

# Relevance Feedback

CS5154/6054

Yizong Cheng

9/27/2022

## 9 *Relevance feedback and query expansion*

## 9.1 Relevance feedback and pseudo relevance feedback

### RELEVANCE FEEDBACK

The idea of *relevance feedback* (RF) is to involve the user in the retrieval process so as to improve the final result set. In particular, the user gives feedback on the relevance of documents in an initial set of results. The basic procedure is:

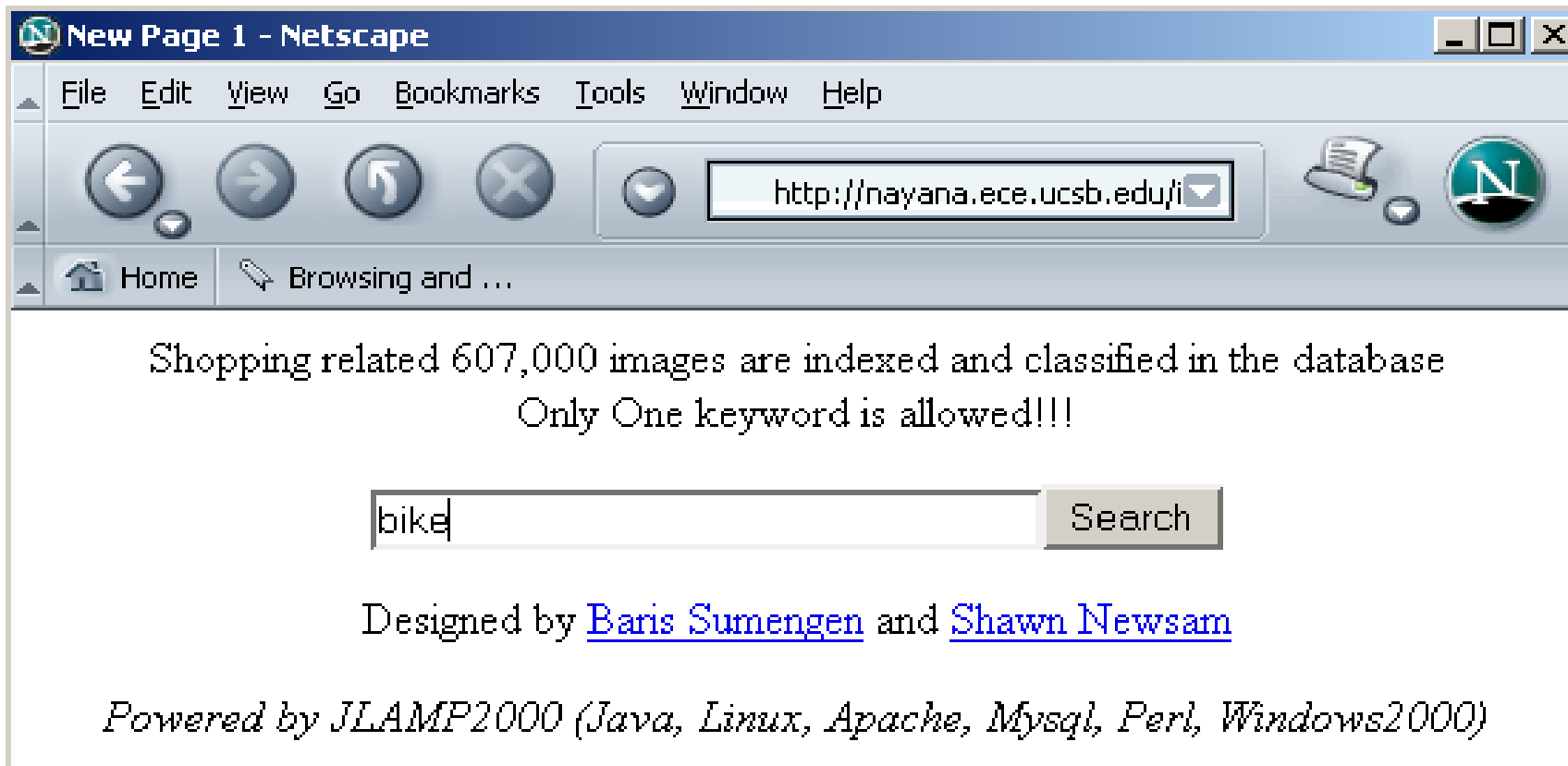
- The user issues a (short, simple) query.
- The system returns an initial set of retrieval results.
- The user marks some returned documents as relevant or nonrelevant.
- The system computes a better representation of the information need based on the user feedback.
- The system displays a revised set of retrieval results.













