Web Search II:

State-of-the-Art Systems
Classic Retrieval Models Rely on Similar Ranking Features

- Term frequency, or $tf$
- Inverse document frequency, or $idf$
- Penalize repeated terms
- Penalize document length
- ... and variants thereof

**Example:** Vector Space Model

And several other models: BM25, Language Models, ...
We Would Like To Use More, Richer Features

- Query terms **proximity** and **order**
- Matching text **style**
- URL **depth**
- Site **spam score**
- Content **readability**
- Page **freshness**, “location,” #images, language, format...
- ...

“Google has over 200 such features” - Amit Singhal (former head of Search at Google) - NYTimes (06/03/2008)
How to Exploit All Ranking Features? Learning to Rank

- Combine **hundreds (and thousands!) of features**
  - Typically implemented as weighted linear combination
  - Weights learned from training set
- Builds on **effective machine learning methods**
- Adapts to different **user input**

Three main approaches:

- **Pointwise**: Is document $d$ relevant for query $q$?
- **Pairwise**: Is document $d_k$ more relevant than document $d_j$ for query $q$?
- **Listwise**: Is ranking $\pi_k$ better than ranking $\pi_j$ for query $q$?
The Learning to Rank Framework

**Training**
- \( q_1 \)
- \( d_{1}^{1} \)
- \( y_1 \)
- \( d_{2}^{1} \)
- ...  
- \( d_{n}^{1} \)

- \( q_2 \)
- \( d_{1}^{2} \)
- \( y_2 \)
- \( d_{2}^{2} \)
- ...
- \( d_{m}^{2} \)

- \( q_t \)
- \( d_{1}^{t} \)
- \( y_t \)
- \( d_{2}^{t} \)
- ...
- \( d_{k}^{t} \)

**Learning to Rank Method**

**Test example**
- \( q \)
- \( d_{1} \)
- \( d_{2} \)
- ...
- \( d_{n} \)

**Learned model**
- \( h(x) \)

**Evaluation**

**Ranking System**
- \( h(x) \)

**Ranked example**
- \( q \)
- \( d_{1} \)
- \( d_{2} \)
- ...
- \( d_{n} \)
Relevance Judgements Come in Different Flavors

- Degree of relevance
  - Binary: relevant vs. irrelevant
  - Multiple ordered categories: Perfect > Excellent > Good > Fair > Bad
- Pairwise preference
  - Document $d_i$ is more relevant than document $d_j$
- Total order
  - Documents are ranked as $\{d_p, d_j, d_k, \ldots\}$ according to their relevance

Apply in all learning-to-rank approaches (with some extra work)
## Pointwise Learning to Rank Builds on Other Learning Tasks

<table>
<thead>
<tr>
<th>Input Space</th>
<th>Output Space</th>
<th>Hypothesis Space</th>
<th>Loss Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regression</td>
<td>Single documents $y_j$</td>
<td>Real values</td>
<td>Scoring function $f(x)$</td>
</tr>
<tr>
<td>Classification</td>
<td>Non-ordered Categories</td>
<td>Ordinal categories</td>
<td></td>
</tr>
<tr>
<td>Ordinal Regression</td>
<td></td>
<td>Ordinal categories</td>
<td></td>
</tr>
</tbody>
</table>

\[
L(f; x_j, y_j)
\]

Table Courtesy of Tie-Yan Liu
Example Pointwise Learning to Rank Method: Classification via Linear SVM [Nallapati 2004]

- Let $h(\phi(d,q)) = \mathbf{w} \cdot \phi(d,q) + b$
  - $\phi$: \{document, query\} $\rightarrow$ feature space
  - Training:
    - $h(x) \geq 1$ for relevant document
    - $h(x) \leq -1$ for non-relevant document
  - Testing:
    - $d$ relevant for $q$ iff $h(\phi(d,q)) \geq 0$

- Some nice properties
  - Generalize to more features
  - Adopt advances in weight learning
Limitations of Pointwise Learning to Rank

- Requires extra work for **class imbalance**
  - $|\text{non-relevant instances}| \gg |\text{relevant instances}|$
  - Solution: Over or under sampling

- Ignores **relation between documents**

- Queries with large number of documents **dominate training**
  - Weighting training instances

