

Outline

Morning program

- Preliminaries

- Text matching I

- Text matching II

Afternoon program

- Learning to rank**

- Modeling user behavior

- Generating responses

- Wrap up

Outline

Morning program

- Preliminaries

- Text matching I

- Text matching II

Afternoon program

Learning to rank

- Overview & basics

- Refresher of cross-entropy

- Pointwise loss

- Pairwise loss

- Listwise loss

- Different levels of supervision

- Toolkits

- Modeling user behavior

- Generating responses

- Wrap up

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 $f : \vec{x} \rightarrow \mathbb{R}$ 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.

Approaches

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 ($d_i \succ_q d_j$) 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)

Outline

Morning program

Preliminaries

Text matching I

Text matching II

Afternoon program

Learning to rank

Overview & basics

Refresher of cross-entropy

Pointwise loss

Pairwise loss

Listwise loss

Different levels of supervision

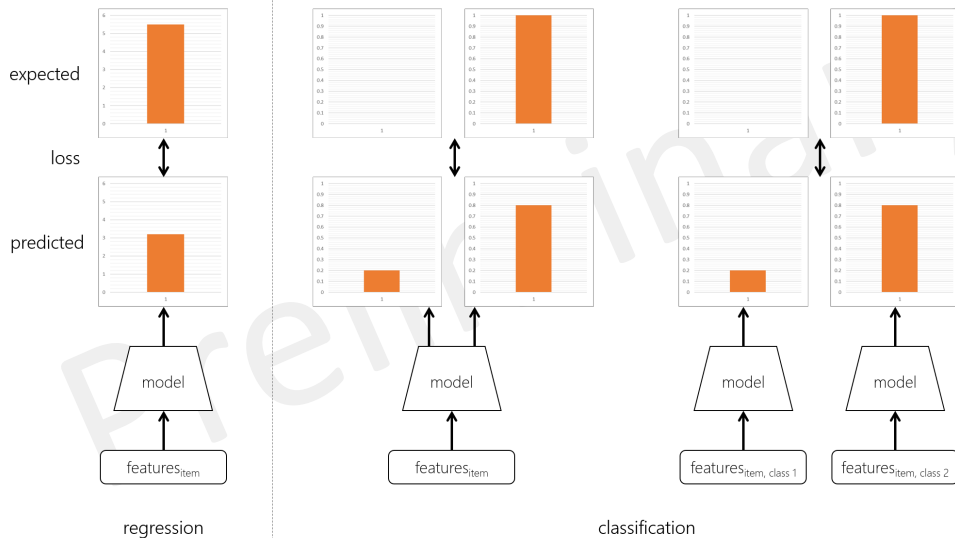
Toolkits

Modeling user behavior

Generating responses

Wrap up

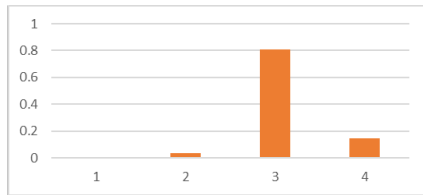
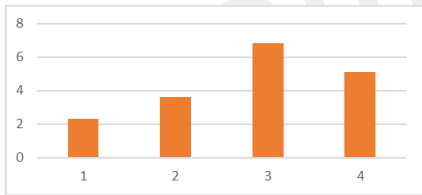
A quick refresher - Neural models for different tasks



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

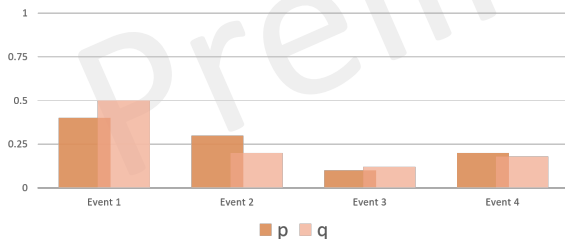
$$p(z_i) = \frac{e^{\gamma z_i}}{\sum_{z \in Z} e^{\gamma z}} \quad (\gamma \text{ is a constant}) \quad (2)$$



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,

$$CE(p, q) = - \sum_i p_i \log(q_i) \quad (3)$$



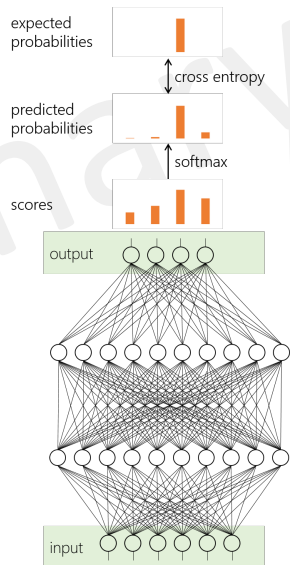
If $p_{correct} = 1$ and $p_i = 0$ for all other values of i then,

$$CE(p, q) = -\log(q_{correct}) \quad (4)$$

A quick refresher - What is the Cross Entropy with Softmax loss?