Relevance feedback can go through one or more iterations of this sort. The

# Relevance Feedback: Example

- Image search engine







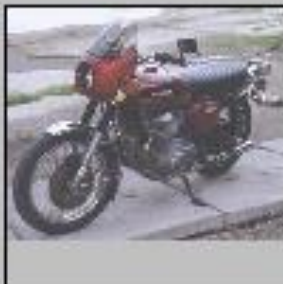





<http://nayana.ece.ucsb.edu/imsearch/imsearch.html>



<div> <a href="#">Browse</a> <a href="#">Search</a> <a href="#">Prev</a> <a href="#">Next</a> <a href="#">Random</a> </div>					
 <p>(144473, 16458) 0.0 0.0 0.0</p>	 <p>(144457, 252140) 0.0 0.0 0.0</p>	 <p>(144456, 262857) 0.0 0.0 0.0</p>	 <p>(144456, 262863) 0.0 0.0 0.0</p>	 <p>(144457, 252134) 0.0 0.0 0.0</p>	 <p>(144483, 265154) 0.0 0.0 0.0</p>
 <p>(144483, 264644) 0.0 0.0 0.0</p>	 <p>(144483, 265153) 0.0 0.0 0.0</p>	 <p>(144518, 257752) 0.0 0.0 0.0</p>	 <p>(144538, 525937) 0.0 0.0 0.0</p>	 <p>(144456, 249611) 0.0 0.0 0.0</p>	 <p>(144456, 250064) 0.0 0.0 0.0</p>

# Relevance Feedback













Browse Search Prev Next Random

					
(144473, 16458)	(144457, 252140)	(144456, 262857)	(144456, 262863)	(144457, 252134)	(144483, 265154)
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
					
(144483, 264644)	(144483, 265153)	(144518, 257752)	(144538, 525937)	(144456, 249611)	(144456, 250064)
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0



# Results after Relevance Feedback

[Browse](#) [Search](#) [Prev](#) [Next](#) [Random](#)

 (144538, 523493) 0.54182 0.231944 0.309876	 (144538, 523835) 0.56319296 0.267304 0.295889	 (144538, 523529) 0.584279 0.280881 0.303398	 (144456, 253569) 0.64501 0.351395 0.293615	 (144456, 253568) 0.650275 0.411745 0.23853	 (144538, 523799) 0.66709197 0.358033 0.309059
 (144473, 16249) 0.6721 0.393922 0.278178	 (144456, 249634) 0.675018 0.4639 0.211118	 (144456, 253693) 0.676901 0.47645 0.200451	 (144473, 16328) 0.700339 0.309002 0.391337	 (144483, 265264) 0.70170796 0.36176 0.339948	 (144478, 512410) 0.70297 0.469111 0.233859

# Initial query/results

- Initial query: *New space satellite applications*

- + 1. 0.539, 08/13/91, [NASA Hasn't Scrapped Imaging Spectrometer](#)
- + 2. 0.533, 07/09/91, [NASA Scratches Environment Gear From Satellite Plan](#)
- 3. 0.528, 04/04/90, [Science Panel Backs NASA Satellite Plan, But Urges Launches of Smaller Probes](#)
- 4. 0.526, 09/09/91, [A NASA Satellite Project Accomplishes Incredible Feat: Staying Within Budget](#)
- 5. 0.525, 07/24/90, [Scientist Who Exposed Global Warming Proposes Satellites for Climate Research](#)
- 6. 0.524, 08/22/90, [Report Provides Support for the Critics Of Using Big Satellites to Study Climate](#)
- 7. 0.516, 04/13/87, [Arianespace Receives Satellite Launch Pact From Telesat Canada](#)
- + 8. 0.509, 12/02/87, [Telecommunications Tale of Two Companies](#)

- User then marks relevant documents with “+”.



## Expanded query after relevance feedback

- 2.074 new
- 30.816 satellite
- 5.991 nasa
- 4.196 launch
- 3.516 instrument
- 3.004 bundespost
- 2.790 rocket
- 2.003 broadcast
- 0.836 oil
- 15.106 space
- 5.660 application
- 5.196 eos
- 3.972 aster
- 3.446 arianespace
- 2.806 ss
- 2.053 scientist
- 1.172 earth
- 0.646 measure

# Results for expanded query

- 2 1. 0.513, 07/09/91, [NASA Scratches Environment Gear From Satellite Plan](#)
- 1 2. 0.500, 08/13/91, [NASA Hasn't Scrapped Imaging Spectrometer](#)
3. 0.493, 08/07/89, [When the Pentagon Launches a Secret Satellite, Space Sleuths Do Some Spy Work of Their Own](#)
4. 0.493, 07/31/89, [NASA Uses 'Warm' Superconductors For Fast Circuit](#)
- 8 5. 0.492, 12/02/87, [Telecommunications Tale of Two Companies](#)
6. 0.491, 07/09/91, [Soviets May Adapt Parts of SS-20 Missile For Commercial Use](#)
7. 0.490, 07/12/88, [Gaping Gap: Pentagon Lags in Race To Match the Soviets In Rocket Launchers](#)
8. 0.490, 06/14/90, [Rescue of Satellite By Space Agency To Cost \\$90 Million](#)

