Introduction to Information Retrieval

Evaluation

Chris Manning and Pandu Nayak

CS276 – Information Retrieval and Web Search
Thanks to your stellar performance in CS276, you quickly rise to VP of Search at internet retail giant nozama.com. Your boss brings in her nephew Sergey, who claims to have built a better search engine for nozama. Do you

- Laugh derisively and send him to rival Tramlaw Labs?
- Counsel Sergey to go to Stanford and take CS276?
- Try a few queries on his engine and say “Not bad”?
- …?
What could you ask Sergey?

- How fast does it index?
  - Number of documents/hour
  - Incremental indexing – nozama adds 10K products/day

- How fast does it search?
  - Latency and CPU needs for nozama’s 5 million products

- Does it recommend related products?

- This is all good, but it says nothing about the quality of Sergey’s search
  - You want nozama’s users to be happy with the search experience
How do you tell if users are happy?

- Search returns products relevant to users
  - How do you assess this at scale?
- Search results get clicked a lot
  - Misleading titles/summaries can cause users to click
- Users buy after using the search engine
  - Or, users spend a lot of $ after using the search engine
- Repeat visitors/buyers
  - Do users leave soon after searching?
  - Do they come back within a week/month/... ?
Happiness: elusive to measure

- Most common proxy: relevance of search results
  - Pioneered by Cyril Cleverdon in the Cranfield Experiments

- But how do you measure relevance?
Measuring relevance

- Three elements:
  1. A benchmark document collection
  2. A benchmark suite of queries
  3. An assessment of either Relevant or Nonrelevant for each query and each document
So you want to measure the quality of a new search algorithm?

- Benchmark documents – nozama’s products
- Benchmark query suite – more on this
- Judgments of document relevance for each query

5 million nozama.com products

50,000 sample queries

Relevance judgment
Relevance judgments

- Binary (relevant vs. non-relevant) in the simplest case
  - More nuanced relevance levels also used (0, 1, 2, 3 ...)
- What are some issues already?
- 5 million times 50K takes us into the range of a quarter trillion judgments
  - If each judgment took a human 2.5 seconds, we’d still need $10^{11}$ seconds, or nearly $300$ million if you pay people $10 per hour to assess
  - 10K new products per day
Crowd source relevance judgments?

- Present query-document pairs to low-cost labor on online crowd-sourcing platforms
  - Hope that this is cheaper than hiring qualified assessors
- Lots of literature on using crowd-sourcing for such tasks
  - You get fairly good signal, but the variance in the resulting judgments is quite high
What else?

- Still need test queries
  - Must be germane to docs available
  - Must be representative of actual user needs
  - Random query terms from the documents are not a good idea
  - Sample from query logs if available

- Classically (non-Web)
  - Low query rates – not enough query logs
  - Experts hand-craft “user needs”
### Early public test Collections (20th C)

<table>
<thead>
<tr>
<th>Collection</th>
<th>NDocs</th>
<th>NQrys</th>
<th>Size (MB)</th>
<th>Term/Doc</th>
<th>Q-D RelAss</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADI</td>
<td>82</td>
<td>35</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AIT</td>
<td>2109</td>
<td>14</td>
<td>2</td>
<td>400</td>
<td>&gt;10,000</td>
</tr>
<tr>
<td>CACM</td>
<td>3204</td>
<td>64</td>
<td>2</td>
<td>24.5</td>
<td></td>
</tr>
<tr>
<td>CISI</td>
<td>1460</td>
<td>112</td>
<td>2</td>
<td>46.5</td>
<td></td>
</tr>
<tr>
<td>Cranfield</td>
<td>1400</td>
<td>225</td>
<td>2</td>
<td>53.1</td>
<td></td>
</tr>
<tr>
<td>LISA</td>
<td>5872</td>
<td>35</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Medline</td>
<td>1033</td>
<td>30</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NPL</td>
<td>11,429</td>
<td>93</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OSHMED</td>
<td>34,856</td>
<td>106</td>
<td>400</td>
<td>250</td>
<td>16,140</td>
</tr>
<tr>
<td>Reuters</td>
<td>21,578</td>
<td>672</td>
<td>28</td>
<td>131</td>
<td></td>
</tr>
<tr>
<td>TREC</td>
<td>740,000</td>
<td>200</td>
<td>2000</td>
<td>89-3543</td>
<td>» 100,000</td>
</tr>
</tbody>
</table>

**Typical TREC**

Recent datasets: 100s of million web pages (GOV, ClueWeb, ...)

---

*Sec. 8.5*

*Introduction to Information Retrieval*
Now we have the basics of a benchmark

- Let’s review some evaluation measures
  - Precision
  - Recall
  - DCG
  - ...

