Generative vs. Discriminative

- A *discriminative* model estimates the probability of belonging to a class directly from the observed features of the document based on the training data
- Generative models perform well with low numbers of training examples
- Discriminative models usually perform better given enough training data
  - Can also easily incorporate many features

3

3

Discriminative Models

Discriminative models typically rely on "machine learning"

There has been considerable interaction between these fields

- 60s: Rocchio algorithm (c.f. Lecture 6) is a simple learning approach, and later in 80s & 90s other learned ranking algorithms based on user feedback were proposed
- 2000s: text classification

All were limited by low amounts of training data

Web query logs have generated a wave of research

- From 2008 e.g., "**Learning to Rank**"

Neural nets have given rise to more efforts in text representation:

- C.f. Word2vec, Glove, BERT

4

4

Motivations for Learning

How to choose term weighting models?

- Term weighting models have different **assumptions** about how relevant documents should be retrieved (c.f. Lecture 8)

Also:

- **Field-based models**: term occurrences in different fields matter differently
- **Proximity-models**: close co-occurrences matter more
- **Priors**: documents with particular lengths or URL/inlink distributions matter more
- ***Query Features:*** Long queries, difficult queries, query type

How to combine all these easily and appropriately?

5

5

FEATURES FOR LEARNING

6

6

Ranking Cascades

Typically, in web-scale search, the ranking process can be seen as a series of cascades [1]

- Rank some documents
- Pass top-ranked onto next cascade for refined re-ranking

Re-Ranking: Try really hard to get the top of the ranking correct, using many signals (features) ⇨ *LEARNING TO RANK*

Simple Ranking: Identify a set most likely to contain relevant documents ⇨ *e.g. BM25*

Boolean: Do query terms occur? ⇨ *e.g. AND/OR*

[1] J. Pedersen, Query understanding at Bing, SIGIR 2010 Industry Day

Learning to Rank

Application of tailored machine learning techniques to automatically (select and) weight retrieval *features*

- Based on **training data** with relevance assessments

Learning to rank has been popularised by commercial search engines (e.g. Bing, Baidu, Yandex)

- They require large training datasets, possibly instantiated from click-through data
- **Click-through data** has facilitated the deployment of learning approaches

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

Types of Features

Typically, commercial search engines use hundreds of features for ranking documents, usually categorised as follows:

Name	Varies depending on…		Examples
	Query	Document	
Query Dependent Features	✓	✓	Weighting models, e.g. BM25, PL2 Proximity models, e.g. Markov Random Fields Field-based weighting models, e.g. PL2F Deep-learned semantic matching
Query Independent Features	✗	✓	PageRank, number of inlinks Spamminess
Query Features	✓	✗	Query length Presence of entities

NB: Unlike text classification, features in LTR do NOT include the presence of a *specific* word in the document

9

9

Learning to Rank

1. *Sample* Identification

- Apply BM25 or similar (e.g. DFR PL2) to rank documents with respect to the query (list of candidate documents)
- Hope that the sample contains *enough* relevant documents

2. Compute more *features*

- Query Dependent
- Query Independent
- Query Features

3A. Learn ranking model

- Based on training data

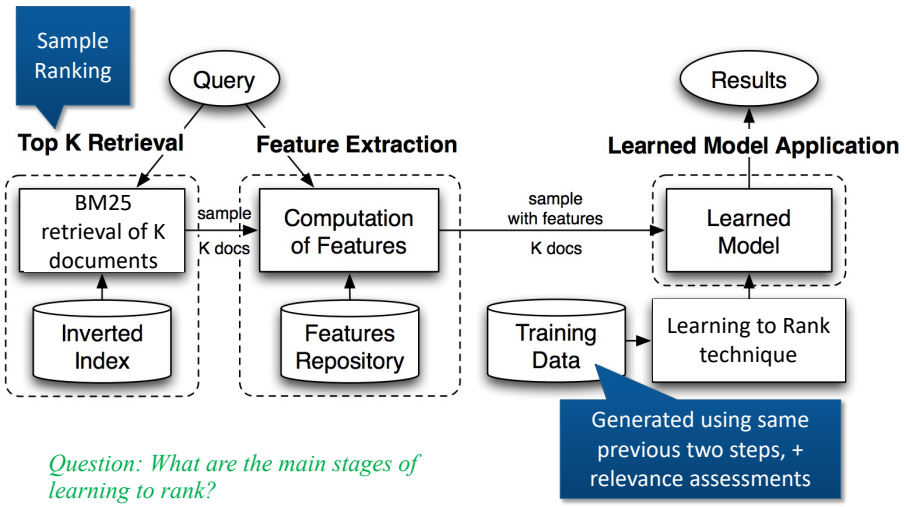
3B. Apply learned model

- Re-rank sample documents

10

10

LTR, Schematically…



Question: What are the main stages of learning to rank?

