CSCE 470 Homework #1

Rodrigo Gomez - Palacio

February 20, 2017

Question 1 - Programming Code with outputs can be found on github at *https://goo.gl/dqW0er*

```
---Preprocessing Output---
['today', u'aggi', 'won', 'go', u'aggi']
[u'aggi', 'won', 'today']
[u'aggi', 'lost', 'last', 'week']
['find', 'latest', u'aggi', u'news']
[u'aggi', 'student', u'texa']

---Incidence matrix---
[1, 1, 0, 0, 0]
[1, 1, 1, 1, 1]
[1, 1, 0, 0, 0]
[1, 0, 0, 0, 0]
[0, 0, 1, 0, 0]
[0, 0, 1, 0, 0]
[0, 0, 1, 0, 0]
[0, 0, 0, 1, 0]
[0, 0, 0, 1, 0]
[0, 0, 0, 1, 0]
[0, 0, 0, 0, 1]
[0, 0, 0, 0, 1]

---Count_matrix:---
[2, 1, 1, 1, 1]
[0, 0, 0, 1, 0]
[1, 0, 0, 0, 0]
[0, 0, 1, 0, 0]
[0, 0, 0, 1, 0]
[0, 0, 1, 0, 0]
[0, 0, 0, 1, 0]
[0, 0, 0, 0, 1]
[0, 0, 0, 0, 1]
[1, 1, 0, 0, 0]
[0, 0, 1, 0, 0]
[1, 1, 0, 0, 0]

---Tf_idf_matrix---
[0.0, 0.0, 0.0, 0.0, 0.0]
[0.0, 0.0, 0.0, 0.21, 0.0]
[0.21, 0.0, 0.0, 0.0, 0.0]
[0.0, 0.0, 0.21, 0.0, 0.0]
[0.0, 0.0, 0.0, 0.21, 0.0]
[0.0, 0.0, 0.21, 0.0, 0.0]
[0.0, 0.0, 0.0, 0.21, 0.0]
[0.0, 0.0, 0.0, 0.0, 0.21]
[0.0, 0.0, 0.0, 0.0, 0.21]
[0.12, 0.12, 0.0, 0.0, 0.0]
[0.0, 0.0, 0.21, 0.0, 0.0]
[0.12, 0.12, 0.0, 0.0, 0.0]

---Cosine Similarity Matrix---
1 [1, 0.882, 0.338, 0.338, 0.39]
2 [0.882, 1, 0.289, 0.289, 0.333]
3 [0.338, 0.289, 1, 0.25, 0.289]
4 [0.338, 0.289, 0.25, 1, 0.289]
5 [0.39, 0.333, 0.289, 0.289, 1]
```

Question 2 - Indexing

1) Consider the documents in Question 1, graphically illustrate the inverted index you will create for the documents.

```
[ aggi 5 ] -> 0 -> 1 -> 2 -> 3 -> 4 -> //

[ find 1 ] -> 3 -> //

[ go 1 ] -> 0 -> //

[ last 1 ] -> 2 -> //

[ latest 1 ] -> 3 -> //

[ lost 1 ] -> 2 -> //

[ news 1 ] -> 3 -> //

[ student 1 ] -> 4 -> //

[ texa 1 ] -> 4 -> //

[ today 2 ] -> 0 -> 1 -> //

[ week 1 ] -> 2 -> //

[ won 2 ] -> 0 -> 1 -> //
```

2) Given the query "aggies won", make use of the index we just build to retrieve the matched documents. Illustrate the query processing procedure in details.

Query: "aggies won"

To process this query using the index, we must follow the procedure:

1) Preprocess the query. This is important because the individual query terms must match the terms in the index. Separate the query into terms "aggies" and "won". The output of this step is "aggi" and "won."

2) Locate the postings for the individual query terms in the dictionary. Each term header in the index dictionary contains a pointer to the document postings in the form of a linked list. We first go to "aggi" in the dictionary and see that

3) Merge the postings to form one long list of documents using skip pointers. The query postings must be aggregated efficiently to return the set of relevant documents.

4) Return the resulting documents.

3) Explain the advantages of inverted index over term-document incidence matrices for organizing texts?

Response: Term-document incidence matrices don't make much sense for very sparse matrices. That is to say, indexing and querying a very large set of documents with many terms is going to be very costly in terms of running time as well as space complexity. This is because the majority of the matrix will be comprised of zeroes. In contrast, an inverted index allows for the quick retrieval of document sets because each term has a direct pointer to a postings list, avoiding the expensive search one has to do in a matrix.

Question 3 - Evaluation Metrics For a particular search query, your IR systems returns 75 relevant documents and 25 irrelevant documents. There are a total of 125 relevant documents in the overall collection.

1) What are the precision and recall of the system on the search?

Precision: $TruePositive/TotalDetected = 75/100 = 0.75$

Recall: $TruePositive/TotalRelevant = 75/125 = 0.6$

2) Can you think of a search scenario where recall is preferred over precision ? Explain.

Yes. When it comes to detecting malicious diseases, it is better to err on the side of less precision in exchange for less risk of a false negative. In other words, it is better to detect more false positives than letting someone with a serious disease go undetected. Because detecting a malicious disease early can sometimes mean the difference between life and death, lowering the precision threshold is a good trade-off for widening the safety net of detection.

3) An obvious alternative to the precision / recall metric is the accuracy score, that is, the percentage of its classifications that are correct. Please explain the vulnerability of accuracy in evaluating the performance of IR systems alone. Provide one example where a decent accuracy score may be accompanied with terrible IR performance.

Response: Accuracy is not always a good measure of a system's performance. For example, a classifier may have little predictive power, yet still have good accuracy. We can define accuracy like this:

$(TP + TN)/(TP + TN + FP + FN) = accuracy$

A good example in relation to IR systems would be a simple spam filter for your email. For example, the spam filter detects 2 spam messages that are indeed, spam. In addition, it labeled "spam" 8 messages that weren't spam. As you can see, it missed 4 times as many spam messages as it got correct. Finally, it labeled "not spam" 90 messages that were indeed not spam. There were no false negatives. The accuracy for this filter would thus be $(2 + 90)/(2 + 90 + 8 + 0) = 92\%$, which is not a good representation of the filter's performance.

Question 4 - Tolerant Retrieval

1) Build a permuterm index for "sydney" and "sidney", respectively.

Sydney: sydney$, ydney$s, dneysy, neysyd, ey$sydn, y$sydne, $sydney

Sidney: sidney$, idney$s, dneysi, neysid, ey$sydn, y$sidne, $sidney

2) Consider the wildcard query "s*dney", explain in details how to search this query in the permuterm indices. What is the rotated wildcard query that we will look up for in the permuterm indices?

First, the query $s * dney$ is broken up into $X = s$ and $Y = dney$, as separated by the asterix. Next, we look at the query formats and see that $X * Y$ should be looked up on $Y\$X*$. Therefore, we will use the index $dney\$s*$ to look up the query.

3) Edit distances:

aggie-aggies: 1

november-december: 6

condense-confidence: 4

4) Trigrams

aggie-aggies: 3 trigram overlaps

november-december: 3 trigram overlaps

condense-confdence: 2 trigram overlaps