Optimizing a Ranking System with Annotations: Supervised Learning to Rank

Harrie Oosterhuis

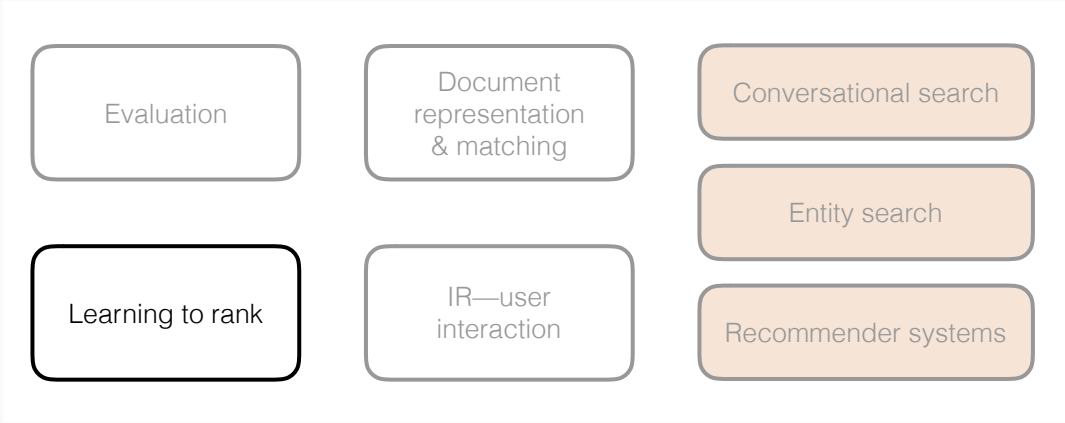
February 18, 2020

University of Amsterdam

oosterhuis@uva.nl

Partly based on the SIGIR 2017 Tutorial:
Neural Networks for Information Retrieval
Tom Kenter, Alexey Borisov, Christophe Van Gysel, Mostafa Dehghani, Maarten de Rijke, and Bhaskar Mitra.

Course Overview: Learning to rank (LTR)



This Lecture

Goals for this lecture:

- Understand why Learning to Rank is a separate category in Machine Learning.
- Cover the main Learning to Rank approaches:
  - Pointwise, Pairwise, and Listwise.
- Understand when it should be applied.
- Be able to implement them yourself (with little additional help).

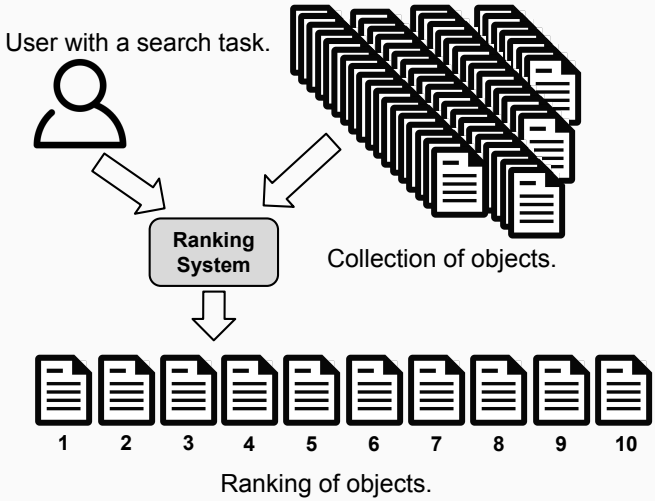
Introduction

Learning to rank (LTR)

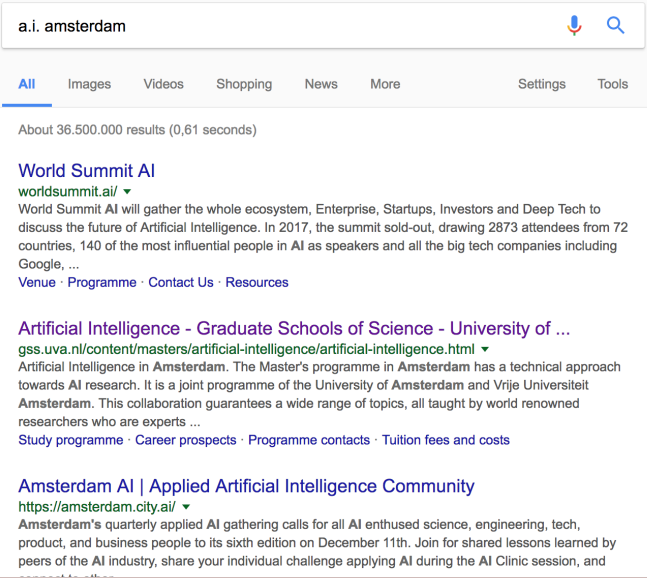
Definition

"... 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]

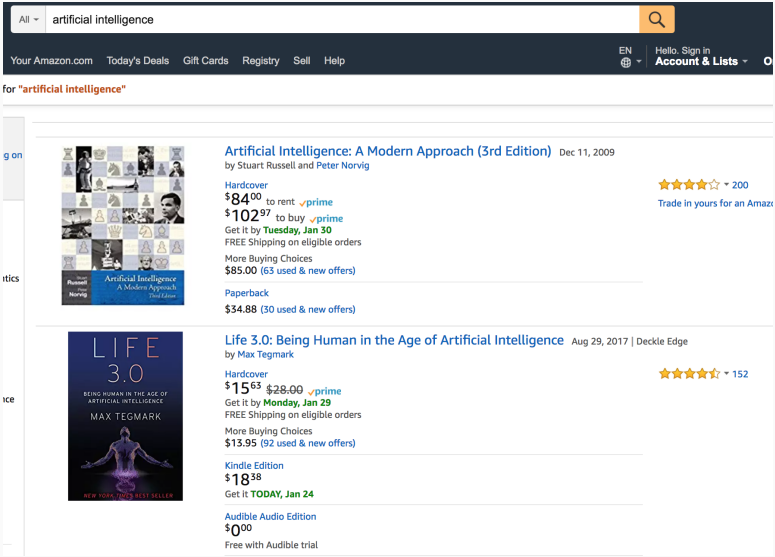
Learning to rank (LTR)