### 9.1.6 Pseudo relevance feedback

PSEUDO RELEVANCE  
FEEDBACK  
BLIND RELEVANCE  
FEEDBACK

*Pseudo relevance feedback*, also known as *blind relevance feedback*, provides a method for automatic local analysis. It automates the manual part of relevance feedback, so that the user gets improved retrieval performance without an extended interaction. The method is to do normal retrieval to find an initial set of most relevant documents, to then *assume* that the top  $k$  ranked documents are relevant, and finally to do relevance feedback as before under this assumption.

This automatic technique mostly works. Evidence suggests that it tends to work better than global analysis (Section 9.2). It has been found to improve performance in the TREC ad hoc task. See for example the results in Figure 9.5. But it is not without the dangers of an automatic process. For example, if the query is about copper mines and the top several documents are all about mines in Chile, then there may be query drift in the direction of documents on Chile.

# 11 *Probabilistic information retrieval*

## 11.3.4 Probabilistic approaches to relevance feedback

We can use (pseudo-)relevance feedback, perhaps in an iterative process of estimation, to get a more accurate estimate of  $p_t$ . The probabilistic approach to relevance feedback works as follows:

1. Guess initial estimates of  $p_t$  and  $u_t$ . This can be done using the probability estimates of the previous section. For instance, we can assume that  $p_t$  is constant over all  $x_t$  in the query, in particular, perhaps taking  $p_t = \frac{1}{2}$ .

2. Determine a guess for the size of the relevant document set. If unsure, a conservative (too small) guess is likely to be best. This motivates use of a fixed size set  $V$  of highest ranked documents.
3. Improve our guesses for  $p_t$  and  $u_t$ . We choose from the methods of Equations (11.23) and (11.25) for re-estimating  $p_t$ , except now based on the set  $V$  instead of  $VR$ . If we let  $V_t$  be the subset of documents in  $V$  containing  $x_t$  and use add  $\frac{1}{2}$  smoothing, we get:

$$(11.26) \quad p_t = \frac{|V_t| + \frac{1}{2}}{|V| + 1}$$

and if we assume that documents that are not retrieved are nonrelevant then we can update our  $u_t$  estimates as:

$$(11.27) \quad u_t = \frac{\text{df}_t - |V_t| + \frac{1}{2}}{N - |V| + 1}$$

4. Go to step 2 until the ranking of the returned results converges.

### 11.3.2 Probability estimates in theory

For each term  $t$ , what would these  $c_t$  numbers look like for the whole collection? (11.19) gives a contingency table of counts of documents in the collection, where  $df_t$  is the number of documents that contain term  $t$ :

(11.19)

documents		relevant	nonrelevant	Total
Term present	$x_t = 1$	$s$	$df_t - s$	$df_t$
Term absent	$x_t = 0$	$S - s$	$(N - df_t) - (S - s)$	$N - df_t$
Total		$S$	$N - S$	$N$

Using this,  $p_t = s/S$  and  $u_t = (df_t - s)/(N - S)$  and

### 11.3.3 Probability estimates in practice

Under the assumption that relevant documents are a very small percentage of the collection, it is plausible to approximate statistics for nonrelevant documents by statistics from the whole collection. Under this assumption,  $u_t$  (the probability of term occurrence in nonrelevant documents for a query) is  $df_t/N$  and

$$(11.22) \quad \log[(1 - u_t)/u_t] = \log[(N - df_t)/df_t] \approx \log N / df_t$$



Let us make an additional simplifying assumption that terms not occurring in the query are equally likely to occur in relevant and nonrelevant documents: that is, if  $q_t = 0$  then  $p_t = u_t$ . (This assumption can be changed, as when doing relevance feedback in Section 11.3.4.) Then we need only consider terms in the products that appear in the query, and so,

$$(11.15) \quad O(R|\vec{q}, \vec{x}) = O(R|\vec{q}) \cdot \prod_{t:x_t=q_t=1} \frac{p_t}{u_t} \cdot \prod_{t:x_t=0, q_t=1} \frac{1-p_t}{1-u_t}$$

The left product is over query terms found in the document and the right product is over query terms not found in the document.