Introduction to Information Retrieval
Evaluating an IR system

- Note: **user need** is translated into a **query**
- Relevance is assessed relative to the **user need**, not the **query**
- E.g., **Information need**: *My swimming pool bottom is becoming black and needs to be cleaned.*
- **Query**: *pool cleaner*
- Assess whether the doc addresses the underlying need, not whether it has these words
Unranked retrieval evaluation:
Precision and Recall – recap from IIR 8/video

- **Binary assessments**

  **Precision:** fraction of retrieved docs that are relevant = $P(\text{relevant} | \text{retrieved})$

  **Recall:** fraction of relevant docs that are retrieved
  
  $= P(\text{retrieved} | \text{relevant})$

<table>
<thead>
<tr>
<th></th>
<th>Relevant</th>
<th>Nonrelevant</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retrieved</td>
<td>tp</td>
<td>fp</td>
</tr>
<tr>
<td>Not Retrieved</td>
<td>fn</td>
<td>tn</td>
</tr>
</tbody>
</table>

- Precision $P = \frac{tp}{tp + fp}$
- Recall $R = \frac{tp}{tp + fn}$
Rank-Based Measures

- Binary relevance
  - Precision@K (P@K)
  - Mean Average Precision (MAP)
  - Mean Reciprocal Rank (MRR)

- Multiple levels of relevance
  - Normalized Discounted Cumulative Gain (NDCG)
Precision@K

- Set a rank threshold K
- Compute % relevant in top K
- Ignores documents ranked lower than K

Ex:
- Prec@3 of 2/3
- Prec@4 of 2/4
- Prec@5 of 3/5

In similar fashion we have Recall@K
A precision-recall curve

Lots more detail on this in the Canvas video
Mean Average Precision

- Consider rank position of each relevant doc
  - $K_1, K_2, \ldots K_R$

*Compute Precision@K for each $K_1, K_2, \ldots K_R$*

- Average precision = average of P@K

*Ex:* has AvgPrec of \( \frac{1}{3} \left( \frac{1}{1} + \frac{2}{3} + \frac{3}{5} \right) \approx 0.76 \)

- MAP is Average Precision across multiple queries/rankings
Average Precision

Ranking #1:

Recall: 0.17 0.17 0.33 0.5 0.67 0.83 0.83 0.83 0.83 1.0
Precision: 1.0 0.5 0.67 0.75 0.8 0.83 0.71 0.63 0.56 0.6

Ranking #2:

Recall: 0.0 0.17 0.17 0.17 0.33 0.5 0.67 0.67 0.83 1.0
Precision: 0.0 0.5 0.33 0.25 0.4 0.5 0.57 0.5 0.56 0.6

Ranking #1: $(1.0 + 0.67 + 0.75 + 0.8 + 0.83 + 0.6)/6 = 0.78$

Ranking #2: $(0.5 + 0.4 + 0.5 + 0.57 + 0.56 + 0.6)/6 = 0.52$
MAP

= relevant documents for query 1

<table>
<thead>
<tr>
<th>Ranking #1</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Recall</td>
<td>0.2</td>
<td>0.2</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
<td>0.6</td>
<td>0.6</td>
<td>0.6</td>
<td>0.8</td>
<td>1.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Precision</td>
<td>1.0</td>
<td>0.5</td>
<td>0.67</td>
<td>0.5</td>
<td>0.4</td>
<td>0.5</td>
<td>0.43</td>
<td>0.38</td>
<td>0.44</td>
<td>0.5</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

= relevant documents for query 2

<table>
<thead>
<tr>
<th>Ranking #2</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Recall</td>
<td>0.0</td>
<td>0.33</td>
<td>0.33</td>
<td>0.33</td>
<td>0.67</td>
<td>0.67</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Precision</td>
<td>0.0</td>
<td>0.5</td>
<td>0.33</td>
<td>0.25</td>
<td>0.4</td>
<td>0.33</td>
<td>0.43</td>
<td>0.38</td>
<td>0.33</td>
<td>0.3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

average precision query 1 = (1.0 + 0.67 + 0.5 + 0.44 + 0.5)/5 = 0.62
average precision query 2 = (0.5 + 0.4 + 0.43)/3 = 0.44

mean average precision = (0.62 + 0.44)/2 = 0.53
Mean average precision

- If a relevant document never gets retrieved, we assume the precision corresponding to that relevant doc to be zero
- MAP is macro-averaging: each query counts equally
- Now perhaps most commonly used measure in research papers
- Good for web search?
- MAP assumes user is interested in finding many relevant documents for each query
- MAP requires many relevance judgments in text collection
BEYOND BINARY RELEVANCE
Discounted Cumulative Gain

- Popular measure for evaluating web search and related tasks

- Two assumptions:
  - Highly relevant documents are more useful than marginally relevant documents
  - the lower the ranked position of a relevant document, the less useful it is for the user, since it is less likely to be examined
Discounted Cumulative Gain