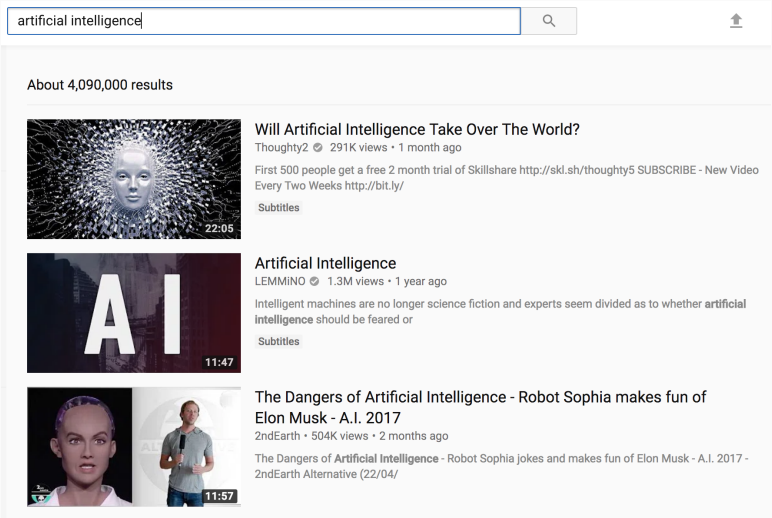
Learning to rank (LTR)



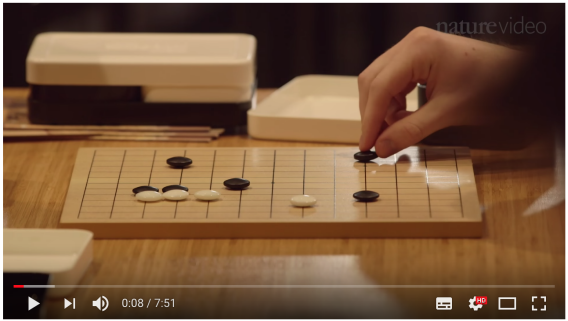
Learning to rank (LTR)



Learning to rank (LTR)



Learning to rank (LTR)



The computer that mastered Go

843,641 views

5K 132 SHARE

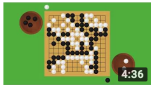
nature video
Published on 27 Jan 2016

SUBSCRIBE 223K

Go is an ancient Chinese board game, often viewed as the game computers could never play. Now researchers from Google-owned company DeepMind have proven the naysayers wrong,

Up next AUTOPLAY

New DeepMind AI Beats AlphaGo 100-0 | Two Minute
Two Minute Papers
93K views



Go - Basic Rules
Udacity
21K views

Let's Learn the Rules of Go!
Sunday Go Lessons - Videos on the
123K views

Go for Beginners: Short 9x9 Game Walkthroughs vs. igowin
Jonathan Markowitz
114K views

Google's Deep Mind Explained! - Self Learning A.I.
ColdFusion
2.4M views

Learning to rank (LTR)

Ranking system are used for a large range of tasks:

1. Web search.
2. Product search.
3. Image search.
4. Video search.
5. Email search.
6. Personal document search.
7. Recommendation:
    1. Video recommendation.
    2. Advertisement recommendation.
    3. Music recommendation.
8. etc.

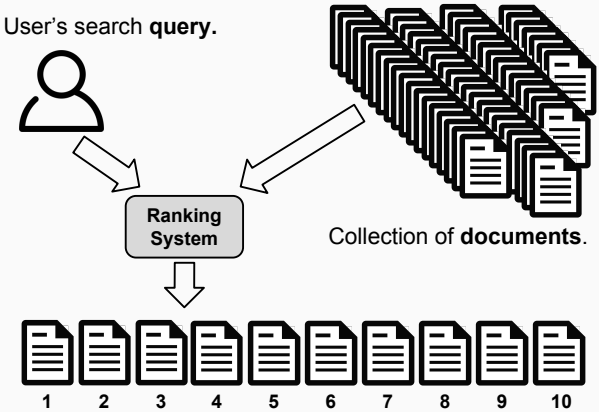
Learning to rank (LTR)

For this lecture we will focus on **web search.**

Users issue **text queries**, the objects to rank are textual **documents**.
Documents should be ranked according to **relevance**.

Learning to rank (LTR)

How do we do this?



Ranking of **documents** according to relevance.

Signals in Ranking Systems

You have seen many signals that can help:

1. Document-Query overlap
2. TF-IDF
3. BM25
4. Language Modelling
5. Neural Matching
6. ...

Signals in Ranking Systems

Relevance signals:

1. Document-Query overlap
2. TF-IDF
3. BM25
4. Language Modelling
5. Neural Matching
6. ...

What about other signals?

Signals in Ranking Systems

Relevance signals:

1. Document-Query overlap
2. TF-IDF
3. BM25
4. Language Modelling
5. Neural Matching
6. ...

Other signals:

1. Pagerank (Do other pages link to this one.)
2. Webpage popularity
3. Spam detection (e.g. clickbait)
4. Query information (type, topic, etc)
5. Language of user/page/query
6. ...

Signals in Ranking Systems

Web search engines use a lot of these signals:

1. **Bing:** 136+ signals/features
2. **Istella:** 220+ signals/features
3. **Yahoo:** 700+ signals/features

How do we combine all these signals to create rankings?

With **machine learning**.

LTR: Preliminaries & Goal

Problem Formulation

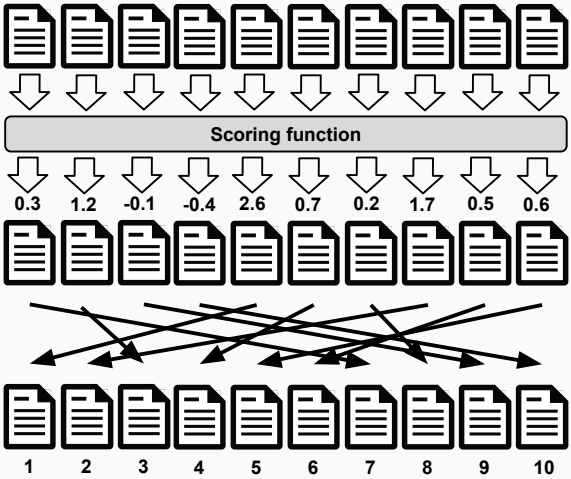
Representation:

- Represent the document and query in a format that a ML model can use: **a numerical vector** $\vec{x} \in \mathbb{R}^n$

Prediction:

- Then a **ranking model** $f : \vec{x} \to \mathbb{R}$ is optimized to score each document-query combination so that relevant documents are scored higher.
- In mathematical terms: f maps vector from a real-valued scores.

Problem Formulation: Illustration



Features

We have already listed a lot of possible features that can be used.

Traditionally features are hand-crafted to encode IR insights,
nowadays we also have *deep learned* features..

