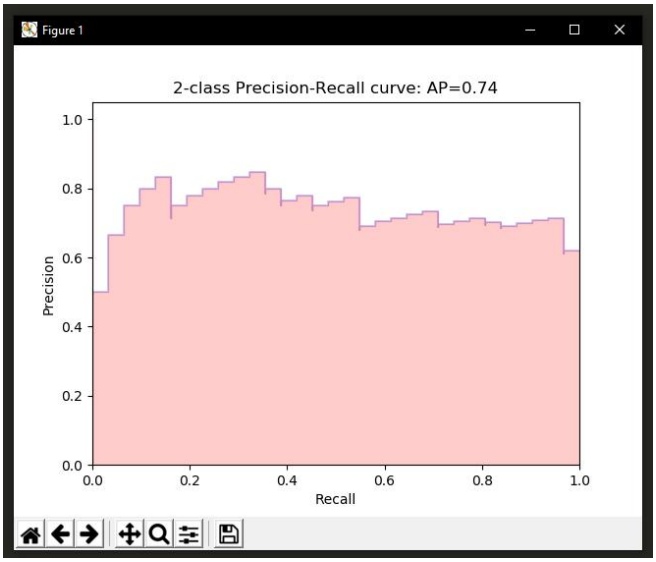
Compute the average precision score:

```
average_precision = average_precision_score(y_test, y_score)
print('Average precision-recall score: {0:0.2f}'.format(average_precision))
```

Draw a recall-precision graph:

```
precision, recall, _ = precision_recall_curve(y_test, y_score)
step_kwargs = ({'step': 'post'}
               if 'step' in signature(plt.fill_between).parameters
               else {})
plt.step(recall, precision, color='b', alpha=0.2,where='post')
plt.fill_between(recall, precision, alpha=0.2, color='r', **step_kwargs)
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.ylim([0.0, 1.05])
plt.xlim([0.0, 1.0])
plt.title('2-class Precision-Recall curve: AP={0:0.2f}'.format(average_precision))
plt.show()
```

Output:



6. Describe a method of representing relevance by using user preference relations, and system evaluation based on user preferences.

For the set of documents D, based on user preference relation $\succ_u$, we have d and d' which means user prefers d to d'. In other word, d is more relevant to d. Due to the user preference is a weak order, it is asymmetric and negatively. We can find a function u: D---->R such that $d \succ_u d' \Leftrightarrow u(d) \succ u(d')$ if only if $\succ_u$ is a weak order. That is an IR system can reproduce the user ranking. Suppose u is a retrieval function used by an IR system, for any two documents d1 and d2 in set of documents D, we can say that u produce a perfect ranking of $d1 \succ_u d2 \Leftrightarrow u(d1) \succ u(d2)$. We say that u produce an acceptable ranking: $d1 \succ d2 \Rightarrow u(d1) \succ u(d2)$. We use acceptable ranking as our design objective.

For system evaluation based on user preferences, we may generate a distance between $\succ_u$ and $\succ_s$ to measure the relevance. A distance between two rankings can be considered into a mathematical problem by analyzing two sets. In addition, ranking is a binary relation and a binary relation is a set of pairs. If we have a set of pairs, we can look at individual pairs. For one pair (d1,d2), we can classify it into three areas: (1) the system and user agree on (d1,d2); (2) if one is a tire and the other one is a preference; (3) the system and user contradict each other. So the distance between $\succ_u$ and $\succ_s$ can be $2*\left|\succ_u^c \cap \succ_s\right| + 1*\left|\succ_u \cap \sim_s\right|$ where $\left|\succ_u^c \cap \succ_s\right|$ is the number of contradiction pairs and $\left|\succ_u \cap \sim_s\right|$ is the number of compatible pairs.