Outline

Outline

Learning to rank (L2R)

Definition
"... the task to automatically construct a ranking model using training data, such that the model can sort new objects according to their degrees of relevance, preference, or importance." - Liu [2009]

L2R models represent a rankable item—e.g., a document—given some context—e.g., a user-issued query—as a numerical vector $\vec{x} \in \mathbb{R}^n$.

The ranking model $f : \vec{x} \to \mathbb{R}$ is trained to map the vector to a real-valued score such that relevant items are scored higher.

We discuss supervised (offline) L2R models first, but briefly introduce online L2R later.

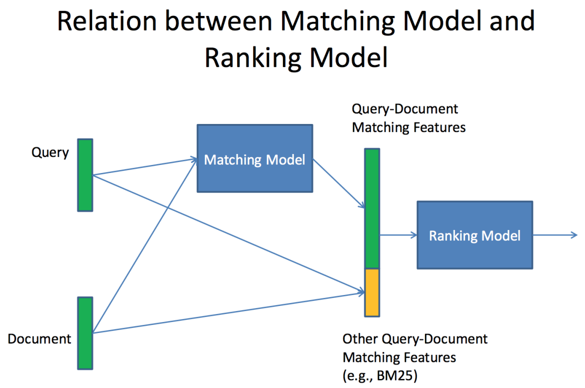
A problem of historical terminology

With the increasing interest within semantic matching models (LTM), the term learning to rank has become ambiguous.

Training data and objectives can be used to optimize models that specifically focus on text matching (see previous section).

In this tutorial, we use learning to rank to refer to signal-agnostic models. That is, models that learn to generate rankings from arbitrary matching, importance or recency signals, amongst others.

A problem of historical terminology


Relation between Matching Model and Ranking Model
Query
Document
Matching Model
Query-Document Matching Features
Ranking Model
Other Query-Document Matching Features (e.g., BM25)

Semantic matching signals as input to a general-purpose ranker. Taken from [Li and Lu, 2016].

How long will this hierarchical view remain valid?

Three training objectives

Liu [2009] categorizes different L2R approaches based on training objectives:

- Pointwise approach: relevance label $y_{q,d}$ is a number—derived from binary or graded human judgments or implicit user feedback (e.g., CTR). Typically, a regression or classification model is trained to predict $y_{q,d}$ given $\vec{x}_{q,d}$.
- Pairwise approach: pairwise preference between documents for a query ($d_i \underset{q}{\succ} d_j$) as label. Reduces to binary classification to predict more relevant document.
- Listwise approach: directly optimize for rank-based metric, such as NDCG—difficult because these metrics are often not differentiable w.r.t. model parameters.

Features

Traditional L2R models employ hand-crafted features that encode IR insights

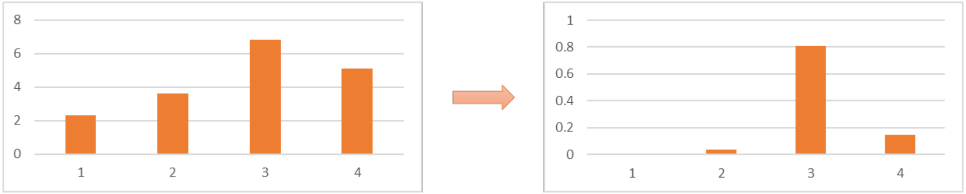
They can often be categorized as:

- Query-independent or static features (e.g., incoming link count and document length)
- Query-dependent or dynamic features (e.g., BM25)
- Query-level features (e.g., query length)

A quick refresher - What is the Softmax function?

In neural classification models, the softmax function is popularly used to normalize the neural network output scores across all the classes

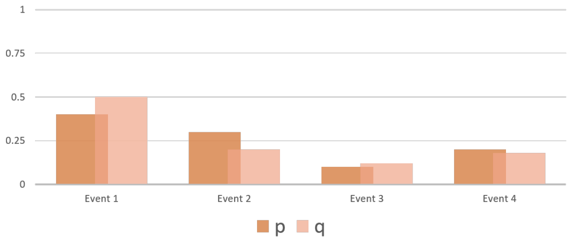
$$p(z_i) = \frac{e^{\gamma z_i}}{\sum_{z \in Z} e^{\gamma z}} \qquad (\gamma \text{ is a constant}) \tag{1}$$



A quick refresher - What is Cross Entropy?