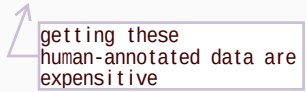
They can be categorized as:

- **Document-only** or *static* features (e.g., document length)
- **Document-Query-combination** or *dynamic* features (e.g., BM25)
- **Query-only** features (e.g., query length)

Learning to rank data

Models can be trained on different data:

- **Offline or Supervised** LTR: learn from annotated data.
  - Expensive and time-consuming.
  - Provides ground-truth.
- **Online/Counterfactual** LTR: learn from user interactions.
  - Virtually free and easy to obtain.
  - Hard to interpret.



Supervised Data

how to collect labels

This lecture is on **offline** LTR, we'll assume that we are rich.

Data is then obtained by:

1. Pay some humans to be annotators.
2. (Train them to be good annotators.)
3. Collect a set of queries.
4. Preselect a large (but not too large) set of documents per query.
5. Show document-query pairs to annotators.
6. Annotators rate every document-query pair on their relevance, (e.g. on a scale from 0 to 4).

Learning to Rank: Goal

Thus we have:

- Feature representation of document-query pairs: $\vec{x}_{q,d} \in \mathbb{R}$.
- Labels indicating the relevance of document-query pairs: $y_{q,d} \in [0, 4]$.

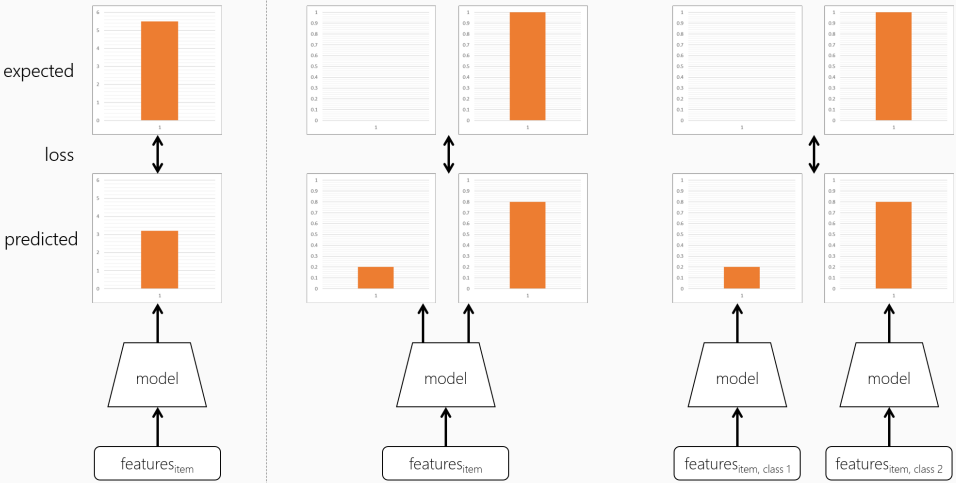
And we want:

- A function $f : \vec{x} \to \mathbb{R}$ that scores documents.
- To get the best ranking by sorting according to $f(\vec{x})$.

How do we find f?

Refresher on Standard ML Losses

A quick refresher - Neural models for different tasks

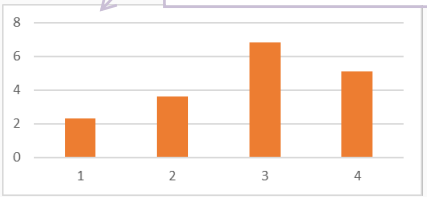


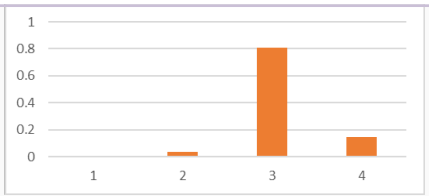
A quick refresher - What is the Softmax function?

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

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

soft max func maps scores in each class into distribution on classes

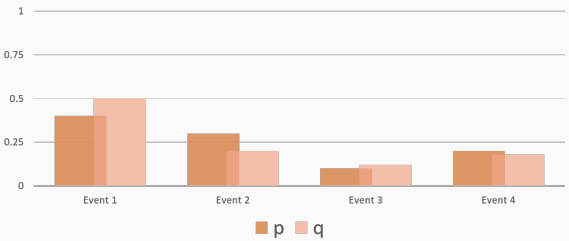




A quick refresher - What is Cross Entropy?

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

$$CE(p, q) = -\sum_i p_i \log(q_i) \qquad (2)$$



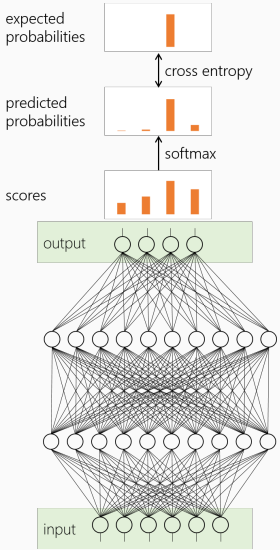
If $p_{correct} = 1$ and $p_i = 0$ for all other values of i then,

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

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

Cross entropy with softmax is a popular loss function for classification:

$$\mathcal{L}_{\text{CE}} = -log\Big(\frac{e^{\gamma z_{correct}}}{\sum_{z \in Z} e^{\gamma z}}\Big) \tag{4}$$



First approach to LTR

Thus we have:

- Feature representation of document-query pairs: $\vec{x}_{q,d} \in \mathbb{R}$.
- Labels indicating the relevance of document-query pairs: $y_{q,d} \in [0, 5]$.

And we want:

- A function $f : \vec{x} \to \mathbb{R}$ that scores documents.
- To get the best ranking by sorting according to $f(\vec{x})$.

How do we find f?

The Pointwise Approach

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,

$$\mathcal{L}_{Squared}(q, d, y_{q,d}) = \|y_{q,d} - f(\vec{x}_{q,d})\|^2 \tag{5}$$

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],

$$\mathcal{L}_{\text{CE}}(q, d, y_{q,d}) = -log\Big(p(y_{q,d}|q, d)\Big) = -log\Big(\frac{e^{\gamma \cdot s_{y_{q,d}}}}{\sum_{y \in Y} e^{\gamma \cdot s_y}}\Big) \tag{6}$$

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

Pointwise approaches to LTR

Regression loss

$$\mathcal{L}_{Squared} = \sum_{q,d} \|y_{q,d} - f(\vec{x}_{q,d})\|^2 \tag{7}$$

Classification loss

$$\mathcal{L}_{\text{CE}} = \sum_{q,d} -log\Big(p(y_{q,d}|q,d)\Big) = \sum_{q,d} -log\Big(\frac{e^{\gamma \cdot s_{y_{q,d}}}}{\sum_{y \in Y} e^{\gamma \cdot s_y}}\Big) \tag{8}$$

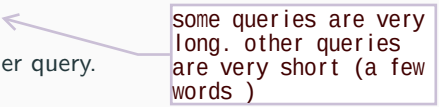
where, $s_{y_{q,d}}$ is the model's score for label $y_{q,d}$.