We can manipulate this expression by including the query terms found in the document into the right product, but simultaneously dividing through by them in the left product, so the value is unchanged. Then we have:

$$(11.16) \quad O(R|\vec{q}, \vec{x}) = O(R|\vec{q}) \cdot \prod_{t:x_t=q_t=1} \frac{p_t(1-u_t)}{u_t(1-p_t)} \cdot \prod_{t:q_t=1} \frac{1-p_t}{1-u_t}$$

$$(11.16) \quad O(R|\vec{q}, \vec{x}) = O(R|\vec{q}) \cdot \prod_{t:x_t=q_t=1} \frac{p_t(1-u_t)}{u_t(1-p_t)} \cdot \prod_{t:q_t=1} \frac{1-p_t}{1-u_t}$$

The left product is still over query terms found in the document, but the right product is now over all query terms. That means that this right product is a constant for a particular query, just like the odds  $O(R|\vec{q})$ . So the only quantity that needs to be estimated to rank documents for relevance to a query is the left product. We can equally rank documents by the logarithm of this term, since log is a monotonic function. The resulting quantity used for ranking is called the *Retrieval Status Value* (RSV) in this model:

RETRIEVAL STATUS  
VALUE

$$(11.17) \quad RSV_d = \log \prod_{t:x_t=q_t=1} \frac{p_t(1-u_t)}{u_t(1-p_t)} = \sum_{t:x_t=q_t=1} \log \frac{p_t(1-u_t)}{u_t(1-p_t)}$$

So everything comes down to computing the *RSV*. Define  $c_t$ :

$$(11.18) \quad c_t = \log \frac{p_t(1-u_t)}{u_t(1-p_t)} = \log \frac{p_t}{(1-p_t)} + \log \frac{1-u_t}{u_t}$$

ODDS RATIO

The  $c_t$  terms are log odds ratios for the terms in the query. We have the odds of the term appearing if the document is relevant ( $p_t/(1-p_t)$ ) and the odds of the term appearing if the document is nonrelevant ( $u_t/(1-u_t)$ ). The *odds ratio* is the ratio of two such odds, and then we finally take the log of that quantity. The value will be 0 if a term has equal odds of appearing in relevant and nonrelevant documents, and positive if it is more likely to appear in relevant documents. The  $c_t$  quantities function as term weights in the model, and the document score for a query is  $RSV_d = \sum_{x_t=q_t=1} c_t$ . Operationally, we

```
# IR11A.py CS5154/6054 cheng 2022
# TfidfVectorizer is used to generate vocabulary
# a random term is the query and the top 5 cosine similarity
# in Tfidf are considered as the initial pseudo relevant for
# probabilistic pseudo relevance feedback (IIR 11.3.4)
# then pt, ut, ct are computed for terms and documents are ranked
# with sum of ct for t in docs
# Usage: python IR11A.py
```

```
import re
import numpy as np
import random
import math
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
pseudorelevant = 5
```

```
f = open("bible.txt", "r")
docs = f.readlines()
f.close()
```

```
tfidf = TfidfVectorizer(binary=True, max_df=0.04, min_df=8)
dt = tfidf.fit_transform(docs)
docsets = [set(d.indices) for d in dt]
```

```
N = len(docs)
terms = list(tfidf.vocabulary_)
T = len(terms)
```

```
query = random.choice(terms)
print(query, tfidf.vocabulary_.get(query))
q = tfidf.transform([query])
sim = cosine_similarity(q[0], dt)
relevantset = set()
for d in np.argsort(sim[0])[::-1][0:pseudorelevant]:
    print(sim[0][d], docs[d])
    relevantset.add(d)
```

```
print(relevantset)
```

```
dfs = np.zeros(T, dtype=int)
```

```
for d in range(N):
```

```
    for t in docsets[d]:
```

```
        dfs[t] = dfs[t] + 1
```

```
ct = np.zeros(T)
```

```
for t in range(T):
```

```
    Vt = 0
```

```
    for d in relevantset:
```

```
        if t in docsets[d]:
```

```
            Vt = Vt + 1
```

```
pt = (Vt + 0.5)/(pseudorelevant + 1.0)
```

```
ut = (dfs[t] - Vt + 0.5)/(N - pseudorelevant + 1.0)
```

```
ct[t] = math.log(pt/(1 - pt) * (1 - ut)/ut)
```

```
rsv = np.zeros(N)
```

```
for d in range(N):
```

```
    for t in docsets[d]:
```

```
        rsv[d] = rsv[d] + ct[t]
```

```
relevantset.clear()
```

```
for d in np.argsort(rsv)[::-1][0:pseudorelevant]:
```

```
    print(rsv[d], docs[d])
```

```
    relevantset.add(d)
```

```
print(relevantset)
```