11-442 / 11-642 / 11-742:
Search Engines

Feature-Based Retrieval

Jamie Callan and Chenyan Xiong
Carnegie Mellon University
callan@cs.cmu.edu

1

Introduction

We have covered many different approaches to retrieval

- Retrieval models: Vector space, BM25, language models, …
- Representations: Title, body, url, inlink, …
- Query templates: Sequential dependency models, …
- Query-independent evidence: PageRank, url depth, …

Different approaches can be combined to improve accuracy

- We have seen combinations that were created manually

Manually exploring the space of combinations is impractical

- Too many combinations and too many parameters
- Use machine learning instead…

2

2

Introduction

A brief introduction to machine learning

- Use training data $X \rightarrow Y$ — X: examples (feature vectors); Y: labels (desired values)

  to learn a model h of how labels are assigned to examples

  $$Y = h(X; \boldsymbol{w}) \quad \boldsymbol{w}: \text{ feature weights}$$

- For a new (unlabeled) example x, predict the label y

  $$y = h(\boldsymbol{x}; \boldsymbol{w})$$

Issues that must be addressed to apply this approach to search

- How are examples encoded as feature vectors?
- What are the examples and labels?

3

3

Introduction

"Learning to rank (LeToR or LTR or L2R)"

- Feature: Evidence about how well query *q* matches document *d*
- Learn a model that combines many types of evidence

Example features

f_1: $BM25_{Title}$ (q, d)
f_2: $BM25_{Body}$ (q, d)
f_3: $Indri_{Title}$ (q, d)
f_4: $Indri_{Body}$ (q, d)
f_5: $Indri\_SDM_{Body}$ (q, d)
f_6: $RankedBoolean_{Body}$ (q, d)
f_7: VSM_{Title} (q, d)
f_8: VSM_{Body} (q, d)
f_9: PageRank (d)
f_{10}: $Length_{URL}$ (d)
f_{11}: $Length_{Terms}$ (q)
: : :

4

4

Introduction

"Learning to rank (LeToR or LTR or L2R)"

- **Feature:** Information about q, d, or the pair (q, d)
  - E.g., $BM25_{Body}$ (q, d) or ContainsCityName (q) or PageRank (d)
- **Feature vector:** A list of features
- **Label:** {Relevant, Not Relevant} or {0, 1, 2, 3, 4} or …
  - Sometimes the label is known (*training*)
  - Sometimes the label is predicted using the model (*testing*)
- **Model:** A method of combining evidence to predict a label

5

5

Introduction

LeToR is a type of supervised learning

Familiar supervised learning tasks

- Classify an example into a discrete category
  - Find h: $X \rightarrow Y; Y \in \{cat, dog, mouse, \ldots\}$
  - E.g. Naïve Bayes, SVM
- Map each example to a numeric score
  - Find h: $X \rightarrow Y; Y \in R$
  - E.g. Linear regression, logistic regression

6

6

Introduction

LeToR is a type of supervised learning

Ranking task

- Produce the best ranking of given documents
- Usually done by finding ranking scores for (q,d)
  - h: $\boldsymbol{x} \rightarrow y; y \in R$
    - » $\boldsymbol{x}$: A feature vector that describes how well q matches d
    - » y: The predicted label (a ranking score)
  - Note: We only care about the ranking order, not the scores
- Predict scores of d_1, d_2, d_3, d_4, … for query q
- Sort documents by their scores to produce a ranking

Introduction

We start with linear models, which are easy

$$Similarity(q,d) = \begin{matrix} w_1 \times f_1(q,d) + \\ w_2 \times f_2(q,d) + \\ \vdots \\ w_n \times f_n(q,d) + \end{matrix} = \sum w_i\, f_i(q,d) = h(\boldsymbol{x}, \boldsymbol{w})$$

f_i: A feature measuring how well q matches d (e.g., $BM25_{Body}$)

w_i: A weight indicating the importance of feature f_i

h: The model

x: A feature vector (i.e., f_1, f_2, … f_n)

w A weight vector (i.e., w_1, w_2, … w_n)

Introduction

If you care about <u>efficiency</u>, you may be worrying about costs

- $Similarity(q, d) = \sum w_i \times f_i(q, d)$
  - Each f_i is a function that may be expensive
    - » E.g., BM25, Indri, SDM, …
  - There may be many features
    - » E.g., 20-100

Won't this be incredibly slow?