What are **issues** with these approaches?

Minor issues with pointwise approaches

Some minor issues are:

- Class imbalance:
  - many irrelevant documents and very few relevant documents.
- Query level feature normalization is needed:
  - the distribution of features differs greatly per query.



These can be overcome.

Pointwise approaches to LTR

Regression loss

$$\mathcal{L}_{Squared} = \sum_{q,d} \|y_{q,d} - f(\vec{x}_{q,d})\|^2 \tag{9}$$

Classification loss

$$\mathcal{L}_{\text{CE}} = \sum_{q,d} -log\Big(p(y_{q,d}|q,d)\Big) = \sum_{q,d} -log\Big(\frac{e^{\gamma \cdot s_{y_{q,d}}}}{\sum_{y \in Y} e^{\gamma \cdot s_y}}\Big) \tag{10}$$

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

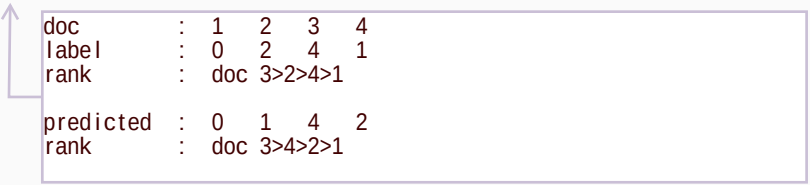
What is **fundamentally wrong** with these methods?

The problem with pointwise LTR

Ranking is not a regression or classification problem.

A document-level loss does not work for ranking problems because document scores should not be considered independently.

In other words, pointwise methods **do not directly optimize ranking quality**.



The problem with pointwise LTR illustrated

What is the loss here?

$$\mathcal{L}_{Squared} = \sum_{q,d} \|y_{q,d} - f(\vec{x}_{q,d})\|^2 \tag{11}$$

The problem with pointwise LTR illustrated

What is the loss here?

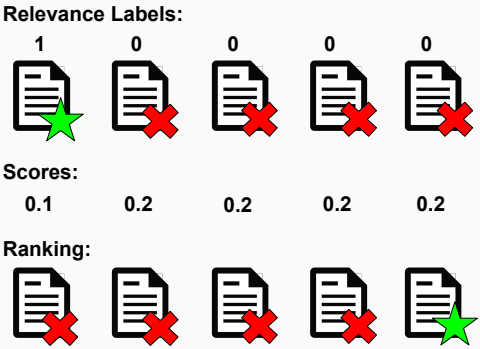
$$\mathcal{L}_{Squared} = 1.16 \tag{12}$$

The problem with pointwise LTR illustrated

What is the loss here?

$$\mathcal{L}_{Squared} = \sum_{q,d} \|y_{q,d} - f(\vec{x}_{q,d})\|^2 \tag{13}$$

The problem with pointwise LTR illustrated



What is the loss here?

$$\mathcal{L}_{Squared} = 0.97 \tag{14}$$

The problem with pointwise LTR illustrated

What is the loss here?

$$\mathcal{L}_{Squared} = 1.16 \tag{15}$$

Solution to pointwise?

Ranking is not a regression or classification problem.

A document-level loss does not work for ranking problems because document scores should not be considered independently.

In other words, pointwise method **do not directly optimize ranking quality**;

a lower loss does not mean a better ranking!

How can we solve this problem?

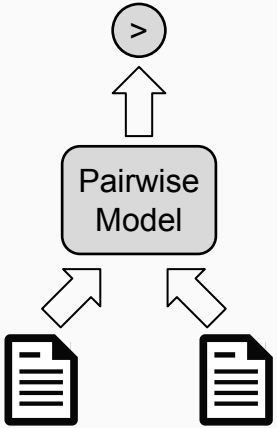
The Pairwise Approach

Pairwise objectives

Instead of looking at document-level, consider pairs of documents.

$$y_{q,d_i} > y_{q,d_j} \rightarrow d_i \succ d_j \tag{16}$$

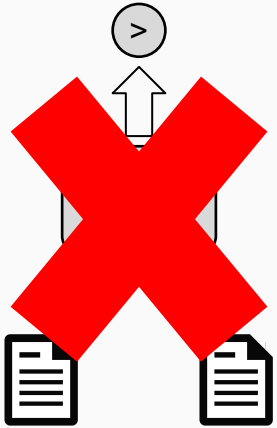
Naive Pairwise Model



Pairwise Model

$$P(d_i \succ d_j) = f(\vec{x}_i, \vec{x}_j) \tag{17}$$

Naive Pairwise Model



$$P(d_i \succ d_j) = f(\vec{x}_i, \vec{x}_j) \tag{18}$$

Naive Pairwise Model

Do **not** change the model to take **document pairs as input!**

$$P(d_i \succ d_j) = f(\vec{x}_i, \vec{x}_j) \tag{19}$$

This method would be quadratic in complexity: $O(N^2)$, during **inference**.

Pair-preferences have to be aggregated somehow.
This can lead to paradoxical situations:

$$\begin{aligned} d_1 &\succ d_2 \\ d_2 &\succ d_3 \\ d_3 &\succ d_1 \end{aligned} \tag{20}$$

The Pairwise Approach

The scoring model remains **unchanged**:

$$f(\vec{x}_i) = s_i \tag{21}$$

But the loss function is based on document pairs:

$$\mathcal{L}_{pairwise} = \sum_{d_i \succ d_j} \phi(s_i - s_j) \tag{22}$$

Thus we still score documents and then order according to scores.

Pairwise Loss Functions

Pairwise objectives

Pairwise loss minimizes the **average number of inversio**

minimize the mistake which is: model predicts that doc i is ranked higher than j. but label says i is ranked lower than j. we want to minimize this mistake

- $d_i \succ_q d_j$ but d_j is ranked higher than d_i

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

$$\mathcal{L}_{pairwise} = \phi(s_i - s_j) \tag{23}$$

where, ϕ can be,

goal: pushes documents away from each other if there's a relevance difference.