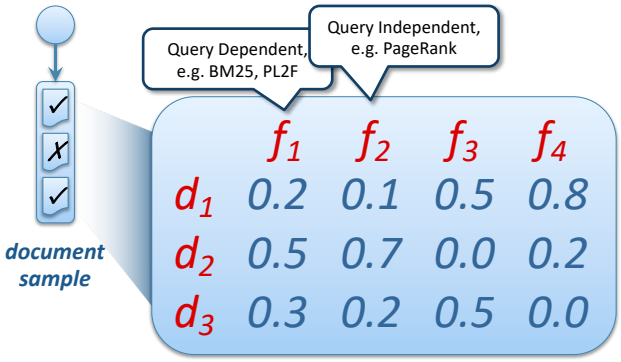
11

Features

Query dependent
- Weighting models
- … including fields
- Proximity/ngram

Query independent
- Link Analysis, e.g. PageRank Score
- URL length
- Spam Score
- Document Type (PDF, Wikipedia)

Query features
- Query length
- Presence of entities
- Predicted query performance



Question: What are the three types of features in learning to rank?

12

Feature Support in Terrier/PyTerrier

Terrier supports ranking using features, as specified in a configuration file

```
WMODEL:BM25
WMODEL:PL2
WMODEL:DirichletLM
WMODEL:PL2F
QI:StaticFeature(OIS,data.inlinks.oos.gz)
QI:StaticFeature(OIS,data.urllength.oos.gz)
DSM:org.terrier.matching.dsms.DFRDependenceScoreModifier
```

Weighting models

Field-based model

Query independent features

Proximity model

Exercise 3 has a standardised feature list, to which you will add new simple URL length and proximity features

13

13

Query-Dependent Feature Extraction

We might typically deploy a number of query dependent features

- Such as additional weighting models, e.g. fields/proximity, that are calculated based on information in the inverted index

We want the result set passed to the final cascade to have all query dependent features computed

- Once first cascade retrieval has ended, it's **too late** to compute features without **re-traversing** the inverted index postings lists
- It would take **too long** to compute all features for all scored documents

Solution: cache the postings for documents that *might* make the sample

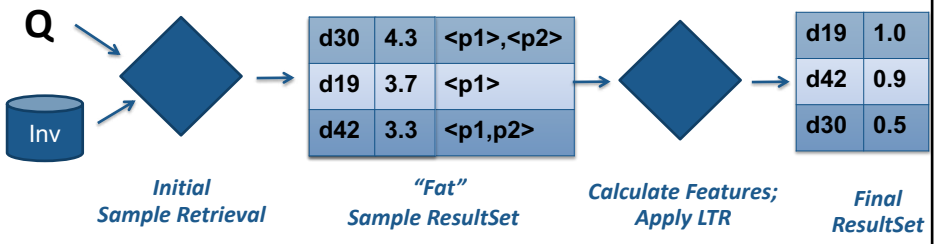
- Postings contain frequencies, positions, fields, etc.

14

14

Query-Dependent Feature Extraction

Solution: cache the postings for documents that *might* make the sample

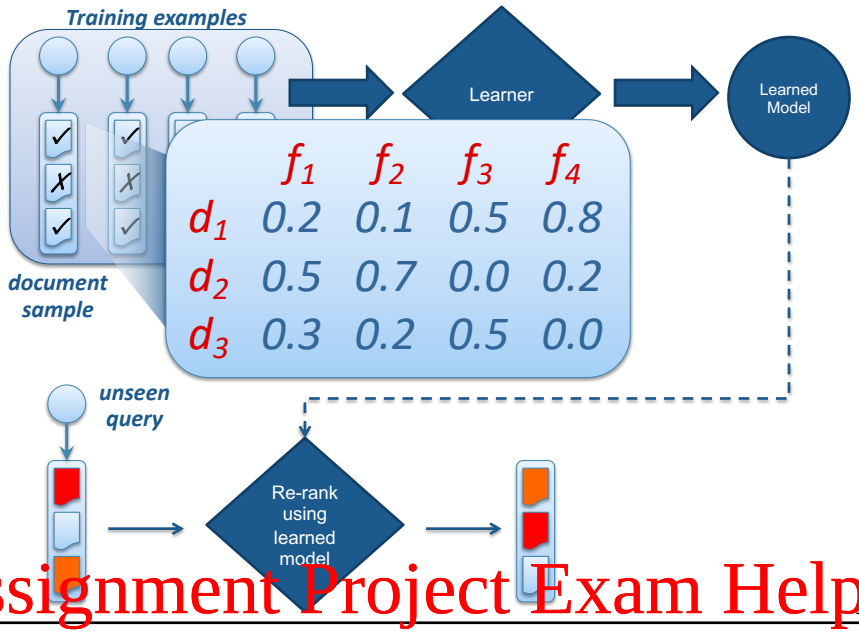


C.Macdonald et al. (2012). About Learning Models with Multiple Query-Dependent Features. TOIS 31(3).