Introduction

Where does learning to rank fit in a large search engine?

First, query classification to determine the type of query

- Make decisions about how the query will be evaluated

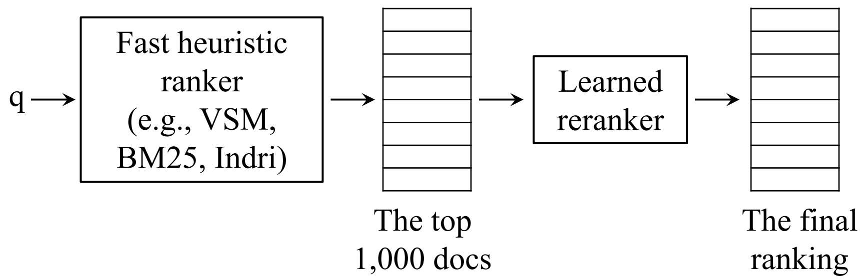
Then retrieval is done by a <u>sequence of retrieval models</u>

One example (there are many variations)

- Exact-match Boolean: <u>Form a set</u> of documents
- Best-match retrieval: <u>Rank</u> the set, select the top n_1
  - E.g., Indri, BM25
- Learned models: <u>Re-rank</u> the top n_1, select the top n_2
  - **Note:** LeToR does not <u>find</u> documents, it just <u>scores</u> them

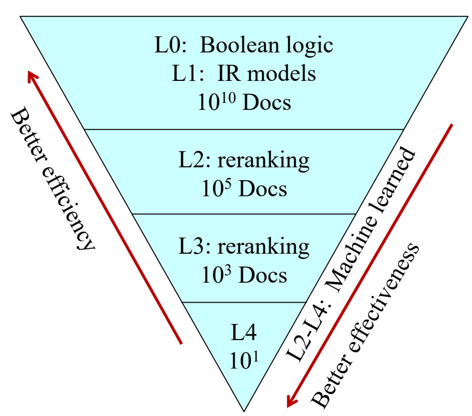
Introduction

A standard reranking architecture



This pipeline can have an arbitrary number of rerankers

Introduction



Each layer ranks, prunes, and returns results

Layered evaluation gives control over search costs

- Simpler models are applied to massive data
  - Efficient
- Sophisticated models are applied to little data
  - Effective

(Pederson, 2010)

Introduction

LeToR methods are described along three main dimensions

- The document representation (the features)
- The type (or style) of training data
- The machine learning algorithm

Our focus is the first two (this is not a machine learning class) … but we will discuss some example learning algorithms, too

- 11-641, *Machine Learning for Text Mining* provides more in-depth coverage of the learning algorithms

13

Introduction

For a query q, model document d as $\vec{x} = \Phi(d, q)$

- Φ extracts features for (d, q). Think of it as a feature factory.
- It produces a feature vector $\vec{x}$ or $\mathbf{x}$

Use training data to learn a model $h(\vec{x})$ **or** $h(\mathbf{x})$

- $h(\vec{x})$ generates a real-valued score that is used to rank d for q

Note: This notation makes q and d implicit in the feature values

- It is consistent with a typical machine learning presentation
- Think only of feature vectors $\vec{x}$ and learned models $h(\vec{x})$
  - Each feature vector $\vec{x}$ describes a (d, q) pair

14

Introduction: Example Features

An initial set of features

• $x_{\text{CoordinationMatch}}$	Number of query terms that match document d
• x_{VSM}	Vector space score for query q and document d (or a field in document d)
• x_{BM25}	BM25 score for query q and document d (BM25 features have been in Bing since ~2006)
• x_{Indri}	Indri score for query q and document d
• x_{PageRank}	PageRank score for document d
• x_{Spam}	Spam score for document d
• x_{UrlDepth}	Depth of document d in a website
• …	