- Loss function **unsuited for ranking**
  - Ignores document position (i.e., low-ranked as important as top-ranked)
  - Different from ranking metrics (e.g., MAP, NDCG)
## Pairwise Learning to Rank Learns Document Preference

<table>
<thead>
<tr>
<th>Input Space</th>
<th>Document pairs ((x_u, x_v))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output Space</td>
<td>Preference (y_{u,v} \in {+1,-1})</td>
</tr>
<tr>
<td>Hypothesis Space</td>
<td>Preference function (h(x_u, x_v) = 2 \cdot I_{{f(x_u) &gt; f(x_v)}} - 1)</td>
</tr>
<tr>
<td>Loss Function</td>
<td>Pairwise classification loss (L(h; x_u, x_v, y_{u,v}))</td>
</tr>
</tbody>
</table>

Table Courtesy of Tie-Yan Liu
Example of Pairwise Learning to Rank
Method: RankSVM [Herbrich et al. 1999, 2000; Joachims et al. 2002]

- Let $h(\phi(d_i, q), \phi(d_j, q)) = \mathbf{w} \cdot (\phi(d_i, q) - \phi(d_j, q)) + b$
  - $\phi$: \{document, query\} $\rightarrow$ feature space
  - Training:
    - $h(x) \geq 1$ if $d_i$ is more relevant than $d_j$
    - $h(x) \leq -1$ if $d_i$ is less relevant than $d_j$
  - Testing:
    - $d_i$ ranks higher than $d_j$ iff $h(x) \geq 0$
    - Trick: Enough to rank by $w \cdot \phi(d, q)$

- Some nice properties
  - Efficient via stochastic gradient descent
  - SVM kernel tricks still apply
Limitations of Pairwise Learning to Rank

● Queries with large number documents still dominate training
  ○ More skewed than with pointwise methods (see figure)
  ○ Solution: Weight training instances, use top-N documents

● Better loss function, still unsuited for ranking
  ○ Ignores document position (i.e., low-ranked as important as top-ranked)
  ○ Different from ranking metrics (e.g., MAP, NDCG)

Queries from commercial search engine
## Listwise Learning to Rank Consists of Two Main Families of Approaches

<table>
<thead>
<tr>
<th>Input Space</th>
<th>Document set $\mathbf{x} = {x_j}_{j=1}^m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output Space</td>
<td>Permutation $\pi_y$</td>
</tr>
<tr>
<td>Hypothesis Space</td>
<td>$h(\mathbf{x}) = \text{sort} \circ f(\mathbf{x})$</td>
</tr>
<tr>
<td>Loss Function</td>
<td>Listwise loss $L(h; \mathbf{x}, \pi_y)$</td>
</tr>
<tr>
<td></td>
<td>1-surrogate measure $L(h; \mathbf{x}, \mathbf{y})$</td>
</tr>
</tbody>
</table>

**The Listwise Approach**

- **Listwise Loss Minimization**
- **Direct Optimization of IR Measure**

*Table Courtesy of Tie-Yan Liu*
Listwise Learning to Rank: Discussion

- **Listwise loss minimization**
  - Optimize loss function defined on permutations
  - Example: KL-Divergence over probability of permutations

- **Direct optimization of IR measure**
  - Optimize over evaluation metric
  - Better yet: Over approximation or bound, to be continuous and differentiable

Some limitations

- **Large number of potential lists compromise scalability**
  - Trick: Focus on top-N retrieved documents

- No substantial benefit over pairwise approaches (so far)
• **Systematic and scalable** methods to combine ranking features
• **Family of approaches** for all ranking settings
  ○ Pointwise: Is document $d$ relevant for query $q$?
  ○ Pairwise: Is document $d_k$ more relevant than document $d_j$ for query $q$?
  ○ Listwise: Is ranking $\pi_k$ better than ranking $\pi_j$ for query $q$?
Relevance Is Not the Only Ranking Goal: Novelty and Diversity

- Returned results need to be **novel** with respect to:
  - *User’s knowledge?* Difficult problem
  - *Other returned results?* Easier, focus of IR

- Ambiguous queries need **diverse** results:
  - *Topic:* Queries carry multiple meanings (e.g., eclipse, jaguar, rio, “take me home”)
  - *Intent:* Queries seek different intents (e.g., “java 9” -> features, download)
  - *Time and Space:* Relevance is context-dependent (e.g., “olympic games,” “hotels nearby”)