The cross entropy between two probability distributions p and q over a discrete set of events is given by,

$$CE(p,q) = -\sum_i p_i \log(q_i) \tag{2}$$



If $p_{correct} = 1$ and $p_i = 0$ for all other values of i then,

$$CE(p,q) = -\log(q_{correct}) \tag{3}$$

Outline

Pointwise objectives

Regression-based or classification-based approaches are popular

Regression loss

Given $\langle q, d \rangle$ predict the value of $y_{q,d}$

E.g., square loss for binary or categorical labels,

$$\mathcal{L}_{Squared} = \|y_{q,d} - f(\vec{x}_{q,d})\|^2 \tag{4}$$

where, $y_{q,d}$ is the one-hot representation [Fuhr, 1989] or the actual value [Cossock and Zhang, 2006] of the label

Pointwise objectives

Regression-based or classification-based approaches are popular

Classification loss

Given $\langle q, d \rangle$ predict the class $y_{q,d}$

E.g., Cross-Entropy with Softmax over categorical labels Y [Li et al., 2008],

$$\mathcal{L}_{\mathsf{CE}}(q, d, y_{q,d}) = -\log\Big(p(y_{q,d}|q,d)\Big) = -\log\left(\frac{e^{\gamma \cdot s_{y_{q,d}}}}{\sum_{y \in Y} e^{\gamma \cdot s_y}}\right) \tag{5}$$

where, $s_{y_{q,d}}$ is the model's score for label $y_{q,d}$

Outline

Pairwise objectives

Pairwise loss minimizes the average number of inversions in ranking—i.e., $d_i \succ_q d_j$ but d_j is ranked higher than d_i

Given $\langle q, d_i, d_j \rangle$, predict the more relevant document

For $\langle q, d_i \rangle$ and $\langle q, d_j \rangle$,

Feature vectors: $\vec{x}_i$ and $\vec{x}_j$

Model scores: $s_i = f(\vec{x}_i)$ and $s_j = f(\vec{x}_j)$

Pairwise loss generally has the followingform [Chen et al., 2009],

$$\mathcal{L}_{pairwise} = \phi(s_i - s_j) \tag{6}$$

where, ϕ can be,

- Hinge function $\phi(z) = \max(0, 1 - z)$ [Herbrich et al., 2000]
- Exponential function $\phi(z) = e^{-z}$ [Freund et al., 2003]
- Logistic function $\phi(z) = \log(1 + e^{-z})$ [Burges et al., 2005]
- etc.

RankNet

RankNet [Burges et al., 2005] is a pairwise loss function—popular choice for training neural L2R models and also an industry favourite [Burges, 2015]

Predicted probabilities: $p_{ij} = p(s_i > s_j) \equiv \frac{e^{\gamma \cdot s_i}}{e^{\gamma \cdot s_i} + e^{\gamma \cdot s_j}} = \frac{1}{1+e^{-\gamma(s_i - s_j)}}$

and $p_{ji} \equiv \frac{1}{1+e^{-\gamma(s_j - s_i)}}$

Desired probabilities: $\bar{p}_{ij} = 1$ and $\bar{p}_{ji} = 0$

Computing cross-entropy between $\bar{p}$ and p,

$$\begin{aligned}
\mathcal{L}_{RankNet} &= -\bar{p}_{ij} \log(p_{ij}) - \bar{p}_{ji} \log(p_{ji}) && (7) \\
&= -\log(p_{ij}) && (8) \\
&= \log(1 + e^{-\gamma(s_i - s_j)}) && (9)
\end{aligned}$$

Cross Entropy (CE) with Softmax over documents

An alternative loss function assumes a single relevant document d^+ and compares it against the full collection D

Probability of retrieving d^+ for q is given by the softmax function,

$$p(d^+|q) = \frac{e^{\gamma \cdot s\left(q,d^+\right)}}{\sum_{d \in D} e^{\gamma \cdot s(q,d)}} \tag{10}$$

The cross entropy loss is then given by,

$$\mathcal{L}_{\mathsf{CE}}(q, d^+, D) = -\log\left(p(d^+|q)\right) \tag{11}$$

$$= -\log\left(\frac{e^{\gamma \cdot s\left(q,d^+\right)}}{\sum_{d \in D} e^{\gamma \cdot s(q,d)}}\right) \tag{12}$$