15

15

Introduction: Example Features

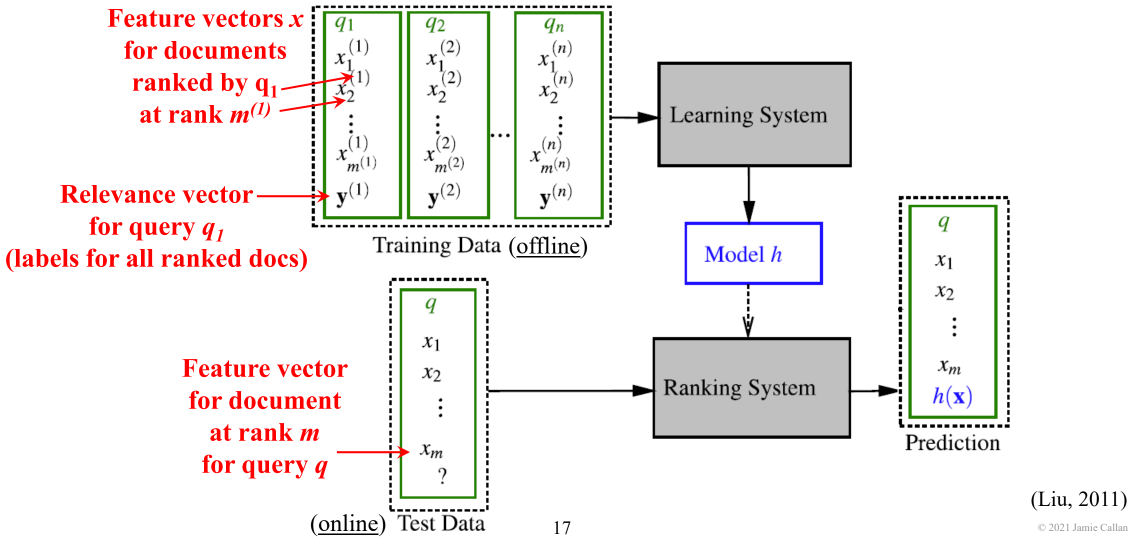
More features…

- Content length
- Query term density
- Number of inlinks
- Number of referring domains
- Number of referring IPs
- Time on site
- Pages per session
- Bounce rate
- Website security (https)

16

16

Introduction: Learning to Rank Framework



17

17

Introduction

Machine learning algorithms are designed to <u>predict labels</u>

What are the labels for (q, d)?

• Relevant and not relevant?	Easy to get, hard prediction problem
• Relevance grades (e.g., 0, 1, 2, 3, 4)?	Easy to get, hard prediction problem
• A score in the range [0.0 … 1.0]?	Hard to get, easier prediction problem
• Position in an ideal ranking?	Hard to get, easier prediction problem
• …?	

Methods differ in how they address this issue

18

18

Introduction

Today we cover three approaches to learning to rank

How are they similar?

- Each uses a trained model h to estimate the score of $\mathbf{x}$
  - $\mathbf{x}$ describes how well document d matches query q
- Documents are ranked by the score $h(\mathbf{x})$

How are they different?

- Three different approaches to training the model h
  - Different types of training data

Outline

Introduction to feature-based methods

Three approaches to training learning algorithms

- Pointwise
- Pairwise
- Listwise

Benchmark datasets

Sample results

Pointwise Approaches

Approach: Train a model using individual documents

- Training data: $\mathbf{x} \rightarrow$ label (q, d, relevance)
- Learned model: h ($\mathbf{x}$) $\rightarrow$ label (q, d, ?)
- (Remember, $\mathbf{x} = \Phi(d_i, q)$)

The type of label depends on the learning algorithm

1. Classification (e.g., SVM): Predict a relevance category
   – h ($\mathbf{x}$) = relevant
2. Regression (e.g., linear regression): Predict a relevance score
   – h ($\mathbf{x}$) = 0.53

21

21

Pointwise Approaches

LeToR Procedure

- **Offline:** Use Cranfield evaluation data to train a model
  – Queries, documents, relevance assessments
- **Online:** Given a new query q
  – Run a ranker (e.g., BM25) to get initial documents
    » E.g., the top 100 returned by BM25
  – For each initial document
    » Generate a feature vector $\mathbf{x_i} = \Phi(d_i, q)$
    » Use the model to generate a reranking label h($\mathbf{x_i}$) for d_i
  – Rerank the initial documents

22

22

Pointwise Approaches: Classification

Classification (e.g., SVM): **Predict a relevance category**

$x_1 = \Phi(d_1, q)$, $y_1 = \text{NRel}$ (not relevant) doc$_1$

$x_2 = \Phi(d_2, q)$, $y_2 = \text{HRel}$ (highly relevant) doc$_2$

Loss function: Typically, 0 if the label is correct, otherwise 1

- The goal is to minimize the number of misclassified examples

Training data is (relatively) easy to get

- Use the Cranfield methodology
- Hire relevance assessors to assign documents to categories
  - E.g., { not relevant, relevant, highly relevant }