Common approach: Focus on diversity, as it makes results novel
Example: Novelty and Diversity via Maximum Marginal Relevance (MMR)

- Operates on document similarity
- Ignores different query topics (implicit diversification method)
- Selects next document according to:

\[
\arg\max_{d \not\in S} \left( \lambda \cdot \text{sim}(q, d) - (1 - \lambda) \cdot \max_{d' \in S} \text{sim}(d', d) \right)
\]

Query relevance & Content similarity

Not necessarily same sim function
And images, videos, shopping, ads...
Not a One-Size-Fits-All Solution: Searching on Different Platforms

Search capabilities on **most connected devices** bring unique ranking challenges across all building blocks:

- **Smartphone/tablet:** Small display, unsupported Web documents
- **Voice assistants:** Natural language queries, no display, one-shot answer, dialog
- **Smart watch, smart TV, car display,** ...

**Compare classic IR system vs. Assistant:**

- “where can I grab a bite with my friends tonight?”
- “what is 23 by 47?”
Is This Search Engine Any Good? Evaluating Search Engines

"If you can not measure it, you can not improve it." - Lord Kelvin

Retrieval evaluation goals:

- **Practicality**: Can I afford this evaluation method in practice?
- **Correctness**: Does evaluation agree with real user judgement?
- **Efficiency**: Is this the best use of my resources?

Focus of second part of class

Main evaluation methods: **Offline** and **Online** evaluation
The most important measure of a search engine is the quality of its search results. While a complete user evaluation is beyond the scope of this paper, our own experience with Google has shown it to produce better results than the major commercial search engines for most searches.

Very practical, at least! :-)

A true test of the quality of a search engine would involve an extensive user study or results analysis which we do not have room for here. Instead, we invite the reader to try Google for themselves at http://google.stanford.edu.

[Sergey Brin and Larry Page, “The anatomy of a large-scale hypertextual web search engine.” WWW ’98]
Traditionally, IR Evaluation Only Happened Offline

**The Cranfield style**
- Build test collection of relevant queries and results relevance judgements
- Measure systems effectiveness (e.g., MAP, NDCG, Precision@5)
- Compare metrics and evaluate statistical significance

How about retrieval evaluation goals?
- **Practical** and **efficient** when existing test collection (e.g., TREC, LETOR) includes enough representative queries
- **Correct** when judgements and metrics match our user’s hypotheses
But Offline Evaluation Has Limitations

- Users and judges may disagree
- Ambiguous queries may be hard to judge
- Relevance definition may change over time
- New data may be costly and time-consuming

Personalized search? Specialized documents?
Most popular intent? Missing intent?
Performance on “olympic games” query?
Crowdsourcing helps! (see example next)
An Example Task on Crowdsourced Search Engine Evaluation

**Query:** “personalized search”  
**Task:** Rate relevance of individual results (5-star scale) and ranking comparison (5-level slider)

<table>
<thead>
<tr>
<th>Base</th>
<th>Experiment</th>
</tr>
</thead>
</table>
| **Personalized Search** | **ACM SIGIR Special Interest Group on Information Retrieval Home Page**  
Welcome to the ACM SIGIR Web site. ACM SIGIR addresses issues ranging from theory to user demands in the application of computers to the acquisition, organization ...  
www.sigir.org |  
**Zakla – Personalized Social Search Engine - edit search, firefox**  
Zakla, unlike other search engines, can be considered as a “Personal Research Assistant” with its ability to dig deeper to get the required information and techop.com/2009/10/15/zakla-personalized-social-search-engine |  
**Exploring folksonomy for personalized search**  
We propose a personalized search framework to utilize folksonomy for ... SIGIR ’08 Proceedings of the 31st annual international ACM SIGIR conference on Research ...  
portal.acm.org/citation.cfm?id=1390363 |
| Related Searches for personalized search research |  
**Ontology-based Personalized S...**  
**Bing Personalized Search**  
**Personalized Search Engines**  
**Personalization Business** |  
Supported by: ACM SIGIR Special Interest Group on Information Retrieval  
Supported by: ACM SIGIR Special Interest Group on Information Retrieval  
Supported by: ACM SIGIR Special Interest Group on Information Retrieval  
Supported by: ACM SIGIR Special Interest Group on Information Retrieval |  
**Personalized search - Wikipedia, the free encyclopedia**  
Personalized search refers to search experiences. A specific groups of people, personalized search depends on a user profile that is unique to the individual. Research ...  
en.wikipedia.org/wiki/Personalized_Search |  
**Personalized search - Wikipedia, the free encyclopedia**  
Personalized search refers to search experiences. Research systems that personalize search results model their users in ... to personalize global Web search”. SIGIR 287 ...  
en.wikipedia.org/wiki/Personalized_Search |
“Online evaluation is evaluation of a fully functioning system based on implicit measurement of real users’ experiences of the system in a natural usage environment” [Hofmann et al.]