Types and Examples of Learning to Rank Algorithms

LEARNING PROCESS

Learning to Rank Process



17

17

Training & Evaluating Learning-to-Rank

We need lots of queries to learn an effective, robust model

We cannot measure the success of a learning-to-rank technique on one of the queries it has been trained upon

- Such training performance is likely to **overestimate** its effectiveness compared to *unseen* queries

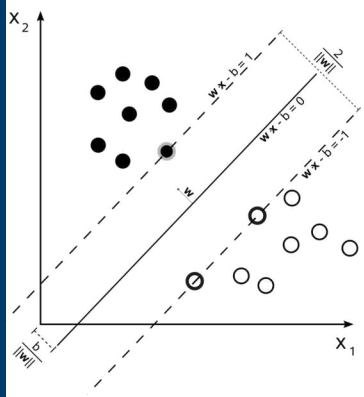
Instead, we need to separate the queries in our test collection into disjoint sets: for training the model; for *tuning* the learning parameters (the validation query set); and for testing how effective the model is for unseen queries

- We use the same processing (sample retrieval and features) for queries in each set
- (More on validation shortly)

18

18

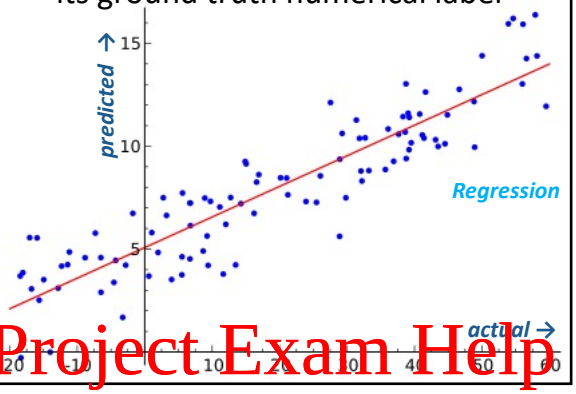
Types of Learning



Classification

Ranking is a form of supervised machine learning. Slightly different from classification, more closely related to regression:

- For each instance, aims to predict its ground truth numerical label



Regression

19

Types of Learning for Ranking Tasks (1)

A 'loss function' is used to estimate how good a particular model is (based on the training data)

- The learner will try many different models (to find the best one)

There are three types of learning appropriate for ranking tasks: pointwise, pairwise & listwise

- Each differ in the way that their loss function is computed

Pointwise approaches are closest to classification/regression

- They aim to predict the correct relevance label
  - ***Classification***: how many documents did the learner correctly predict as relevant/ non-relevant?
  - ***Regression***: how different is the predicted score from the correct relevance label for each document

OPRF: N Fuhr. (1989). Optimal Polynomial Retrieval Functions Based on the Probability Ranking Principle. TOIS. 7(3), 1989.
GBRT: S. Tyree, K.Q. Weinberger, K. Agrawal, J. Paykin: Parallel boosted regression trees for web search ranking. WWW 2011.

20

20

Types of Learning for Ranking Tasks (2)

Pointwise techniques suffer from a limitation for ranking tasks, in that they consider each document independently, instead of trying to get the relevant documents ranked ABOVE the non-relevant ones

Pairwise

- Take two documents with some preference
  - E.g. different relevance labels, 0 and 1, less and more clicks
- Try to minimise a loss function based on the ranking error between pairs of documents
- Problem? Doesn't always approximate a real evaluation measure
- Examples: RankSVM, RankNet

Listwise

- Consider the entire ranking of documents
- Encapsulates a standard IR evaluation measure within the loss function
- Not all measures are appropriate for list wise learning, e.g. P@3 vs NDCG@1000
- Examples: **AFS, LambdaMART**
- Generally more effective than Pointwise or Pairwise

RankSVM: T. Joachims. Optimizing Search Engines using Clickthrough Data. CIKM'03.
RankNet: C Burges et al. Learning to Rank using Gradient Descent. ICML'05.
AFS: D. Metzler. Automatic Feature Selection for the Markov Random Field Model for IR. CIKM'07.
LambdaMART: C. J.C Burges. From RankNet to LambdaRank to LambdaMART: An Overview. Technical Report MSR-TR-2010-82. Microsoft Research.

LINEAR MODELS FOR LTR

Types of Learned Models (1)
Linear Model

Linear Models (the most intuitive to comprehend)

- Many learning to rank techniques generate a linear combination of feature values:

$$score(d,Q) = \sum_f w_f \cdot value_f(d)$$

- Linear Models make some assumptions:
  - ***Feature Usage***: They assume that the same features are needed by all queries
  - ***Model Form***: The model is only a linear combination of feature values.
    - Contrast this with genetic algorithms, which can learn functional combinations of features, by randomly introducing operators (e.g. try divide feature *a* by feature *b*), but are unpractical to learn
- **Question**: How do we set w_f values that maximise the performance of an IR evaluation metric?