Pointwise Approaches: Problems with Classification

Categories are ordinal (ordered), which most algorithms ignore

- Some errors are more serious than others
- Consider two trained models…

C_1	Label	Predicted	
d_1	HRel	Rel	×
d_2	Rel	HRel	×
d_3	NRel	NRel	✓

Loss: 2

C_2	Label	Predicted	
d_1	HRel	NRel	×
d_2	Rel	Rel	✓
d_3	NRel	NRel	✓

Loss: 1

  – C_1 is the better ranker, but C_2 has lower loss

- What is the relative importance of different types of errors?
  – There isn't good theory to guide us

Pointwise Approaches: Regression

Regression (e.g., linear regression): Predict a relevance score

$x_1 = \Phi(d_1, q), \quad y_1 = -1 \qquad \text{doc}_1$

$x_2 = \Phi(d_2, q), \quad y_2 = 3 \qquad \text{doc}_2$

Loss function: Squared error

- The goal is to avoid large errors

Training data is difficult to get

- What is the desired score of d_{21} for q?
  (see next slide)

25 © 2021 Jamie Callan

25

Pointwise Approaches: Problems with Regression

Predicting numeric labels is difficult because numeric labels are arbitrary

- During training, different mappings from relevance categories to numeric values cause the learning algorithm to learn different models
  - { NRel, Rel, HRel } → { 0, 1, 2 } Equal differences among categories
  - { NRel, Rel, HRel } → { 1, 4, 9 } Prefer HRel correct
  - { NRel, Rel, HRel } → { 1, 16, 81 } Must get HRel correct
- What is the relative importance of different types of errors?
  - What are the right numbers to assign for Nrel, Rel, and HRel?
  - There isn't good theory to guide us
  - Thus, pointwise training data is inherently noisy

26 © 2021 Jamie Callan

26

Pointwise Approaches: Problems

Pointwise methods focus on making good predictions for individual documents

- E.g., minimize the number of misclassified examples
- E.g., minimize the sizes of classification errors

Search metrics care about the order in a set of documents

- E.g., NDCG@k, MAP@k

Pointwise methods tend to be less accurate than other methods

- Noise is inherent in the training data
- The loss functions don't model the task well
- The model focuses on the wrong goal (predict scores, not ranks)

27

27

Outline

Introduction to feature-based methods

Three approaches to training learning algorithms

- Pointwise
- Pairwise
- Listwise

Benchmark datasets

Sample results

28

28

Pairwise Approaches

Approach: Train a model using pairs of documents

- Training data: prefer ($\mathbf{x_1}$, $\mathbf{x_2}$)
- Learned model: h ($\mathbf{x_1}$) > h ($\mathbf{x_2}$)

What is the pair value?

- Binary assessments $\{\mathbf{x_1} > \mathbf{x_2}, \mathbf{x_1} < \mathbf{x_2}\}$

Loss function

- 0 if a pair is ordered correctly, otherwise 1

Minimize the number of misclassified document pairs

- Focus on preference, not specific scores or labels

29

29

Pairwise Approaches

How is training data produced?

Relevance Data	**Training Data$_q$**
d_1, relevant	$\mathbf{x_1} = \Phi(d_1, q) > \mathbf{x_2} = \Phi(d_2, q)$
d_2, not relevant	$\mathbf{x_2} < \mathbf{x_1}$
d_3, not relevant	$\mathbf{x_1} > \mathbf{x_3}$
d_4, relevant	$\mathbf{x_3} < \mathbf{x_1}$
d_5, not relevant	: :
d_6, relevant	$\mathbf{x_4} > \mathbf{x_2}$
d_7, not relevant	$\mathbf{x_2} < \mathbf{x_4}$
: : :	: :

Each preference is one example (one instance of training data)

30

30

Pairwise Approaches

How is preference data used to train a linear classifier?