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

RankNet

RankNet

RankNet [Burges et al., 2005] is a *pairwise* loss function—popular choice for training neural LTR 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,

$$\begin{aligned} \mathcal{L}_{RankNet} &= -\bar{P}_{ij} \log(P_{ij}) - \bar{P}_{ji} \log(P_{ji}) \\ &= -\log(P_{ij}) \\ &= log(1 + e^{-\gamma(s_i - s_j)}) \end{aligned} \tag{24}$$

Deep Dive into RankNet

There is a famous factorization of RankNet [Burges, 2015, Burges et al., 2005]

Let $S_{ij} \in \{-1, 0, 1\}$ indicate the preference between d_i and d_j.
Then the desired probability for a pair is:

$$\bar{P}(d_i \succ d_j) = \frac{1}{2}(1 - S_{ij}). \tag{25}$$

The predicted probability is:

$$P(d_i \succ d_j) = \frac{1}{1 + e^{-\gamma(s_i - s_j)}}. \tag{26}$$

The cross-entropy loss is then:

$$\mathcal{L}_{ij} = \frac{1}{2}(1 - S_{ij})\gamma(s_i - s_j) + \log(1 + e^{-\gamma(s_i - s_j)}). \tag{27}$$

The cross-entropy loss is then:

$$\mathcal{L}_{ij} = \frac{1}{2}(1 - S_{ij})\gamma(s_i - s_j) + \log(1 + e^{-\gamma(s_i - s_j)}). \tag{28}$$

The derivate w.r.t. s_i:

$$\frac{\delta \mathcal{L}_{ij}}{\delta s_i} = \gamma(\frac{1}{2}(1 - S_{ij}) - \frac{1}{1 + e^{-\gamma(s_i - s_j)}}) = -\frac{\delta \mathcal{L}_{ij}}{\delta s_j}. \tag{29}$$

Then we can factorize the loss it so that:

$$\frac{\delta \mathcal{L}_{ij}}{\delta w} = \frac{\delta \mathcal{L}_{ij}}{\delta s_i}\frac{\delta s_i}{\delta w} + \frac{\delta \mathcal{L}_{ij}}{\delta s_j}\frac{\delta s_j}{\delta w} = \gamma(\frac{1}{2}(1 - S_{ij}) - \frac{1}{1 + e^{-\gamma(s_i - s_j)}})(\frac{\delta s_i}{\delta w} - \frac{\delta s_j}{\delta w}). \tag{30}$$

The factorized cross entropy loss:

$$\frac{\delta \mathcal{L}_{ij}}{\delta w} = \gamma(\frac{1}{2}(1 - S_{ij}) - \frac{1}{1 + e^{-\gamma(s_i - s_j)}})(\frac{\delta s_i}{\delta w} - \frac{\delta s_j}{\delta w}). \tag{31}$$

We choose λ so that:

$$\frac{\delta \mathcal{L}_{ij}}{\delta w} = \lambda_{ij}(\frac{\delta s_i}{\delta w} - \frac{\delta s_j}{\delta w}), \tag{32}$$

where:

$$\lambda_{ij} = \gamma(\frac{1}{2}(1 - S_{ij}) - \frac{1}{1 + e^{-\gamma(s_i - s_j)}}). \tag{33}$$

These lambdas act like *forces* pushing pairs of documents apart or together.

$$\lambda_{ij} = \gamma(\frac{1}{2}(1 - S_{ij}) - \frac{1}{1 + e^{-\gamma(s_i - s_j)}}). \tag{34}$$

These lambdas act like *forces* pushing pairs of documents apart or together.

On document level the same can be done:

$$\lambda_i = \sum_j \lambda_{ij} \tag{35}$$

The Pairwise Approach

The scoring model scores documents independently: $f(\vec{x}_{d_i}) = s_i$.

The loss is based on document pairs, and minimizes the number of incorrect inversions:

$$\mathcal{L}_{pairwise} = \sum_{d_i \succ d_j} \phi(s_i - s_j) \tag{36}$$

For instance, RankNet:

$$\mathcal{L}_{RankNet} = \sum_{d_i \succ d_j} log(1 + e^{-\gamma(s_i - s_j)}) \tag{37}$$

What is **wrong** with this approach?

The Pairwise Approach

Minor issue: RankNet is based on virtual probabilities: $P(d_i \succ d_j)$.

In reality the ranking model does not follow these probabilities.

Not elegant, but not a big deal.

The Pairwise Approach

The scoring model scores documents independently: $f(\vec{x}_{d_i}) = s_i$.

The loss is based on document pairs, and minimizes the number of incorrect inversions:

$$\mathcal{L}_{pairwise} = \sum_{d_i \succ d_j} \phi(s_i - s_j) \tag{38}$$

For instance, RankNet:

$$\mathcal{L}_{RankNet} = \sum_{d_i \succ d_j} log(1 + e^{-\gamma(s_i - s_j)}) \tag{39}$$

What is **fundamentally wrong** with this approach?

Problem with the Pairwise Approach

Ranking 1:

Ranking 2:

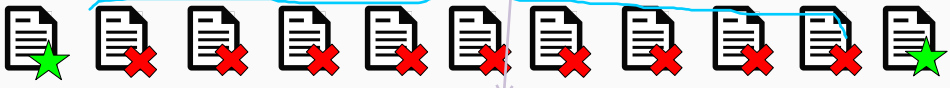
Which ranking do you think is better?

How many inversions does each ranking get correct?

Problem with the Pairwise Approach

correct means: doc2>doc3, doc3>doc4.this ranking order is the same as labeled ranking order.
we don't compare the predicted score with labeled score

Ranking 1:



Pairs correct: 9

Ranking 2:

pairwise loss func prefer result 2. but we prefer result 1. because, if 1000doc are relevant, the docs in the middle position (rank as 500th)will not be read by users. not every doc pair (doci, docj) are equally important.
即实际情况是 doc2>doc3 比 doc3>doc4更重要

The bottom ranking is better than the top according to the pairwise approach!

The Pairwise Approach

The scoring model scores documents independently: $f(\vec{x}_{d_i}) = s_i$.
The loss is based on document pairs, and minimizes the number of incorrect inversions:

$$\mathcal{L}_{pairwise} = \sum_{d_i \succ d_j} \phi(s_i - s_j) \tag{40}$$

However, **not every document pair is equally important**.

It is much **more important to get the correct ordering of top documents** than of the bottom documents.

For instance, the order of the top-5 is much more important than the order of documents after position 10.

The Listwise Approach

Holy grail of LTR

The **fundamental problem** with the approaches so far is that they did not optimize **ranking quality** directly.

A LTR method should directly optimize the ranking metric we care about.

What ranking metrics do we care about?