Difficulty in Training

IR evaluation measures are not continuous, and hence are not differentiable wrt a feature weight

- i.e. they are not smooth as we vary a feature weight *w*

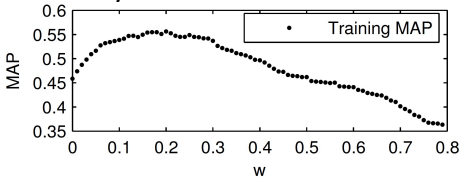
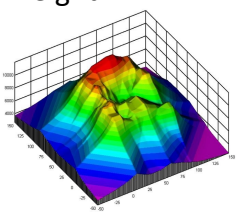


Figure 2: Tuning w, for *log* PageRank combination.

Why?



Metrics don't respond to changes in the scores of documents, but only when a swap between the ordering of two documents takes place

- We need advanced gradient descent/hill-climbing/simulated annealing
- Many of the advances in learning to rank try to address this

Q: Why are evaluation measures non-smooth wrt features weights?

Figures from: (a) Craswell et al., SIGIR 2005 (b) Hongning Wang

(Greedy) Automatic Feature Selection (AFS)

Consider a linear model:

$$score(d,Q) = \sum_f w_f \cdot value_f(d)$$

AFS: Select a $w_f > 0$ for the feature f that best improves an evaluation measure M

Step 1: Select the most effective single feature according to M

	f_1	f_2	f_3	f_4
d_1	0.2	0.1	0.5	0.8
d_2	0.5	0.7	0.0	0.2
d_3	0.3	0.2	0.5	0.0

f1	0.30
f2	0.35
f3	0.05
f4	0.10

→ *(f2, 1)*

w_1	w_2	w_3	w_4

Current Best M = 0

D. Metzler. Automatic Feature Selection for the Markov Random Field Model for IR. CIKM'07.

25

25

(Greedy) Automatic Feature Selection (AFS)

Consider a linear model:

$$score(d,Q) = \sum_f w_f \cdot value_f(d)$$

AFS: Select a $w_f > 0$ for the feature f that best improves an evaluation measure M

Step 2: Select the feature and corresponding w_f that most improves M (e.g. using sim. annealing)

	f_1	f_2	f_3	f_4
d_1	0.2	0.1	0.5	0.8
d_2	0.5	0.7	0.0	0.2
d_3	0.3	0.2	0.5	0.0

f2 + ?	x f1	
f2 + ?	x f3	
f2 + ?	x f4	

w_1	w_2	w_3	w_4
	1		

Current Best M = 0.35

D. Metzler. Automatic Feature Selection for the Markov Random Field Model for IR. CIKM'07.

26

26

(Greedy) Automatic Feature Selection (AFS)

Consider a linear model:

$$score(d,Q) = \sum_f w_f \cdot value_f(d)$$

AFS: Select a $w_f > 0$ for the feature f that best improves an evaluation measure M

Step 2: Select the feature and corresponding w_f that most improves M (e.g. using sim. annealing)

	f_1	f_2	f_3	f_4
d_1	0.2	0.1	0.5	0.8
d_2	0.5	0.7	0.0	0.2
d_3	0.3	0.2	0.5	0.0

f2 + 1.3 x f1	0.36
f2 + 0.3 x f3	0.40
f2 + 0.9 x f4	0.34

→ *(f3, 0.3)*

w_1	w_2	w_3	w_4
	1	0.3	

Current Best M = 0.40

D. Metzler. Automatic Feature Selection for the Markov Random Field Model for IR. CIKM'07.

27

27

(Greedy) Automatic Feature Selection (AFS)

Consider a linear model:

$$score(d,Q) = \sum_f w_f \cdot value_f(d)$$

AFS: Select a $w_f > 0$ for the feature f that best improves an evaluation measure M

Step 2: Select the feature and corresponding w_f that most improves M (e.g. using sim. annealing)

	f_1	f_2	f_3	f_4
d_1	0.2	0.1	0.5	0.8
d_2	0.5	0.7	0.0	0.2
d_3	0.3	0.2	0.5	0.0

f2 + 1.3 x f1	0.36
f2 + 0.3 x f3	0.40
f2 + 0.9 x f4	0.34

→ *(f3, 0.3)*

Step 3: Goto step 2, until we observe no more improvements in M, or no more features