- Uses \textit{graded relevance} as a measure of usefulness, or \textit{gain}, from examining a document.
- Gain is accumulated starting at the top of the ranking and may be reduced, or \textit{discounted}, at lower ranks.
- Typical discount is $1/\log (rank)$.
  - With base 2, the discount at rank 4 is $1/2$, and at rank 8 it is $1/3$. 
Summarize a Ranking: DCG

- What if relevance judgments are in a scale of [0,r]? r>2
- Cumulative Gain (CG) at rank n
  - Let the ratings of the n documents be r₁, r₂, … rₙ (in ranked order)
  - CG = r₁+r₂+…rₙ
- Discounted Cumulative Gain (DCG) at rank n
  - DCG = r₁ + r₂/log₂2 + r₃/log₂3 + … rₙ/log₂n
    - We may use any base for the logarithm
Discounted Cumulative Gain

- DCG is the total gain accumulated at a particular rank $p$:

$$DCG_p = rel_1 + \sum_{i=2}^{p} \frac{rel_i}{\log_2 i}$$

- Alternative formulation:

$$DCG_p = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{\log(1+i)}$$

- used by some web search companies
- emphasis on retrieving highly relevant documents
DCG Example

- 10 ranked documents judged on 0–3 relevance scale:
  3, 2, 3, 0, 0, 1, 2, 2, 3, 0

- discounted gain:
  3, 2/1, 3/1.59, 0, 0, 1/2.59, 2/2.81, 2/3, 3/3.17, 0
  = 3, 2, 1.89, 0, 0, 0.39, 0.71, 0.67, 0.95, 0

- DCG:
  3, 5, 6.89, 6.89, 6.89, 7.28, 7.99, 8.66, 9.61, 9.61
NDCG for summarizing rankings

- Normalized Discounted Cumulative Gain (NDCG) at rank $n$
  - Normalize DCG at rank $n$ by the DCG value at rank $n$ of the ideal ranking
  - The ideal ranking would first return the documents with the highest relevance level, then the next highest relevance level, etc
- Normalization useful for contrasting queries with varying numbers of relevant results
- NDCG is now quite popular in evaluating Web search
NDCG - Example

4 documents: d₁, d₂, d₃, d₄

<table>
<thead>
<tr>
<th>i</th>
<th>Ground Truth</th>
<th>Ranking Function₁</th>
<th>Ranking Function₂</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Document Order</td>
<td>rᵢ</td>
<td>Document Order</td>
</tr>
<tr>
<td>1</td>
<td>d₄</td>
<td>2</td>
<td>d₃</td>
</tr>
<tr>
<td>2</td>
<td>d₃</td>
<td>2</td>
<td>d₄</td>
</tr>
<tr>
<td>3</td>
<td>d₂</td>
<td>1</td>
<td>d₂</td>
</tr>
<tr>
<td>4</td>
<td>d₁</td>
<td>0</td>
<td>d₁</td>
</tr>
</tbody>
</table>

NDCG<sub>GT</sub>=1.00, NDCG<sub>RF₁</sub>=1.00, NDCG<sub>RF₂</sub>=0.9203

\[
DCG_{GT} = 2 + \left( \frac{2}{\log_2 2} + \frac{1}{\log_2 3} + \frac{0}{\log_2 4} \right) = 4.6309 \\
DCG_{RF₁} = 2 + \left( \frac{2}{\log_2 2} + \frac{1}{\log_2 3} + \frac{0}{\log_2 4} \right) = 4.6309 \\
DCG_{RF₂} = 2 + \left( \frac{1}{\log_2 2} + \frac{2}{\log_2 3} + \frac{0}{\log_2 4} \right) = 4.2619 \\
MaxDCG = DCG_{GT} = 4.6309
What if the results are not in a list?

- Suppose there’s only one Relevant Document
- Scenarios:
  - known-item search
  - navigational queries
  - looking for a fact
- Search duration $\sim$ Rank of the answer
  - measures a user’s effort
Mean Reciprocal Rank

- Consider rank position, $K$, of first relevant doc
  - Could be – only clicked doc

- Reciprocal Rank score = \( \frac{1}{K} \)

- MRR is the mean RR across multiple queries
Human judgments are

- Expensive
- Inconsistent
  - Between raters
  - Over time
- Decay in value as documents/query mix evolves
- Not always representative of “real users”
  - Rating vis-à-vis query, don’t know underlying need
  - May not understand meaning of terms, etc.
- So – what alternatives do we have?
USING USER CLICKS
Introduction to Information Retrieval

User Behavior

- Search Results for “CIKM” (in 2009!)

Taken with slight adaptation from Fan Guo and Chao Liu’s 2009/2010 CIKM tutorial: Statistical Models for Web Search: Click Log Analysis
User Behavior

- Adapt ranking to user clicks?
What do clicks tell us?

