



The Machine Learned Ranking Story

Jan Pedersen
18 August 2008



Overview

- Ranking as Function Estimation
- Historical Development of MLR
 - Alta Vista
 - Yahoo/Inktomi
- Challenges
- Intent-based Ranking
 - Moving forward



The Ranking Problem

Given a result set for query q , find an ordering $r = \{d_i\}_{i=1}^m$ that maximizes a ranking measure $L(r, q)$

Typically

$$L(r, q) = \sum_{i=1}^m w_i v(d_i, q)$$

where $v(d, q)$ is a measure of relevance, either binary or graded, and w_i are positive decreasing weights



Ranking Metrics

- Precision

$$P(r, q) = \frac{1}{m} \sum_{i=1}^m v(d_i, q)$$

- Recall

$$R(r, q) = \frac{\sum_{i=1}^m v(d_i, q)}{\sum_{i=1}^{\infty} v(d_i, q)}$$

- Average Precision

$$A(r, q) = \frac{\sum_{i=1}^m v(d_i, q) P(\{d_j\}_{j=1}^i, q)}{\sum_{i=1}^m v(d_i, q)}$$

- Discounted Cumulative Gain

$$DCG(r, q) = \sum_{i=1}^m \frac{v(d_i, q)}{\log(i+1)}$$



Statistical Optimization

Maximize the expected value of $L(r, q)$

$$r^* = \arg \max_r \{E_q(L(r, q))\}$$

Cossock and Zhang (COLT 2006) show that

If $L(r, q)$ is DCG, $E(v(d, q) | d, q)$ is Bayes optimal

Implies a per-item scoring function $s(d, q)$
can be an optimal ranking procedure



Function Estimation

Learn a scoring function $s(d, q)$ by estimating $E(v(d, q) | d, q)$

Typical solution is regression:

$y = v(d, q)$ and x are ranking features

$$s(x) = \arg \min_s \{E((y - s(x))^2 | x)\}$$



Historical Development



Berkeley at Trec3 (1995)

- Cooper, Chen and Gey
- Ranking Features
 - Document dependent (e.g. length)
 - Query dependent (e.g. sum query tf)
 - Query/Document dependent (e.g. sum document tf)
- Logistic regression
 - to determine optimal combination
- Superseded by BM25



Alta Vista (2002)

- Cossock and Travis
- Goal: add new ranking features
 - to existing hand-tuned polynomial
- Method: Boosted Tree Regression
 - Powerful, universal learning procedure
 - Gradient descent optimization of empirical loss
- Problem: Gather adequate training data
 - Optimize use of editorial effort



Novel Features

- Automatically generated ranking function
 - Code generated from estimated model
 - Runtime cost restricted model size
- Factoring of feature development from optimization
 - Enabling parallel feature development teams
- Importance sampling
 - Most documents are irrelevant
 - Emphasize positive example, sample negative example