w_1	w_2	w_3	w_4
	1	0.3	

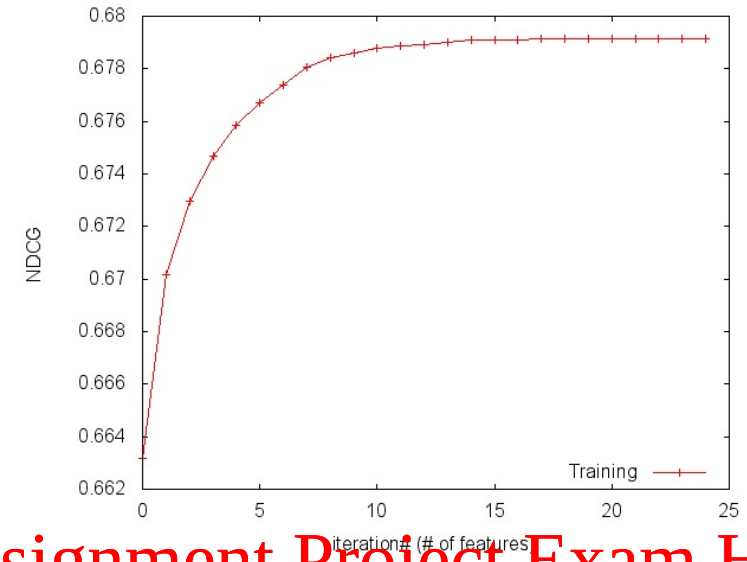
Current Best M = 0.40

D. Metzler. Automatic Feature Selection for the Markov Random Field Model for IR. CIKM'07.

28

28

Training performance, by Iteration

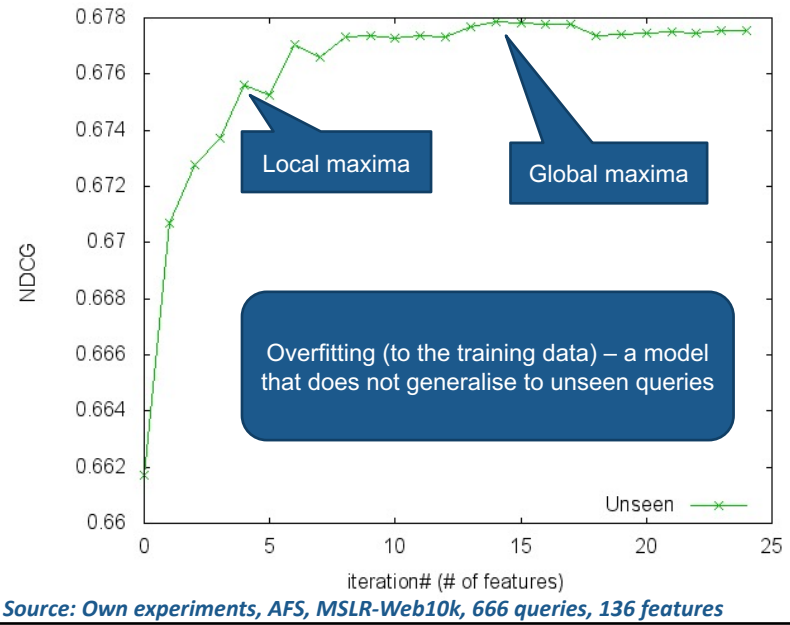


Source: Own experiments, AFS, MSLR-Web10k, 1334 queries, 136 features

29

29

Performance on Unseen Data

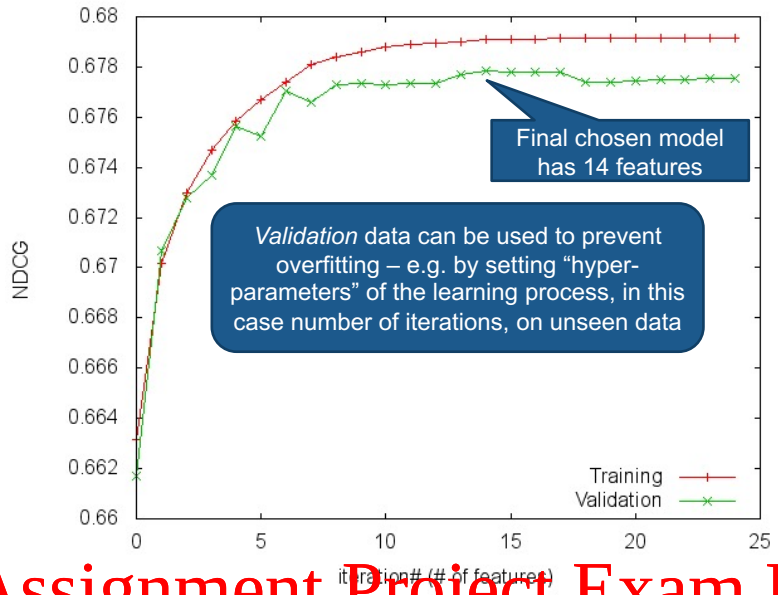


Source: Own experiments, AFS, MSLR-Web10k, 666 queries, 136 features

30

30

Solution: Use Validation Data



What is the role of validation data during learning?

31

31

Summary: Evaluating a Learning to Rank Technique

Evaluation of learning to rank (L2R) uses a test collection

- Documents & queries with known relevant documents

We split queries into three subsets:

- Training – for learning the model
- Validation – for setting hyper-parameters, e.g. number of iterations. Unseen to the learner; prevents overfitting to the training data
- Testing – for determining how effective the learner is on unseen queries

In Exercise 3, you will evaluate on the HP04 (test) topics;
We have provided separate training and validation topic sets.
c.f. PyTerrier's L2R documentation on how to learn using training & validation topics