Evaluation Metrics in IR

Ranking metrics can range from simple:

$$precision(R) = \frac{1}{|R|} \sum_{R_i} relevance(R_i) \tag{41}$$

to much more complex, e.g. discounted cumulative gain:

$$DCG(R) = \sum_{R_i} \frac{2^{relevance(R_i)-1}}{\log(i+1)} \tag{42}$$

Evaluation Metrics in IR

How do we optimize for these metrics?

$$precision(R) = \frac{1}{|R|} \sum_{R_i} relevance(R_i) \tag{43}$$

$$\frac{\delta}{\delta w} precision(R) = \quad ??? \tag{44}$$

for discounted cumulative gain:

it is a step function. either doc i is in top 10, or not. there is no status in between.

$$DCG(R) = \sum_{R_i} \frac{2^{relevance(R_i)-1}}{\log(i+1)} \tag{45}$$

$$\frac{\delta}{\delta w} DCG(R) = \quad ??? \tag{46}$$

These metrics are **non-continuous** and **non-differentiable**.

Listwise

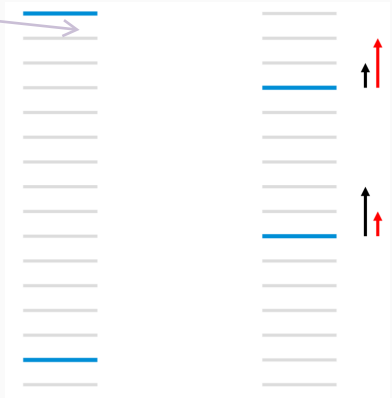
pairwise loss give low error in the right result. we want listwise loss to improve the right result. we want the two blue relevant doc to be ranked as high as possible. (because grey are non-relevant doc)

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

LambdaRank

Key observations:

- To train a model we do not need the costs themselves, only the gradients (of the costs w.r.t. model scores).
- The gradient should 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:

$$\lambda_{LambdaRank} = \lambda_{RankNet} \cdot |\Delta \text{NDCG}| \tag{47}$$

LambdaRank

LambdaRank [Burges et al., 2006]

Multiply actual gradients with the change in NDCG by swapping the rank positions of the two documents

$$\lambda_{LambdaRank} = \lambda_{RankNet} \cdot |\Delta \text{NDCG}| \tag{48}$$

This approach also works with other metrics, e.g. $|\Delta \text{Precision}|$

Empirically LambdaRank was shown to directly optimize IR metrics.

Recently, it was **theoretically proven** that LambdaRank optimizes a **lower bound** on certain IR metrics [Wang et al., 2018]

通过 push up the lower bound, the IR metrics will be optimized. so lambdarank is an approximation

ListNet and ListMLE

ListNet and ListMLE

Create a probabilistic model for ranking, which is differentiable.

Sample documents from a Plackett-Luce distribution:

$$P(d_i) = \frac{\phi(s_i)}{\sum_{d_j \in D} \phi(s_j)} \tag{49}$$

For instance, $\phi(s_i) = e^{s_i}$:

$$P(d_i) = \frac{e^{s_i}}{\sum_{d_j \in D} e^{s_j}} \tag{50}$$

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:

d2>d1>d4>d3

$$P(\pi|s) = \frac{\phi(s_2)}{\phi(s_1) + \phi(s_2) + \phi(s_3) + \phi(s_4)} \cdot \frac{\phi(s_1)}{\phi(s_1) + \phi(s_3) + \phi(s_4)} \cdot \frac{\phi(s_4)}{\phi(s_3) + \phi(s_4)} \tag{51}$$

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.

Conclusion

Learning to Rank Quick Recap

Learning to Rank

Ranking is very important in places were **search or recommendation** is involved.
Methods should **scale** to large collections and work **fast** enough to help users.
Search engines use large numbers of signals/features.

Learning to Rank Quick Recap

- **Pointwise Approach**
  - Predict the **relevance per item**, simple but very naive.
  - **Ignores** that **ordering** of items is what matters.
- **Pairwise Approach**
  - **Loss** based on **document pairs**, minimize the number of incorrect inversions.
  - Ignores that **not** all document pairs have the **same impact**.
  - Often used in the industry.
- **Listwise Approach**
  - Tries to **optimize** for **IR metrics**, but they are **not differentiable**.
  - **Approximations** by heuristics, bounding or probabilistic approaches to ranking.
  - Best approach out of the three.

very bad

does the job. has been widely applied. but not good enough

because derivative of that metrics is not differentiable, so we use approximation. lambda trick is a heuristic approximation

My Personal Opinion on the Point/Pair/List-wise Distiction

I do not think the distinction between pairwise and listwise is helpful:

- The derivatives of listwise losses always reduce to weighted pairwise derivatives (LambdaRank is explicit, ListNet does this implicitly).
- The LambdaRank with the Average Relevant Rank metric is equivalent to the pairwise RankNet loss.
- In the field there are often categorization mistakes, e.g. calling LambdaRank a pairwise method.

Personally I prefer non-ranking losses (pointwise) vs. ranking losses (pairwise & listwise).

Conclusion

In this lecture we discussed:

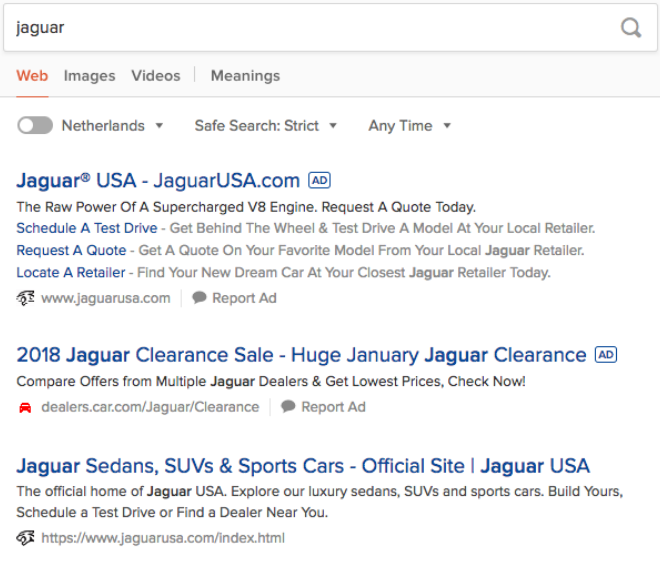
- why Learning to Rank is a separate category in Machine Learning.
- the main Learning to Rank approaches:
  - Pointwise, Pairwise, and Listwise.
- the problems/value of each approach.
- the most important methods:
  - RankNet, LambdaRank, ListNet.