Notes on Cross Entropy (CE) loss

- If we consider only a pair of relevant and non-relevant documents in the denominator, CE reduces to RankNet
- Computing the denominator is prohibitively expensive—L2R models typically consider few negative candidates [Huang et al., 2013, Mitra et al., 2017, Shen et al., 2014]
- Large body of work in NLP to deal with similar issue that may be relevant to future L2R models
    - E.g., hierarchical softmax [Goodman, 2001, Mnih and Hinton, 2009, Morin and Bengio, 2005], Importance sampling [Bengio and Senécal, 2008, Bengio et al., 2003, Jean et al., 2014, Jozefowicz et al., 2016], Noise Contrastive Estimation [Gutmann and Hyvärinen, 2010, Mnih and Teh, 2012, Vaswani et al., 2013], Negative sampling [Mikolov et al., 2013], and BlackOut [Ji et al., 2015]

Outline

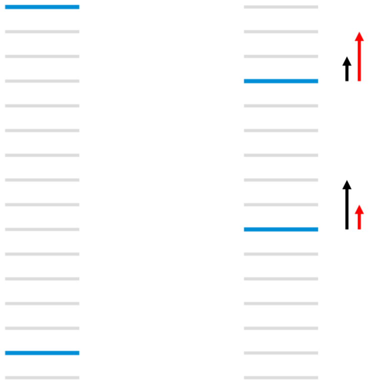
Listwise

Blue: relevant   Gray: non-relevant

NDCG and ERR higher for left but pairwise errors less for right

Due to strong position-based discounting in IR measures, errors at higer ranks are much more problematic than at lower ranks

But listwise metrics are non-continuous and non-differentiable



[Burges, 2010]

LambdaRank

Key observations:

- To train a model we dont need the costs themselves, only the gradients (of the costs w.r.t model scores)
- It is desired that the gradient be bigger for pairs of documents that produces a bigger impact in NDCG by swapping positions

LambdaRank [Burges et al., 2006]
Multiply actual gradients with the change in NDCG by swapping the rank positions of the two documents

$$\lambda_{LambdaRank} = \lambda_{RankNet} \cdot |\Delta NDCG| \tag{13}$$

LambdaMart

LambdaMART combines LambdaRank and MART (Multiple Additive Regression Trees).

- MART is a boosted tree model in which the output of the model is a linear combination of the outputs of a set of regression trees.
- While MART uses gradient boosted decision trees for prediction tasks, LambdaMART uses gradient boosted decision trees using a cost function derived from LambdaRank for solving a ranking task.
- LambdaMART is able to choose splits and leaf values that may decrease the utility for some queries, as long as the overall utility increases.
- On experimental datasets, LambdaMART has shown better results than LambdaRank and the original RankNet.

ListNet and ListMLE

According to the Luce model [Luce, 2005], given four items $\{d_1, d_2, d_3, d_4\}$ the probability of observing a particular rank-order, say $[d_2, d_1, d_4, d_3]$, is given by:

$$p(\pi|s) = \frac{\phi(s_2)}{\phi(s_1) + \phi(s_2) + \phi(s_3) + \phi(s_4)} \cdot \frac{\phi(s_1)}{\phi(s_1) + \phi(s_3) + \phi(s_4)} \cdot \frac{\phi(s_4)}{\phi(s_3) + \phi(s_4)} \tag{14}$$

where, π is a particular permutation and ϕ is a transformation (e.g., linear, exponential, or sigmoid) over the score s_i corresponding to item d_i

ListNet and ListMLE

ListNet [Cao et al., 2007]
Compute the probability distribution over all possible permutations based on model score and ground-truth labels. The loss is then given by the K-L divergence between these two distributions.

This is computationally very costly, computing permutations of only the top-K items makes it slightly less prohibitive

ListMLE [Xia et al., 2008]
Compute the probability of the ideal permutation based on the ground truth. However, with categorical labels more than one permutation is possible which makes this difficult.

Outline

Training under different levels of supervision

Data requirements for training an off-line L2R system

Query/document pairs that encode an ideal ranking given a particular query.

Ideal ranking? Relevance, preference, importance [Liu, 2009], novelty & diversity [Clarke et al., 2008].

What about personalization? Triples of user, query and document.

Related to evaluation. Pairs also used to compute popular off-line evaluation measures.

Graded or binary. ”documents may be relevant to a different degree” [Järvelin and Kekäläinen, 2000]

Absolute or relative? Zheng et al. [2007]

How to satisfy data-hungry models?

There are different ways to obtain query/document pairs.