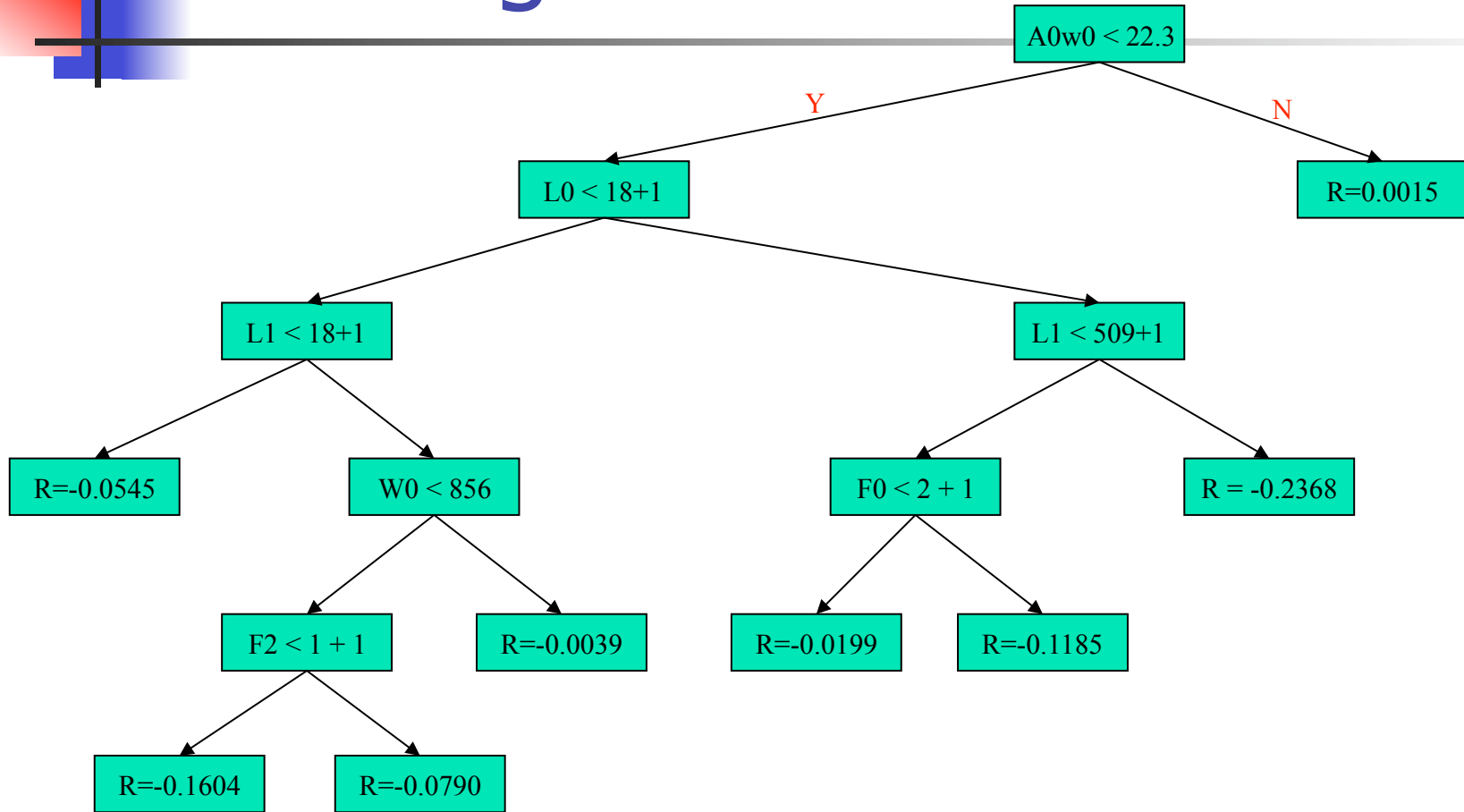


Ranking Features

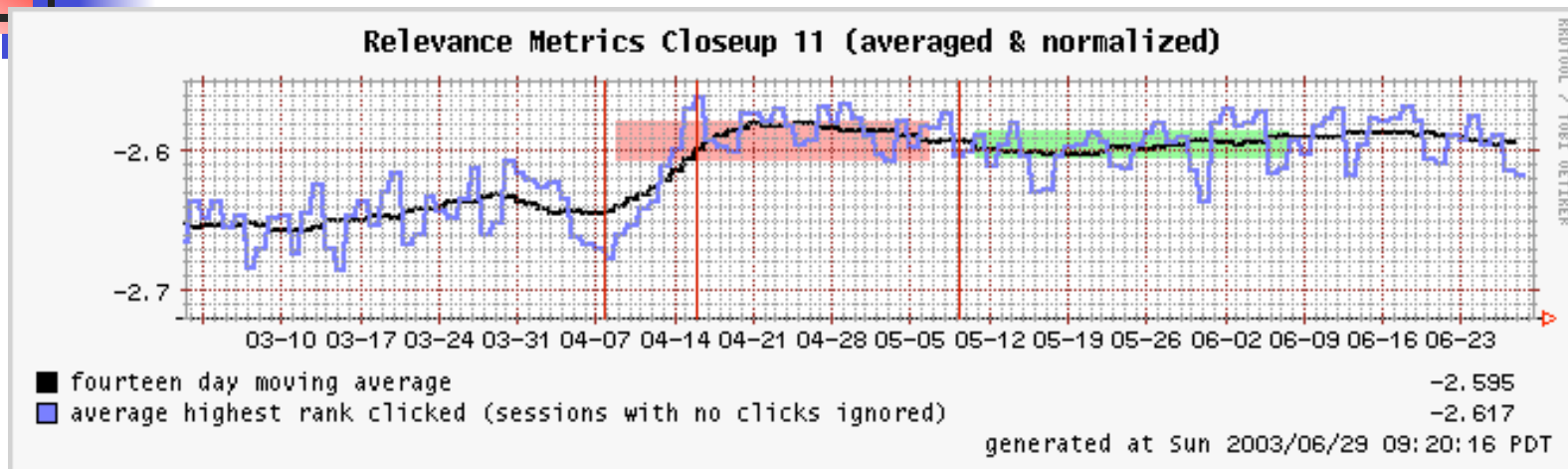
- A0 - A4 anchor text score per term
- W0 - W4 term weights
- L0 - L4 first occurrence location
(encodes hostname and title match)
- SP spam index: logistic regression of 85 spam filter variables
(against relevance scores)
- F0 - F4 term occurrence frequency within document
- DCLN document length (tokens)
- ER Eigenrank
- HB Extra-host unique inlink count
- ERHB $ER * HB$
- A0W0 etc. $A0 * W0$
- QA Site factor –
logistic regression of 5 site link and url count ratios
- SPN Proximity
- FF family friendly rating
- UD url depth



Ranking Tree



Impact of First Release



Average highest rank clicked perceptibly increased with the release of a new rank function.



Yahoo/Inktomi (2004)

- Alta Vista Relevance Team
- Goal: increase improvement rate
 - Existing method supported two releases/year
- Method: Factor feature development from optimization
 - Focus effort on building feature pipeline
 - Deploy features already in use at AltaVista
- Problem: Generalize to 20+ markets
 - Reduce cost of training set development



Novel Features

- Active Learning
 - Iterative learning with re-judging
 - Effectively address ranking errors
- Large Scale training set generation
 - Recycled data from competitive metrics
- Large runtime models
 - Score pruning



Challenges

- Semi-frozen training data
 - Documents captured, but not webmap
 - Webmap assumed to evolve slowly
 - Enables incremental construction
 - Brittle procedure prone to error
 - Feature computation errors frequently delay releases
 - Time sensitive queries not addressable
 - Features must be captured at a moment in time



Challenges (cont'd)

- Market Specialization
 - Each market treated as a separate problem
 - Unique features
 - Dedicated training data
 - Scales poorly to 20+ markets
 - Prioritization to key markets
 - Insufficient resources to cover the tail
 - A Better approach:
 - Treats language and regionality as document features
 - Pool training data across markets



Challenges (cont'd)

- Adaptation to rich query annotation
 - Sparse, high-value features; e.g.
 - Named entity instances
 - Weighted alternatives
 - Specialized use cases; e.g.
 - Query categorization