Future Directions

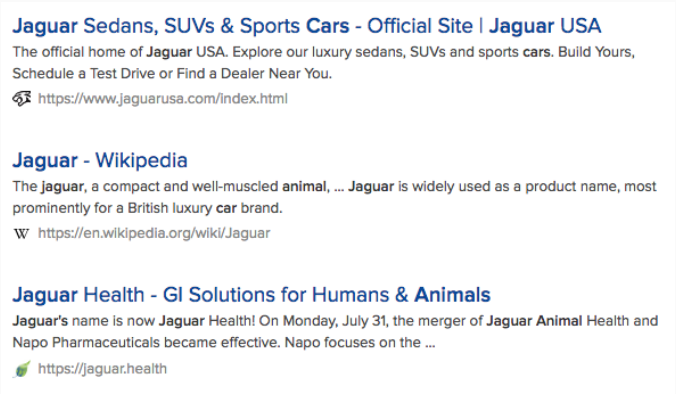
What is left to solve?

With Listwise approaches, isn't Learning to Rank solved?

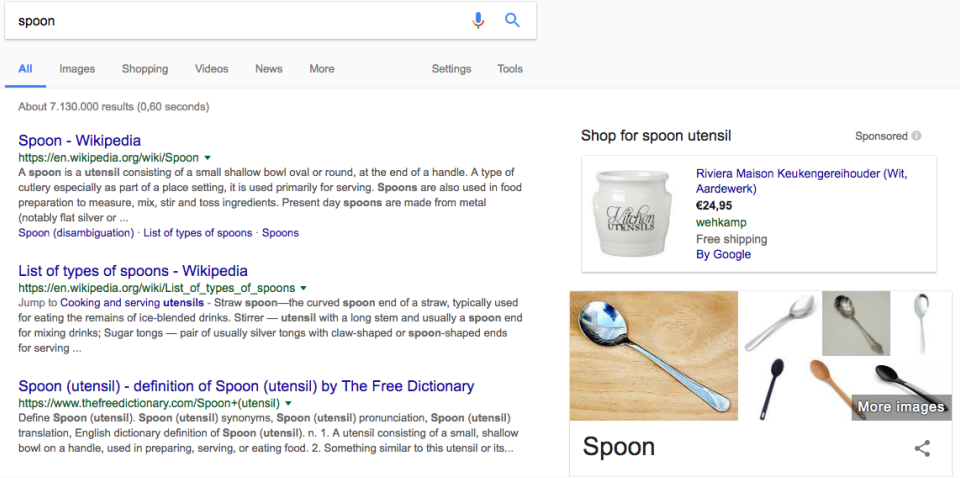
Diversity


jaguar
Web
Images
Videos
Meanings
Netherlands
Safe Search: Strict
Any Time
Jaguar® USA - JaguarUSA.com
AD
The Raw Power Of A Supercharged V8 Engine. Request A Quote Today.
Schedule A Test Drive - Get Behind The Wheel & Test Drive A Model At Your Local Retailer.
Request A Quote - Get A Quote On Your Favorite Model From Your Local Jaguar Retailer.
Locate A Retailer - Find Your New Dream Car At Your Closest Jaguar Retailer Today.
www.jaguarusa.com
Report Ad
2018 Jaguar Clearance Sale - Huge January Jaguar Clearance
AD
Compare Offers from Multiple Jaguar Dealers & Get Lowest Prices, Check Now!
dealers.car.com/Jaguar/Clearance
Report Ad
Jaguar Sedans, SUVs & Sports Cars - Official Site | Jaguar USA
The official home of Jaguar USA. Explore our luxury sedans, SUVs and sports cars. Build Yours, Schedule a Test Drive or Find a Dealer Near You.
https://www.jaguarusa.com/index.html

Diversity


Jaguar Sedans, SUVs & Sports Cars - Official Site | Jaguar USA
The official home of Jaguar USA. Explore our luxury sedans, SUVs and sports cars. Build Yours, Schedule a Test Drive or Find a Dealer Near You.
https://www.jaguarusa.com/index.html
Jaguar - Wikipedia
The jaguar, a compact and well-muscled animal, ... Jaguar is widely used as a product name, most prominently for a British luxury car brand.
https://en.wikipedia.org/wiki/Jaguar
Jaguar Health - GI Solutions for Humans & Animals
Jaguar's name is now Jaguar Health! On Monday, July 31, the merger of Jaguar Animal Health and Napo Pharmaceuticals became effective. Napo focuses on the ...
https://jaguar.health

Personalization



Personalization

spoon

All Images Videos Shopping News More Settings Tools

About 69.400.000 results (0,38 seconds)

SPOON - HOT THOUGHTS
www.spoontheband.com/
Official site for **Spoon**. New Album 'Hot Thoughts' out now.

SPOON (@spoontheband) · Twitter
https://twitter.com/spoontheband

Gonna be extra in london june 2 at @allpointseastuk. www.allpointseastfestiv… pic.twitter.com/PCak1CG…
4 hours ago · Twitter

A true inspiration pic.twitter.com/auueaSg…
10 hours ago · Twitter

We is rill thrilled to add DC's wonderful @sneaks_zia to the lineup for our March US East Coast dates. www.spoontheband.com/#l… pic.twitter.com/V7kKaKG…
18 hours ago · Twitter

Spoon (band) - Wikipedia
https://en.wikipedia.org/wiki/Spoon_(band)
Spoon is an American rock band formed in Austin, Texas. The band comprises Britt Daniel (vocals, guitar), Jim Eno (drums), Rob Pope and Alex Fischel (keyboard, guitar). Critics have described the band's musical style as indie rock, indie pop, art rock, and experimental rock. Formed in 1993 in Austin, Texas by Britt Daniel ...
Spoon discography · Britt Daniel · Hot Thoughts · Jim Eno

Spoon
Rock band

spoontheband.com

Available on
YouTube
Spotify
Deezer

Spoon is an American rock band formed in Austin, Texas. The band comprises Britt Daniel, Jim Eno, Rob Pope, and Alex Fischel. Critics have described the band's musical style as indie rock, indie pop, art rock, and experimental rock. Wikipedia

Origin: Austin, Texas, United States (1993)

Members: Britt Daniel, Jim Eno, Rob Pope, Eric Harvey, Alex Fischel, Joshua Zarbo, Roman Kuebler

Dynamic Relevance

trump

All News Images Videos Maps More Settings Tools

About 263.000.000 results (0,46 seconds)

Top stories

Trump Says He Is Willing to Speak Under Oath to Mueller

The New York Times

12 hours ago

Trump stirs pot with Mueller interview offer

CNN.com

3 hours ago

Davos 2018: Theresa May and Donald Trump to meet - live updates

The Guardian

5 mins ago

More for trump

Donald J. Trump (@realDonaldTrump) · Twitter

https://twitter.com/realDonaldTrump

Will soon be heading to Davos, Switzerland, to tell the world how great America is and is doing. Our economy is