- **Training data:** prefer (d_1, d_2)
- Extract feature vectors: $\overrightarrow{x_1} = \Phi(d_1, q)$ $\overrightarrow{x_2} = \Phi(d_2, q)$
- Assume a linear model: $h(\overrightarrow{x_i}) = w^T \overrightarrow{x_i}$

prefer (d_i,d_j) $\rightarrow h(\overrightarrow{x_i}) > h(\overrightarrow{x_j})$

$\rightarrow w^T \overrightarrow{x_i} > w^T \overrightarrow{x_j}$

$\rightarrow w^T \overrightarrow{x_i} - w^T \overrightarrow{x_j} > 0$

$\rightarrow w^T \left(\overrightarrow{x_i} - \overrightarrow{x_j}\right) > 0$

$\rightarrow h\left(\overrightarrow{x_i} - \overrightarrow{x_j}\right) > 0$

	x_i	x_j	x_i-x_j
f_1	0.67	0.59	0.08
f_2	0.01	0.02	-0.01
f_3	0.23	0.19	0.04
f_4	0.15	0.13	0.02
f_5	0.80	0.82	-0.02
	: :	: :	: :

feature vectors

- Now it is a standard two-class learning problem: $h\left(\overrightarrow{x_j} - \overrightarrow{x_i}\right) < 0$ $h\left(\overrightarrow{x_i} - \overrightarrow{x_j}\right) > 0$
  - Use your favorite algorithm (e.g., regression, SVM)

31

31

Example: Ranking SVM

Goal: Minimize the number of misclassified document pairs

minimize: $\frac{1}{2} \vec{w} \cdot \vec{w}$ $+$ $C \sum \xi_{i,j,k}$

complexity **accuracy (on training data)**

subject to:

$\forall i \, \forall j \, \forall k: \; \xi_{i,j,k} \geq 0$

$\forall \left(d_i, d_j\right) \in r_1 : \vec{w} \cdot \overrightarrow{x_i} > \vec{w} \cdot \overrightarrow{x_j} + 1 - \xi_{i,j,k}$

…

$\forall \left(d_i, d_j\right) \in r_n : \vec{w} \cdot \overrightarrow{x_i} > \vec{w} \cdot \overrightarrow{x_j} + 1 - \xi_{i,j,k}$

Preference pairs as constraints

C: Controls model complexity (small C) and training error (large C)
ξ: Slack variable
r_n: k training pairs from the n^{th} ranking (query)

(Croft, et al., 2010)

32

32

Example: Ranking SVM

Properties of Ranking SVM

- Generalizes well
- Kernels can be used to improve accuracy
  - But linear kernels often work well
- Inherits desirable properties of SVM
  - Many open source tools
  - Much research effort on optimization → fast training
  - Theoretical guarantees (in learning)

33

33

Pairwise Approaches

LeToR Procedure

- **Offline:** Use Cranfield evaluation data to train a model
  - Queries, documents, relevance assessments
- **Online:** Given a new query q
  - Run a ranker (e.g., BM25) to get initial documents
  - For each initial document
    - » Generate a feature vector $\mathbf{x_i} = \Phi(d_i, q)$
    - » Use the model to generate a reranking label $h(\mathbf{x_i})$
  - Rerank the initial documents

34

34

Pairwise Approaches

There have been many pairwise approaches

- RankNet (Burges, et al., 2005)
- Frank (Tsai, et al., 2007)
- RankBoost (Freund, et al., 2003)
- Ranking SVM (Herbrich, et al., 2000; Joachims, 2002)
- MHR
- IR-SVM
- …

(Liu, 2011)

35

35

Problems with Pairwise Approaches

Queries with an <u>even balance</u> of relevant and non-relevant documents dominate the training data

- Number of pairs = |R| × |NR|
  - q_1: 5 relevant, 5 not relevant: 5 R × 5 NR = 25 pairs
  - q_2: 2 relevant, 8 not relevant: 2 R × 8 NR = 16 pairs
  - q_3: 9 relevant, 1 not relevant: 9 R × 1 NR = 9 pairs

The pairwise approach is more sensitive to noisy labels

- One noisy label generates many training instances

Usually the ranking position is ignored

- As with the pointwise approaches

36

36

Outline

Introduction to feature-based methods

Three approaches to training learning algorithms

- Pointwise
- Pairwise
- Listwise

Benchmark datasets

Sample results

Listwise Approaches

Approach: Train a model using a list of documents

- Training data: $\mathbf{x}_1 > \mathbf{x}_2 > \ldots > \mathbf{x}_n$
- Learned model: $h(\mathbf{x}_1) > h(\mathbf{x}_2) > \ldots$

What is the value of a particular ranking?