32

32

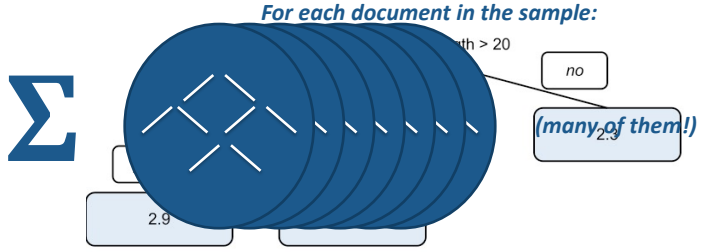
REGRESSION TREES FOR LTR

33

Type of Learned Models (2)
Regression Trees

Tree Models

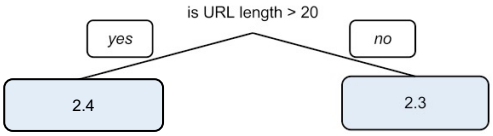
- A *regression tree* is series of decisions, leading to a partial score output
- The outcome of the learner is a "forest" of many such trees, used to calculate the final score of a document for a query
- Their ability to customise branches makes them more effective than linear models
- Regression trees are **pointwise** in nature, but several major search engines have created adapted regression tree techniques that are listwise
  - E.g. Microsoft's LambdaMART is at the heart of the Bing search engine



S. Tyree, K. Weinberger, K. Agrawal, J. Paykin. (2011). Parallel Boosted Regression Trees for Web Search Ranking. WWW'11.

34

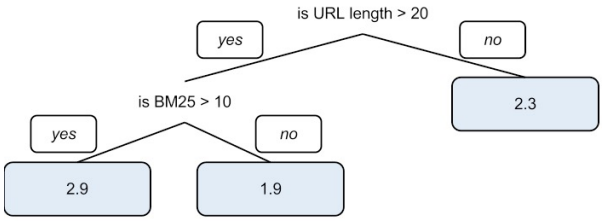
Growing a Tree



	f_1	f_2	f_3	f_4	L
d_1	0.2	0.1	0.5	0.8	1
d_2	0.5	0.7	0.0	0.2	2
d_3	0.3	0.2	0.5	0.0	0

Loss = 1.4

Q: Can we add another decision point?



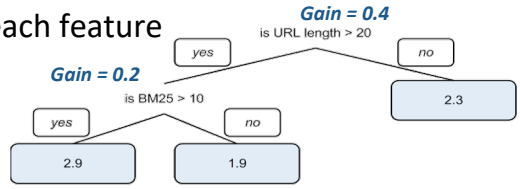
Loss = 1.2 ⟹ *Gain = 0.2*

35

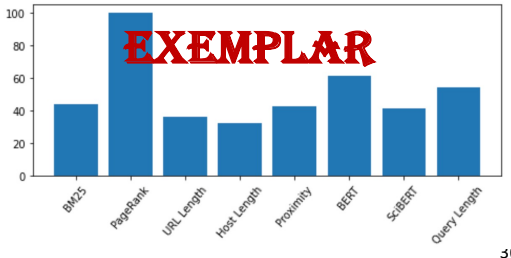
Feature Importance

Regression Trees have an in-built measure of feature importance

- By recording the "gain" at each node during training time
- Summing the gains for each feature



Feature	Norm. Total Gain
BM25	45
PageRank	100
URL Length	39
…	…



NB: sklearn's RandomForest, as well as XGBoost and LightGBM, all provide a `feature_importances_` variable

Summary of LTR Types

Some "take-aways":

- Listwise techniques are typically more effective; linear models are easy to understand; regression trees are very adaptable
  - LambdaMART enhances pointwise regression trees with a listwise measure
  - Currently one of the benchmarks to beat!
- Validation data is equally important to use for all types of supervised learning, including LTR, to prevent overfitting
  - Always use a validation set!

In Exercise 3 of your coursework, you will be experimenting with the stateof-the-art LambdaMART technique, as provided by LightGBM

37

37

Example Performances

Example performances on 8000 queries from a Microsoft learning to rank dataset

Learning to Rank Technique	Type	NDCG@10	+	-
BM25	n/a	0.2987	-	
AFS (Simulated Annealing)	Listwise: Linear	0.4263	5596	1778
Boosted Regression Trees	Pointwise: Tree	0.4240	5802	1808
LambdaMART	Listwise: Tree	**0.4373**	5760	1844

- Differences might be small in magnitude, but are statistically significant over hundreds of queries

In Exercise 3 of your coursework, you will be experimenting with the stateof-the-art LambdaMART technique, as provided by LightGBM