Most expensive

1. Human judgments
2. Explicit user feedback
3. Implicit user feedback
4. Pseudo relevance

Least expensive

Human judgments

Human judges determine the relevance of a document for a given query.

How to determine candidate query/document pairs?

- Obtaining human judgments is expensive.
- List of queries: sample of incoming traffic or manually curated.
- Use an existing rankers to obtain rankings and pool the outputs [Sparck Jones and van Rijsbergen, 1976].
- Trade-off between number of queries (shallow) and judgments (deep) [Yilmaz and Robertson, 2009].

Explicit user feedback

When presenting results to the user, ask the user to explicitly judge the documents.

Unfortunately, users are only rarely willing to give explicit feedback [Joachims et al., 1997].

Extracting pairs from click-through data (training)

Extract implicit judgments from search engine interactions by users.

- Assumption: user clicks $\Rightarrow$ relevance (or, preference).
- Virtually unlimited data at very low cost, but interpretation is more difficult.
- Presentation bias: users are more likely to click higher-ranked links.
- How to deal with presentation bias? Joachims [2003] suggest to interleave different rankers and record preference.
- Chains of queries (i.e., search sessions) can be identified within logs and more fine-grained user preference can be extracted [Radlinski and Joachims, 2005].

Extracting pairs from click-through data (evaluation)

Clicks can also be used to evaluate different rankers.

- Radlinski et al. [2008] discuss how absolute metrics (e.g., abandonment rate) do not reliable reflect retrieval quality. However, relative metrics gathered using interleaving methods, do reflect retrieval quality.
- Carterette and Jones [2008] propose a method to predict relevance score of unjudged documents. Allows for comparisons across time and datasets.

Side-track: Online LTR

As mentioned earlier, we focus mostly on offline LTR. Besides an active learning set-up, where models are re-trained frequently, neural models have not yet conquered the online paradigm.

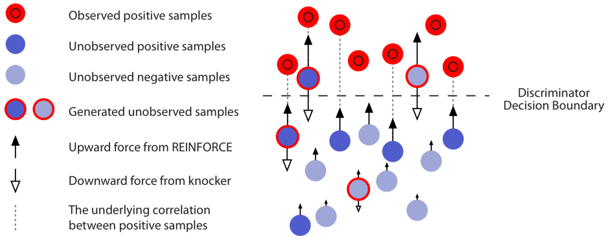
See the SIGIR'16 tutorial of Grotov and de Rijke [2016] for an overview.

Outline

IRGAN [Wang et al., 2017]

There are two main way of thinking about modeling retrieval:

- The generative retrieval focusing on predicting relevant documents given a query
- The discriminative retrieval focusing on predicting relevancy given a query-document pair.


Observed positive samples
Unobserved positive samples
Unobserved negative samples
Generated unobserved samples
Upward force from REINFORCE
Downward force from knocker
The underlying correlation between positive samples
Discriminator Decision Boundary

Main idea: a theoretical minimax game to iteratively optimize both of these models based on the idea of GAN.

Two cool, new ideas

- Learning to Rank with Query-level Semi-supervised Autoencoders [Xu et al., 2017]: Besides the reconstruction loss, they introduce extra supervision using a query-level constraint.
  - Objectives:
    - Minimizing the distance between its inputs and output (reconstruction loss)
    - Minimizing differences of the query-level retrieval performance between the inputs and the outputs.
- Alternating Pointwise and Pairwise Learning [Lei et al., 2017]
  - Try to get the best of both worlds: alternating between Pointwise and Pairwise loss over pairwise examples
  - Evaluated on personalized item ranking

Outline

Toolkits for off-line learning to rank

RankLib	: `https://sourceforge.net/p/lemur/wiki/RankLib`
shoelace	: `https://github.com/rjagerman/shoelace` [Jagerman et al., 2017]
QuickRank	: `http://quickrank.isti.cnr.it` [Capannini et al., 2016]
RankPy	: `https://bitbucket.org/tunystom/rankpy`
pyltr	: `https://github.com/jma127/pyltr`
jforests	: `https://github.com/yasserg/jforests` [Ganjisaffar et al., 2011]
XGBoost	: `https://github.com/dmlc/xgboost` [Chen and Guestrin, 2016]
SVMRank	: `https://www.cs.cornell.edu/people/tj/svm_light` [Joachims, 2006]
sofia-ml	: `https://code.google.com/archive/p/sofia-ml` [Sculley, 2009]
pysofia	: `https://pypi.python.org/pypi/pysofia`