- Some metric over the ranking
  - E.g., NDCG@n, with n=1, 3, 5, 10, …
  - E.g., MAP@n

The goal is to maximize the value of the metric

- Directly align the model with the ranking target

Listwise Approaches

How is training data produced?

Binary Relevance	Multi-Valued Relevance
d_1, relevant	d_4, perfect
d_4, relevant	d_1, excellent
d_6, relevant	d_6, very good
d_2, not relevant	d_2, poor
d_3, not relevant	d_3, poor
d_5, not relevant	d_5, poor
d_7, not relevant	d_7, poor
: : :	: : :

Order documents by their relevance scores (best to worst)

The training algorithm generates and scores rankings

39

39

Listwise Approaches: Challenges

Directly optimizing some metrics is hard

- Popular metrics (e.g., NDCG@n) are not continuous, nor convex
  - Very hard for optimization algorithms

Two common strategies in listwise approaches:

1. Find another metric that is intuitive and easy to optimize.
   - E.g. likelihood of 'best' rankings in training data
2. Directly optimize ranking evaluation metrics, with approximation

40

40

Listwise Approach: ListMLE
***List*wise *M*aximum *L*ikelihood *E*stimation**

General Idea

- Construct the probability of a ranking p($\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$)
- Find the model $h(x)$ that maximizes the probability (likelihood) of best rankings in the training data
- Use $h(x)$ in ranking

The key step is constructing p($\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$)

- It is impossible to define $p(x_1, x_2, \ldots, x_n)$ directly
  - The sample space is all possible ranks: $n!$
- ListMLE defines a generative process with much smaller space
  - With the help of independence assumptions

(Xia, et al., 2008)

41

41

Listwise Approach: ListMLE
***List*wise *M*aximum *L*ikelihood *E*stimation**

ListMLE's <u>generative definition</u> of the ranking process

- The ranking is assembled iteratively from candidate documents
- Given a set of unranked candidate documents $S_i = \{d_1, \ldots, d_n\}$
  - Pick the best candidate d_i from S_i to appear at rank i

$$p(d_i | S_i; w) = \frac{\exp(w^T x_i)}{\sum_{x_j \in S_i} \exp(w^T x_j)}$$

  - Remove d_i from the set of unranked documents: $S_{i+1} = S_i \setminus d_i$
- Repeat until all candidate documents are ranked
- **Assumption:** $p(d_i|S_i;w)$ is independent of other documents

Thus, the learning task is to learn $\exp(w^T x)$

(Xia, et al., 2008)

42

42

Listwise Approaches: ListMLE

The likelihood of ranking r_k

$$\mathrm{l}(r_k; w) = p(x_1, x_2, \dots, x_n; w) = \prod_{i=1}^{n} p(d_i | S_i; w)$$

Use Maximum Likelihood Estimation to find parameters

$$w^* = \operatorname{argmax}_w \mathrm{l}(r_k; w)$$

- Solved by typical gradient methods

For new query, documents are ranked by:

$$h(x) = \exp(w^T x)$$

Note: Different learning process, but familiar type of learned model

(Xia, et al., 2008)

Listwise Approaches: ListMLE

LeToR Procedure

- **Offline:** Use Cranfield evaluation data to train a model
  - Queries, documents, relevance assessments
- **Online:** Given a new query q
  - Run a ranker (e.g., BM25) to get initial documents
  - For each initial document
    - » Generate a feature vector $\mathbf{x_i} = \Phi(\mathrm{d}_i, q)$
    - » Use the model to generate a reranking label $\mathrm{h}(\mathbf{x_i})$
  - Rerank the initial documents

Three Approaches to Training: Summary

Pointwise

- Training data is a document category or score
- Accurate category or score ≠ accurate ranking
- Position information ignored

Pairwise

- Training data is a preference among a pair of documents
- Accurate preference ≠ accurate ranking
- Position information ignored

Listwise

- Training data is a ranking of documents
- Hard to directly optimize ranking metrics

Three Approaches to Training: Summary

Pointwise is the weakest of the three approaches

Pairwise and listwise are about equally useful

- Pairwise has an imperfect learning target, but is easy to achieve
  - Minimizes pairwise errors, but we want the best ranking
  - A simpler learning problem with theoretical guarantees
- Listwise has a perfect learning target, but is harder to achieve
  - Learning target is exactly the same as what we want
  - A harder learning problem

Which do you prefer?