Intent-based Ranking



Key Concepts

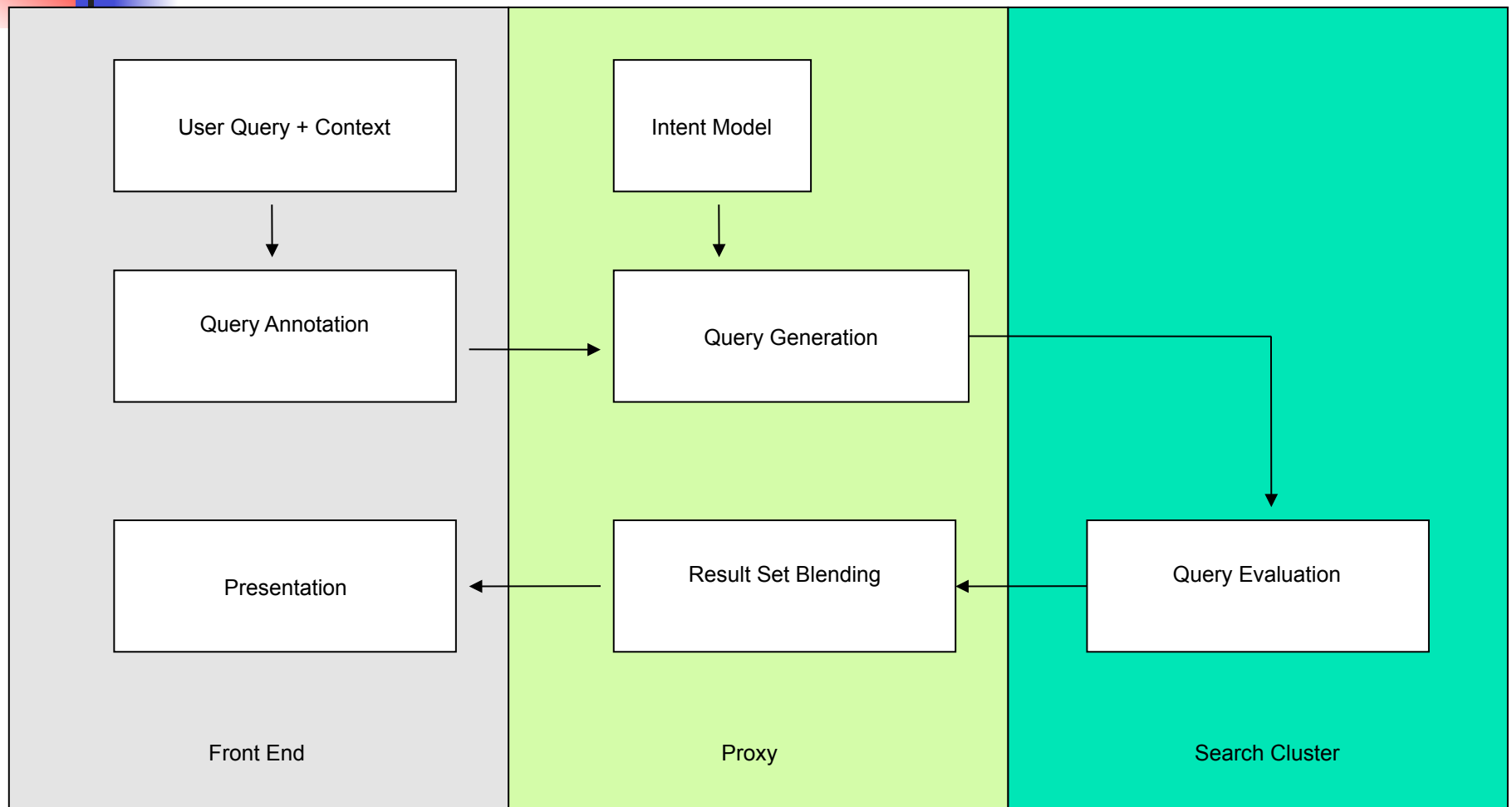
- An intent
 - A fully specified query that accurately captures a user intent
- Query analysis
 - Query annotation via NLP
- Intent modeling
 - Mapping from an annotated query to one or more intents
- Intent Blending
 - Merging of multiple intent-specific result sets into a single result set based on posterior estimates of relevance



Intent-based Ranking

1. Given a set of weighted intents
2. Score candidates for each intent
3. And blend across intents
4. To produce a ranked list of results

Overall Flow





Comparison with MLR

- MLR: average over all intents

- Rank by:

$$E\{R(d) | q\}$$

- Per-item criterion
- Not specialized to query type
- Horizontal features/improvements

- iMLR: blend across intents

- Rank by:

$$\sum E\{R(d) | I\}P\{I | q\}$$

$$P\{I | q\} = P\{I\} \frac{P\{q | I\}}{P\{q\}}$$

- Posterior estimate of Intent
- Specialized to query type
- Category specific features/improvement

- Methodology for query-specific improvement



Summary

- MLR is a key continuous improvement technology
 - In production for many years and Yahoo! (and Microsoft)
- Active area for research
 - Multiple KDD and SIGIR workshops
- Can be extended to intent-based ranking