- Key assumption: Observable user behavior reflects relevance
- Advantages: Low cost, large scale, instant feedback
- Complements offline evaluation
- Why not explicit? Users that provide explicit feedback are far from random
Online Evaluation Metrics are Defined By Quality and Granularity

<table>
<thead>
<tr>
<th>Quality</th>
<th>Granularity</th>
<th>Document</th>
<th>Ranking</th>
<th>Session</th>
<th>User</th>
</tr>
</thead>
<tbody>
<tr>
<td>Absolute</td>
<td></td>
<td>• Click-through rate</td>
<td>• Click rank (recip.)</td>
<td>• Num. of queries</td>
<td>• Absence time</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Dwell time</td>
<td>• CTR@k</td>
<td>• Unique queries</td>
<td>• Loyalty (queries, daily sessions, success rate)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Mouse movement</td>
<td>• pSkip</td>
<td>• Length</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Characteristics</td>
<td>• Time to click</td>
<td>• End action type</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Abandonment</td>
<td>• Mouse movement</td>
<td></td>
</tr>
<tr>
<td>Relative</td>
<td></td>
<td>• Click-skip</td>
<td>• <strong>Interleaving (see next)</strong></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• FairPairs</td>
<td></td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

26
Main Building Block of Online Evaluation: Controlled Experiment

**Goal**: Test if observed changes in user behavior are caused by system changes

Three-step approach:
- Define hypothesis
- **Deploy experiment to collect metrics**
- Evaluate statistical significance

<table>
<thead>
<tr>
<th>Control (C): Current system</th>
<th>Treatment (T): Experimental system</th>
</tr>
</thead>
</table>

**Experiment Designs**
- **Between-subject Experiment**: Assign user randomly to C or T
- **Within-subject Experiment**: Expose user to both C and T (interleaved comparison)
Within-Subject Experiment: Interleaving

Given a query $q$, produce rankings with $C$ and $T$

1. Merge the rankings using **mixing policy**
2. Present merged ranking and collect interactions
3. Choose winning ranking using **scoring rule**

Advantages:
- Correlates with offline metrics
- Reduce data requirements considerably

Limitations:
- Are not generally applicable (e.g., UI experiments)

Example: Team-Draft Interleaving

- **Mixing policy:** Rounds of doc selection
  - Randomize rankers in each round
  - Each ranker selects its highest (not merged) ranked document
- **Scoring rule:** Prefer system with more clicks

[Radlinski et. al, CIKM ’08]
Other Aspects to Consider When Performing Online Evaluation

- Randomize at user level to avoid confounding variables
- Adjust for unit of experimentation (user vs. query)
- Increase throughput by running experiments in parallel
- Account for network effects when necessary
- Account for sources of bias:
  - Position, presentation, trust, quality-of-context bias
- Never forget high-level system goals
Beyond Searching Web Results: User Interface and Platforms Evaluation

- **User interface evaluation**
  - Between-subject experiments are best suited
  - New UI elements require interaction measurement

- **Other platforms evaluation**
  - Mobile device: Taps, context switch
  - Voice assistant: Acoustics, new features, dialog
Summing Up Search Engine Evaluation

- Evaluation is key for search engine improvement
- Complimentary offline and online evaluation
  - Practicality
  - Correctness
  - Efficiency

Should we deploy a new system?
- Perform first round of offline evaluation
- Perform second (final) round of online evaluation
Thanks!