Outline

Introduction to feature-based methods

Three approaches to training learning algorithms

- Pointwise
- Pairwise
- Listwise

Benchmark datasets

Sample results

Benchmark Datasets

Three popular LeToR benchmark datasets

1. The LeToR collection (Microsoft) (2007)
2. Yahoo! Challenge datasets (2010)
3. Microsoft Learning to Rank datasets (2010)

Benchmark Dataset #1: The LeToR Collections

Created by Microsoft using existing publicly-available data

- **The .gov corpus**
  - 1M .gov web documents from 2002
  - 350 queries (topic, homepage, named page)
  - Top 1000 documents per query returned by BM25
  - 64 features
- **The OHSUMED corpus**
  - 350K PubMed abstracts from 1987-1991
  - 106 queries (informational)
  - All judged documents
  - 40 features

(http://research.microsoft.com/en-us/projects/mslr/feature.aspx)

(Qin, et al., 2010)

49

49

Benchmark Dataset #1: The LeToR Collections

Field-specific features (title, anchor, url, whole) for (*d*, *q*)

- Tf: Sum over all query terms: $\sum_{q_i \in q} tf_{q_i}$
- Idf: Sum over all query terms: $\sum_{q_i \in q} idf_{q_i}$
- Tf × Idf: Sum over all query terms: $\sum_{q_i \in q} tf_{q_i} \times idf_{q_i}$
- Field length
- Retrieval model scores
  - Boolean, VSM, BM25, LM_{Abs}, $LM_{Dirichlet}$, LM_{JM}
- Hyperlink-based features, HITS, Topic-specific PageRank, …

(Qin, et al., 2010)

50

50

Benchmark Dataset #1: The LeToR Collection

Document-level features for (*d*, *q*)

- PageRank, number of inlinks
- Url: Number of '/', length
- Number of child pages

Note that *q* is irrelevant to these features

- These features prefer certain types of pages

(Qin, et al., 2010)

51

51

Benchmark Dataset #2: The Yahoo Challenge! Datasets

Created by Yahoo using proprietary data

- **Set 1**
  - 710K feature vectors
  - 30K queries
  - 519 features (not described)
  - Relevance scale with 5 values
- **Set 2**
  - 173K feature vectors
  - 6K queries
  - 596 features (not described)
  - Relevance scale with 5 values

(Liu, 2011)

52

52

Benchmark Dataset #3: Microsoft Learning to Rank Dataset

Created by Microsoft using proprietary data

- 3.7M web documents
- 30K queries
- 136 features
- Relevance scale with 5 values

Example data

0	qid:1	1:3 2:0 3:2 4:2 ... 135:0 136:0
2	qid:1	1:3 2:3 3:0 4:0 ... 135:0 136:0

relevance   query id   features

(http://research.microsoft.com/en-us/projects/mslr/)

Benchmark Dataset #3: Microsoft Learning to Rank Dataset

Field-specific features (title, anchor, body, url, whole) for (*d*, *q*)

- Covered query term number, covered query term ratio
  - Terms in query q that appear in d
- Field length, field length normalized in various ways
- Idf
- Tf: Min, Max, Sum, Mean, Variance
  - "apple pie recipe": Min (tf_{apple} tf_{pie} tf_{recipe})
- Tf × idf: Min, Max, Sum, Mean, Variance
- Retrieval model scores: Boolean, VSM, BM25, $LM_{Dirichlet}$, LM_{JM}

(http://research.microsoft.com/en-us/projects/mslr/feature.aspx)

Benchmark Dataset #3: Microsoft Learning to Rank Dataset

Document-level features for (*d*, *q*)

- Url: Number of '/', length
- Number of inlinks and outlinks
- PageRank, SiteRank, url click count, url dwell time
- Two quality scores
- Query-url click count

Note that *q* is irrelevant to these features

- These features prefer certain types of pages

(http://research.microsoft.com/en-us/projects/mslr/feature.aspx)

55

55

Outline

Introduction to feature-based methods

Three approaches to training learning algorithms

- Pointwise
- Pairwise
- Listwise

Benchmark datasets

Sample results

56

56

Sample Experimental Results

LeToR Benchmark (.gov dataset, topic distillation (TD) queries)