- Tools needed for non-trivial cases

Strong position bias, so absolute click rates unreliable
Eye-tracking User Study
Higher positions receive more user attention (eye fixation) and clicks than lower positions.

This is true even in the extreme setting where the order of positions is reversed.

“Clicks are informative but biased”.

[Joachims+07]
Relative vs absolute ratings

Hard to conclude Result1 > Result3
Probably can conclude Result3 > Result2
Evaluating pairwise relative ratings

- Pairs of the form: DocA better than DocB for a query
  - Doesn’t mean that DocA relevant to query
- Now, rather than assess a rank-ordering wrt per-doc relevance assessments ...
- Assess in terms of conformance with historical pairwise preferences recorded from user clicks
- BUT!
- Don’t learn and test on the same ranking algorithm
  - I.e., if you learn historical clicks from nozama and compare Sergey vs nozama on this history ...
Comparing two rankings via clicks (Joachims 2002)

Query: [support vector machines]

<table>
<thead>
<tr>
<th>Ranking A</th>
<th>Ranking B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kernel machines</td>
<td>Kernel machines</td>
</tr>
<tr>
<td>SVM-light</td>
<td>SVMs</td>
</tr>
<tr>
<td>Lucent SVM demo</td>
<td>Intro to SVMs</td>
</tr>
<tr>
<td>Royal Holl. SVM</td>
<td>Archives of SVM</td>
</tr>
<tr>
<td>SVM software</td>
<td>SVM-light</td>
</tr>
<tr>
<td>SVM tutorial</td>
<td>SVM software</td>
</tr>
</tbody>
</table>
Interleave the two rankings

This interleaving starts with B

<table>
<thead>
<tr>
<th>Kernel machines</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kernel machines</td>
</tr>
<tr>
<td>SVMs</td>
</tr>
<tr>
<td>SVM-light</td>
</tr>
<tr>
<td>Intro to SVMs</td>
</tr>
<tr>
<td>Lucent SVM demo</td>
</tr>
<tr>
<td>Archives of SVM</td>
</tr>
<tr>
<td>Royal Holl. SVM</td>
</tr>
<tr>
<td>SVM-light</td>
</tr>
</tbody>
</table>

…
Remove duplicate results

- Kernel machines
  - SVMs
    - SVM-light
  - Intro to SVMs
  - Lucent SVM demo
  - Archives of SVM
  - Royal Holl. SVM
  - SVM-light
  ...

---

*Introduction to Information Retrieval*
Count user clicks

Ranking A: 3
Ranking B: 1

Kernel machines

SVMs

SVM-light

Intro to SVMs

Lucent SVM demo

Archives of SVM

Royal Holl. SVM

SVM-light

…

Clacks

A, B

A

A
Interleaved ranking

- Present interleaved ranking to users
  - Start randomly with ranking A or ranking B to even out presentation bias

- Count clicks on results from A versus results from B

- Better ranking will (on average) get more clicks
A/B testing at web search engines

- Purpose: Test a single innovation

- Prerequisite: You have a large search engine up and running.

- Have most users use old system

- Divert a small proportion of traffic (e.g., 0.1%) to an experiment to evaluate an innovation
  - Interleaved experiment
  - Full page experiment
Facts/entities (what happens to clicks?)

mount everest height

29,029' (8,848 m)
Mount Everest, Elevation

Mount Everest - Wikipedia, the free encyclopedia
https://en.wikipedia.org/wiki/Mount_Everest
By the same measure of base to summit, Mount McKinley, in Alaska, is also taller than Everest. Despite its height above sea level of only 6,193.6 m (20,320 ft), ...

List of deaths on eight - List of people who died ... - Timeline of climbing Mount

Facts About Mt. Everest - Scholastic
teacher.scholastic.com/activities/hillary/archive/evfacts.htm
Number of people to successfully climb Mt. Everest: 660. Number of
Recap

- Benchmarks consist of
  - Document collection
  - Query set
  - Assessment methodology

- Assessment methodology can use raters, user clicks, or a combination
  - These get quantized into a *goodness measure* – Precision/NDCG etc.
  - Different engines/algorithms compared on a benchmark together with a *goodness measure*. 
User behavior

- User behavior is an intriguing source of relevance data
  - Users make (somewhat) informed choices when they interact with search engines
  - Potentially a lot of data available in search logs

- But there are significant caveats
  - User behavior data can be very noisy
  - Interpreting user behavior can be tricky
  - Spam can be a significant problem
  - Not all queries will have user behavior
Incorporating user behavior into ranking algorithm

- Incorporate user behavior features into a ranking function like BM25F
  - But requires an understanding of user behavior features so that appropriate $V_j$ functions are used

- Incorporate user behavior features into learned ranking function

- Either of these ways of incorporating user behavior signals improve ranking