38

38

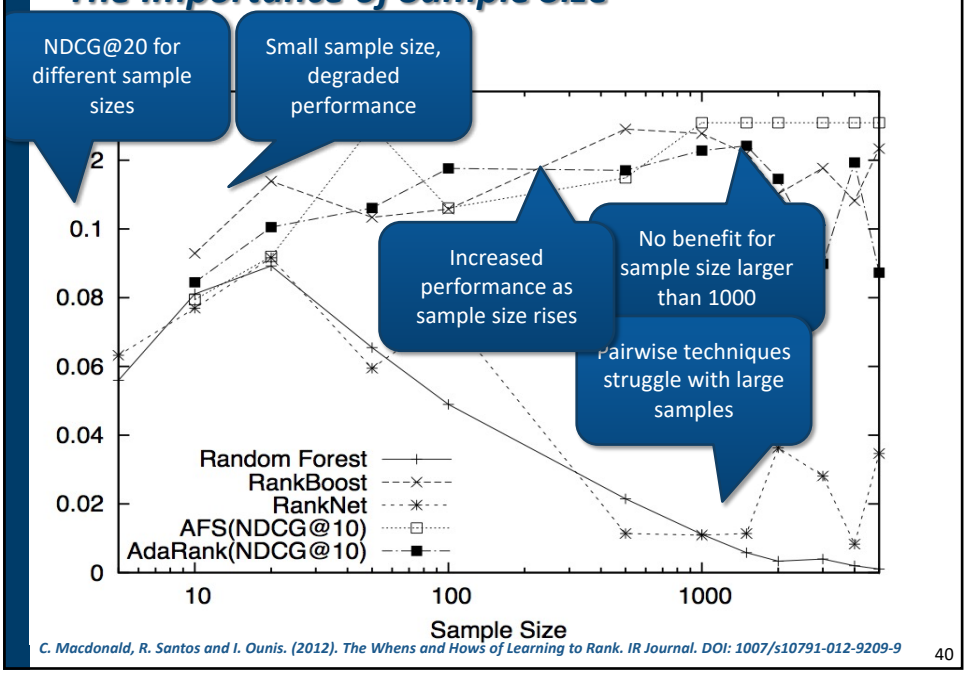
The Importance of Sample Size

What sample size is necessary for effective learning?

- Small samples: faster learning, faster retrieval
- Large samples: higher recall

(See next figure)

The Importance of Sample Size



C. Macdonald, R. Santos and I. Ounis. (2012). The Whens and Hows of Learning to Rank. IR Journal. DOI: 1007/s10791-012-9209-9

Learning to Rank Best Practices (Basics)

Q: What is the role of the sample in learning-to-rank?

About the sample – it must be:

- **Small enough** that calculating features doesn't take too long
- **Large enough** to have enough sufficient recall of relevant douments
  - Previous slide says 1000 documents needed for TREC Web corpora
- **Simple early cascade**: Only time for a single weighting model to create the sample
  - E.g. PL2 < BM25 with no features; but when features are added, no real difference

Multiple weighting models as features:

- Using more weighting models do improve the learned model

Field Models as Features

- Adding field-based models as features always improves effectiveness

In general, Query Independent features (e.g. PageRank, Inlinks, URL length) improve effectiveness

C. Macdonald, R. Santos, I. Ounis, B. He. About Learning Models with Multiple Query Dependent Features. TOIS 31(3), 2013.
C. Macdonald & I. Ounis (2012). The Whens & Hows of Learning to Rank. INRT, 16(5), 2012.

41

Learning to Rank Best Practices (Query Features)

Tree-based learning to rank techniques have an inherent advantage

- They can activate different parts of their learned model depending on feature values
- We can add "query features", which allow different sub-trees for queries with different characteristics
  - E.g. long query, apply proximity

Class of Query Features	NDCG@20
Baseline (47 document features) LambdaMART	*0.2832*
Query Performance Predictors	0.3033*
Query Topic Classification (entities)	**0.3109***
Query Concept Identification (concepts)	0.3049*
Query Log Mining (query frequency)	0.3085*

C. Macdonald, R. Santos and I. Ounis. (2012). On the Usefulness of Query Features for Learning to Rank. CIKM'12.

42

Summary

Generative models such as LM and DFR are increasingly replaced by discriminative models

- However, such generative models have a role as strong features, or for sample identification
- Web search nowadays relies on discriminative learning-to-rank techniques to perform fast, effective rankings (see next lectures)

Generally speaking, listwise approaches are the most effective family of learner type

- Techniques based on regression trees (e.g. Boosted Regression Trees, LambdaMART) are considered to be the state-of-the-art [at least in Web search!]

There are increasingly more tools to deploy such algorithms

- LightGBM, Xgboost, Ranklib, TF-Ranking

To some extent being replaced by Neural IR…

43

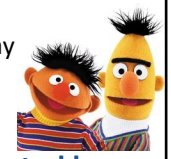
What's the latest? Neural Net LMs

LTR uses *hand-engineered* features (BM25, PL2F, PageRank etc)

- More research is now focused on learning the ranking model from bare text. This is achievable because of advances in **neural network architectures**

1. Word2vec – shallow neural model that creates a dense representation of words.

- Each word can be represented by an *embedding* vector of say numbers; similar words have similar vectors
- Trained to predict a word given its *context* words