Cross entropy with softmax is a popular loss function for classification

$$\mathcal{L}_{\text{CE}} = -\log\left(\frac{e^{\gamma z_{\text{correct}}}}{\sum_{z \in Z} e^{\gamma z}}\right) \quad (5)$$



Outline

Morning program

- Preliminaries

- Text matching I

- Text matching II

Afternoon program

Learning to rank

- Overview & basics

- Refresher of cross-entropy

- Pointwise loss**

- Pairwise loss

- Listwise loss

- Different levels of supervision

- Toolkits

- Modeling user behavior

- Generating responses

- Wrap up

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,

$$\mathcal{L}_{Squared} = \|y_{q,d} - f(\vec{x}_{q,d})\|^2 \quad (6)$$

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],

$$\mathcal{L}_{\text{CE}}(q, d, y_{q,d}) = -\log\left(p(y_{q,d}|q, d)\right) = -\log\left(\frac{e^{\gamma \cdot s_{y_{q,d}}}}{\sum_{y \in Y} e^{\gamma \cdot s_y}}\right) \quad (7)$$

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

Outline

Morning program

- Preliminaries

- Text matching I

- Text matching II

Afternoon program

Learning to rank

- Overview & basics

- Refresher of cross-entropy

- Pointwise loss

- Pairwise loss**

- Listwise loss

- Different levels of supervision

- Toolkits

- Modeling user behavior

- Generating responses

- Wrap up

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 following form [Chen et al., 2009],

$$\mathcal{L}_{pairwise} = \phi(s_i - s_j) \quad (8)$$

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 ,

$$\mathcal{L}_{RankNet} = -\bar{p}_{ij} \log(p_{ij}) - \bar{p}_{ji} \log(p_{ji}) \quad (9)$$

$$= -\log(p_{ij}) \quad (10)$$

$$= \log(1 + e^{-\gamma(s_i - s_j)}) \quad (11)$$

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,

$$p(d^+|q) = \frac{e^{\gamma \cdot s(q, d^+)}}{\sum_{d \in D} e^{\gamma \cdot s(q, d)}} \quad (12)$$

The cross entropy loss is then given by,

$$\mathcal{L}_{\text{CE}}(q, d^+, D) = -\log(p(d^+|q)) \quad (13)$$

$$= -\log\left(\frac{e^{\gamma \cdot s(q, d^+)}}{\sum_{d \in D} e^{\gamma \cdot s(q, d)}}\right) \quad (14)$$

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

Morning program

- Preliminaries

- Text matching I

- Text matching II

Afternoon program

Learning to rank

- Overview & basics

- Refresher of cross-entropy

- Pointwise loss

- Pairwise loss

- Listwise loss**

- Different levels of supervision

- Toolkits

- Modeling user behavior

- Generating responses

- Wrap up

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 higher 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 don't 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

$$\lambda_{\text{LambdaRank}} = \lambda_{\text{RankNet}} \cdot |\Delta \text{NDCG}| \quad (15)$$

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:

$$p(\pi|s) = \frac{\phi(s_2)}{\phi(s_1) + \phi(s_2) + \phi(s_3) + \phi(s_4)} \cdot \frac{\phi(s_1)}{\phi(s_1) + \phi(s_3) + \phi(s_4)} \cdot \frac{\phi(s_4)}{\phi(s_3) + \phi(s_4)} \quad (16)$$

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

Morning program

Preliminaries

Text matching I

Text matching II

Afternoon program

Learning to rank

Overview & basics

Refresher of cross-entropy

Pointwise loss

Pairwise loss

Listwise loss

Different levels of supervision

Toolkits

Modeling user behavior

Generating responses

Wrap up

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

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 reliably 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.

Pseudo relevance judgments

Pseudo relevance collections (discussed first on Slide 96) can also be used to train LTR systems.

Web search Asadi et al. [2011] construct a pseudo relevance collection from anchor texts in a web corpus. LTR trained using pseudo relevance outperform non-supervised retrieval functions (e.g., BM25) on TREC collections.

Microblog search Berendsen et al. [2013] use hashtags as a topical relevance signal. Queries are constructed by sampling terms from tweets.

Personalized product search Ai et al. [2017] synthesize purchase behavior from Amazon user reviews. Queries and relevance are constructed according to the human-curated Amazon product categories [Van Gysel et al., 2016]. They learn vector space representations for query terms, users and products.

Outline

Morning program

- Preliminaries

- Text matching I

- Text matching II

Afternoon program

Learning to rank

- Overview & basics

- Refresher of cross-entropy

- Pointwise loss

- Pairwise loss

- Listwise loss

- Different levels of supervision

Toolkits

- Modeling user behavior

- Generating responses

- Wrap up

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]

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>