	Algorithm	NDCG@ 1	NDCG@ 3	NDCG@ 10	P@ 1	P@ 3	P@ 10	MAP
Point	Regression	0.320	0.307	0.326	0.320	0.260	0.178	0.241
Pair	RankSVM	0.320	**0.344**	0.346	0.320	**0.293**	0.188	0.263
	RankBoost	0.280	0.325	0.312	0.280	0.280	0.170	0.227
	FRank	0.300	0.267	0.269	0.300	0.233	0.152	0.203
List	ListNet	**0.400**	0.337	**0.348**	**0.400**	**0.293**	**0.200**	**0.275**
	AdaRank	0.260	0.307	0.306	0.260	0.260	0.158	0.228
	SVM^{Map}	0.320	0.320	0.328	0.320	0.253	0.170	0.245

RankSVM is similar to ListNet except at rank 1

(Liu, 2011)

57

57

Sample Experimental Results

LeToR Benchmark (.gov dataset, named page (NP) queries)

	Algorithm	NDCG@ 1	NDCG@ 3	NDCG@ 10	P@ 1	P@ 3	P@ 10	MAP
Point	Regression	0.447	0.614	0.665	0.447	0.220	0.081	0.564
Pair	RankSVM	0.580	0.765	0.800	0.580	**0.271**	0.092	0.696
	RankBoost	**0.600**	0.764	**0.807**	**0.600**	0.269	**0.094**	**0.707**
	FRank	0.540	0.726	0.776	0.540	0.253	0.090	0.664
List	ListNet	0.567	0.758	0.801	0.567	0.267	0.092	0.690
	AdaRank	0.580	0.729	0.764	0.580	0.251	0.086	0.678
	SVM^{Map}	0.560	**0.767**	0.798	0.560	0.269	0.089	0.687

RankSVM, and RankBoost are best

(Liu, 2011)

58

58

Lessons Learned

Observations about the effectiveness of different algorithms

- Many learning algorithms perform relatively well
- Relative effectiveness: Listwise ≈ Pairwise > Pointwise
  - As expected

Many ML algorithms work with pointwise & pairwise LeToR

- Easy to develop, reasonably effective

Listwise algorithms may be more effective

- But, there are fewer off-the-shelf solutions
- Still an open research topic

59

59

Lessons Learned

Much of the LeToR literature uses lots of training data

- Research is driven by web companies that have a lot of data
- But … <u>you</u> may not have a lot of data
  - Their conclusions may not apply to your situation

60

60

Lessons Learned

Observation from academic research

- 100-200 labeled queries can support 50-100 features
  - Surprising compared to other classification/regression tasks, which need a higher ratio of examples to features
- The theory behind LeToR is still an open research topic

Use large numbers of features cautiously

- Start with a small set of high-quality features, then grow it
- Design features carefully

Lessons Learned

Much research is driven by machine learning researchers …their focus is on machine learning algorithms

The features used in most systems are surprisingly simple

- Simple statistics (e.g., tf, idf, tf × idf, field length)
- Obvious variations of existing ranking algorithms
- A few page quality metrics

Better understanding of search can produce better features … and better search accuracy

- A nice opportunity for future improvement

(Liu, 2011)

Outline

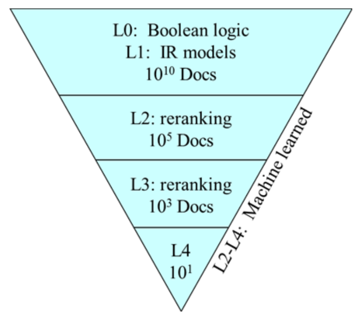
Introduction to feature-based methods

Three approaches to training learning algorithms

- Pointwise
- Pairwise
- Listwise

Benchmark datasets

Sample results



63

63

For More Information

- B. Croft, D. Metzler, and T. Strohman. *Search Engines: Information Retrieval in Practice*. Addison Wesdley. 2010.
- T.-Y. Liu. *Learning to Rank for Information Retrieval*. Springer. 2011.
- T. Qin, T.-Y. Liu, J. Xu, and H. Li. LETOR: A Benchmark Collection for Research on Learning to Rank for Information Retrieval. *Information Retrieval Journal*. 2010.
- F. Xia, T.-Y. Liu, J. Wang, W. Zhang, and H. Li. Listwise approach to learning to rank – Theory and algorithm. In *Proceedings of the 25th International Conference on Machine Learning*. 2008.

64

64