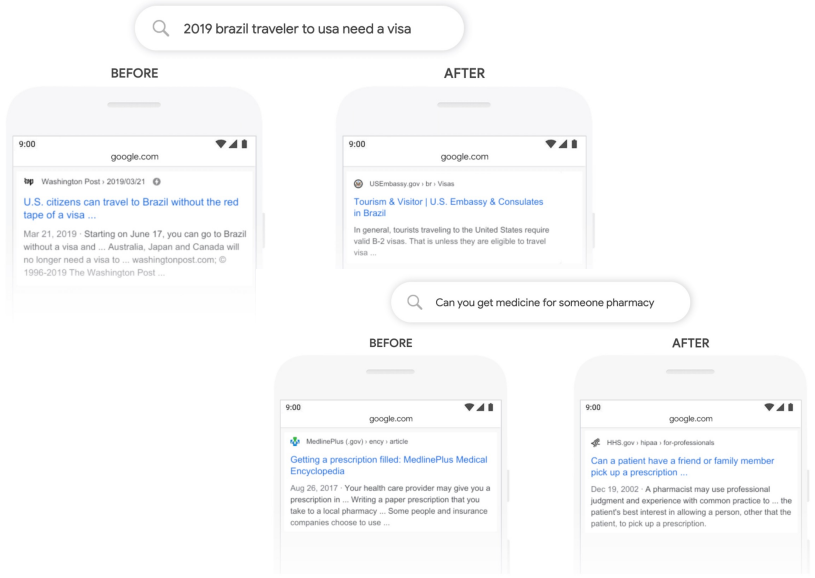


2. BERT – A deep attention-based embedding space invented by Google aka Neural Network Language Models (NNLM)

- Uses attention while training the neural network to predict the *next sentence*. Attention helps identify which contextual words affect the meaning of a given word
- Difficult to train, but can be *fine-tuned* to adapt to a new task
- Has proven to be **very effective** at ranking, but **slow** (400 paragraphs/sec on a "commodity GPU" card

44

BERT in Action



Google reports that BERT is quite effective for long queries

45

45

Contrast: LTR vs NNLM

LTR dataset:

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 Label | Query ID | Hand-Engineer Features Values (QD/QI)

NNLM learning dataset

1 188714 1000052 foods and supplements to lower blood sugar

Watch portion sizes: Even healthy foods will cause high blood sugar if you eat too Make sure each of your meals has the

Relevance Label | Query ID | Query Text | Document text

Neural ranking learns representation for queries and documents, and how to measure their similarity, directly from the labels

- Needs MUCH more data than LTR – e.g. MSMARCO 300k training queries

Neural models are expensive – *thus far* they have been typically used as re-rankers, e.g. BERT features in a LTR approach.

More on Neural IR in Lecture 15

46

46

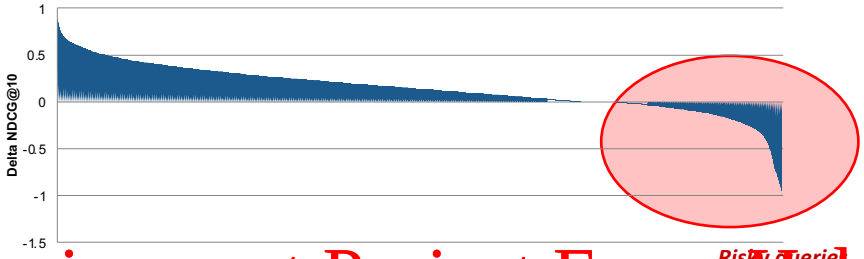
Final thoughts/Recent Research: Risk-sensitive LTR

Learning to rank improves most queries over the sample model (e.g. BM25), but not all

- Risk occurs when a LTR system underperforms compared to the sample model for a given query

BM25: Mean NDCG@10 = 0.2987
L'MART: Mean NDCG@10 = 0.4373

Improvement in applying LambdaMART (n=8000 queries)



47

47

Avoiding Risk

Why is risk bad?

Search engines want to satisfy users for every single query issued to the system

- Unsatisfied users might switch engine, leading to loss in trust and consequently ad revenue, etc

Hence, a search engine aims to avoid any abject failures - i.e. queries where performances decrease viz. of a minimum standard:

- Learning to rank should not hurt performance, e.g. > BM25, PL2, etc

48

48

Risk-sensitive LTR

We have been working on this problem: how can we make LambdaMART more risk-averse during learning

- i.e. create models that identify poor performing queries and learn to resort to more stable models (e.g. BM25)

We do this by emphasising reductions in effectiveness compared to BM25 during learning

- So trees that are risky are not learned

Let *ΔM* be the change in a metric for query *j* for a given tree

$$\Delta M' = \begin{cases} \Delta M & \text{if } M_m(j) + \Delta M \geq M_b(j); \\ \Delta M \cdot (1+\alpha) & \text{otherwise.} \end{cases}$$

- So, here we emphasise *ΔM* by *α* if the tree is poor (*α*>1)

Let us know if this kind of research sounds interesting!

D.T. Dinger, C Macdonald and I Ounis: Hypothesis Testing for Risk-Sensitive Evaluation of Retrieval Systems, SIGIR'14

Acknowledgements

A couple of slides are based on:

- Bruce Croft et al. – see book: "Search Engines: Information Retrieval in Practice"
- Hongning Wang's presentation

The following people gave input to the creation of these slides:

- Iadh Ounis
- Richard McCreadie