It was my great honor to welcome Mayor's from across America to the WH. My Administration will always

Earlier today, I spoke with @GovMattBevin of Kentucky regarding yesterday's shooting at Marshall County

Donald Trump

45th U.S. President

Donald John Trump is the 45th and current President of the United States, in office since January 20, 2017. Before entering politics, he was a businessman and television personality. Trump was born and grew up in the New York City borough of Queens. Wikipedia

Born: June 14, 1946 (age 71), Jamaica Hospital Medical Center, New York City, New York, United States

Height: 1.91 m

Net worth: 3.1 billion USD (2018) Forbes

Spouse: Melania Trump (m. 2005), Marla Maples (m. 1993–1999), Ivana Trump (m. 1977–1992)

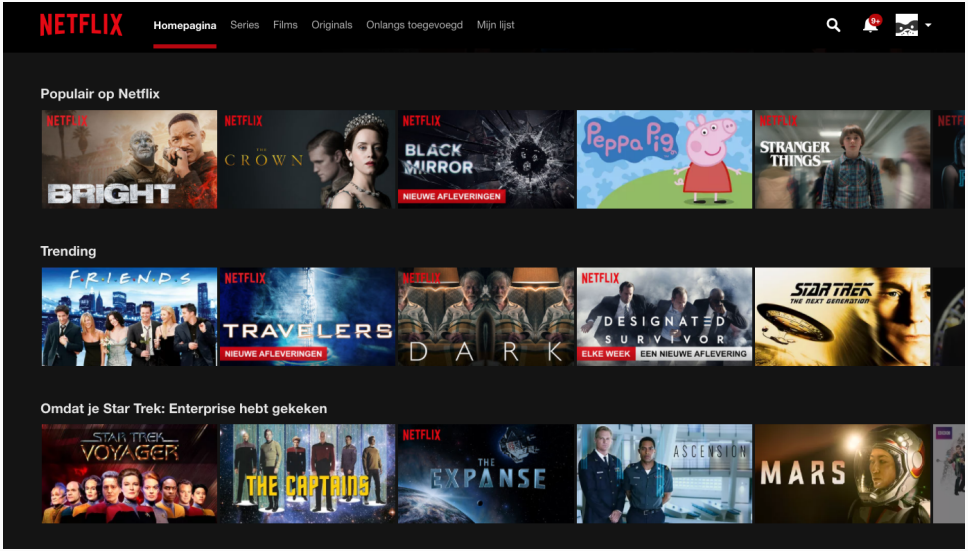
Education: Wharton School of the University of Pennsylvania (1966–1968), MORE

Quotes View 7+ more

What separates the winners from the losers is how a person reacts to each new twist of fate.

All of the women on 'The Apprentice' flirted with me — consciously or

Complicated Layouts



Human Annotators

Are the annotators **representative** of the user base you will have?

Is it **ethical** for the annotators to look at the documents in the first place?

Offline Learning to Rank Limitations

It is almost infeasible to get labeled data that accounts for:

Diversity	Novelty & diversity in search results [Clarke et al., 2008].
Personalization	Triples of user, query and document.
Dynamic Relevance	Content where relevance changes continuously.
Complicated Layouts	What is relevant and where should it be displayed?
Annotators	Do the annotators agree with your userbase?
Ethical problems	What if the document content is private?

LTR from User Interactions

In upcoming lectures we will look at how we can **learn from user interactions**.

End - Q&A

Thank you for your attention!

Questions?

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

References i

C. Burges. Ranknet: A ranking retrospective, 2015. Accessed July 16, 2017.

C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pages 89–96. ACM, 2005.

C. J. Burges. From ranknet to lambdarank to lambdamart: An overview. *Learning*, 11(23-581):81, 2010.

C. J. Burges, R. Ragno, and Q. V. Le. Learning to rank with nonsmooth cost functions. In *NIPS*, volume 6, pages 193–200, 2006.

Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, pages 129–136. ACM, 2007.

G. Capannini, C. Lucchese, F. M. Nardini, S. Orlando, R. Perego, and N. Tonellotto. Quality versus efficiency in document scoring with learning-to-rank models. *IPM*, 52(6):1161–1177, 2016.

References ii

T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *KDD*, pages 785–794. ACM, 2016.

W. Chen, T.-Y. Liu, Y. Lan, Z.-M. Ma, and H. Li. Ranking measures and loss functions in learning to rank. In *Advances in Neural Information Processing Systems*, pages 315–323, 2009.

C. L. Clarke, M. Kolla, G. V. Cormack, O. Vechtomova, A. Ashkan, S. Büttcher, and I. MacKinnon. Novelty and diversity in information retrieval evaluation. In *SIGIR*, pages 659–666. ACM, 2008.

D. Cossock and T. Zhang. Subset ranking using regression. In *COLT*, volume 6, pages 605–619. Springer, 2006.

Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *Journal of machine learning research*, 4(Nov):933–969, 2003.

N. Fuhr. Optimum polynomial retrieval functions based on the probability ranking principle. *ACM Transactions on Information Systems (TOIS)*, 7(3):183–204, 1989.

Y. Ganjisaffar, R. Caruana, and C. Lopes. Bagging gradient-boosted trees for high precision, low variance ranking models. In *SIGIR*, pages 85–94. ACM, 2011.

References iii

R. Herbrich, T. Graepel, and K. Obermayer. Large margin rank boundaries for ordinal regression. 2000.

R. Jagerman, J. Kiseleva, and M. de Rijke. Modeling label ambiguity for neural list-wise learning to rank. In *Neu-IR SIGIR Workshop*, 2017.

T. Joachims. Training linear svms in linear time. In *KDD*, pages 217–226. ACM, 2006.

P. Li, Q. Wu, and C. J. Burges. Mcrank: Learning to rank using multiple classification and gradient boosting. In *Advances in neural information processing systems*, pages 897–904, 2008.

T.-Y. Liu. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval*, 3(3):225–331, 2009.

R. D. Luce. *Individual choice behavior: A theoretical analysis*. Courier Corporation, 2005.

D. Sculley. Large scale learning to rank. In *In NIPS 2009 Workshop on Advances in Ranking*, 2009.

X. Wang, C. Li, N. Golbandi, M. Bendersky, and M. Najork. The lambdaloss framework for ranking metric optimization. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 1313–1322, 2018.

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*, pages 1192–1199. ACM, 2008.

Acknowledgments

All content represents the opinion of the author(s), which is not necessarily shared or endorsed by their employers and/or sponsors.