

Efficiency/Effectiveness Trade-offs in Learning to Rank

Tutorial @ ECML PKDD 2018

<http://learningtorank.isti.cnr.it>

Claudio Lucchese

Ca' Foscari University of Venice

Venice, Italy

Franco Maria Nardini

HPC Lab, ISTI-CNR

Pisa, Italy

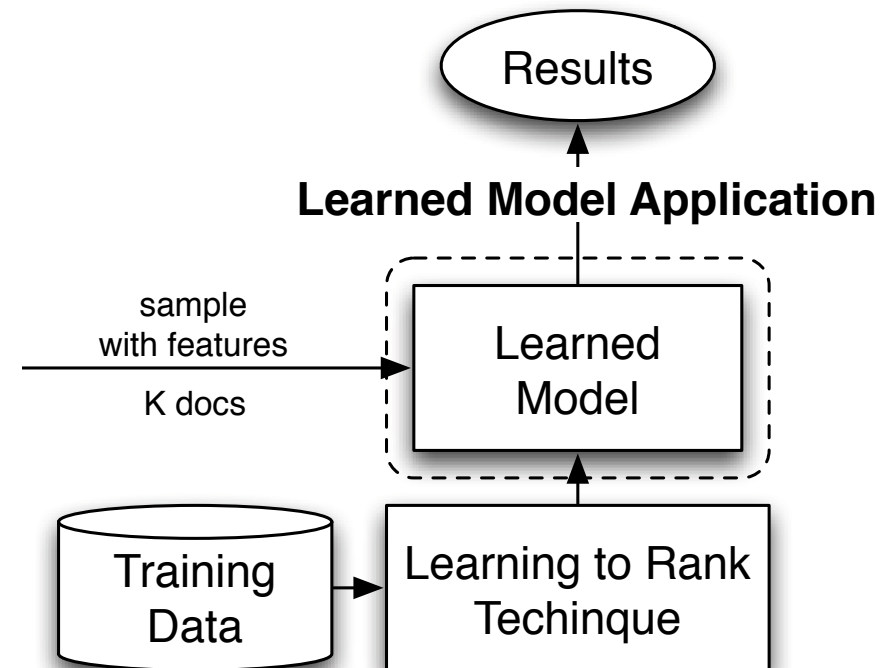


Two-stage (or more) Ranking Architecture



Efficiency/Effectiveness Trade-offs

- Efficiency in Learning to Rank (LtR) has been addressed in different ways
- Main research lines
 - Feature selection
 - Optimizing efficiency within the learning process
 - Approximate score computation and efficient cascades
 - Efficient traversal of tree-based models
- Different impact on the architecture



Feature Selection

Feature Selection

- Feature selection techniques allows to **reduce redundant features**
 - Redundant features are useless both at training and scoring time
- Filtering out the irrelevant features to enhance the **generalization performance** of the learned model
- Identifying key features also helps to **reverse engineer** the predictive model and to **interpret** the results obtained
- A reduced set of highly discriminative and non redundant features results in a **reduced feature extraction cost** and in a **faster learning and classification/prediction/ranking**

Feature Selection Methods

- Feature selection for ranking inherits methods from classification
- Classification of feature selection methods [GE03]
 - **Filter** methods: feature selection is defined as a preprocessing step and can be independent from learning
 - **Wrapper** methods: utilizes a learning system as a black box to score subsets of features
 - **Embedded** methods: perform feature selection within the training process
- Wrapper or embedded methods: higher computational cost / algorithm dependent
 - not suitable for a LtR scenario involving hundreds of continuous or categorical features
- Focus on **filter** methods
 - Allow for a fast pre-processing of the dataset
 - Totally independent from the learning process

[GE03] Isabelle Guyon and Andre Elisseeff. *An introduction to variable and feature selection*. The Journal of Machine Learning Research, 3:1157–1182, 2003.

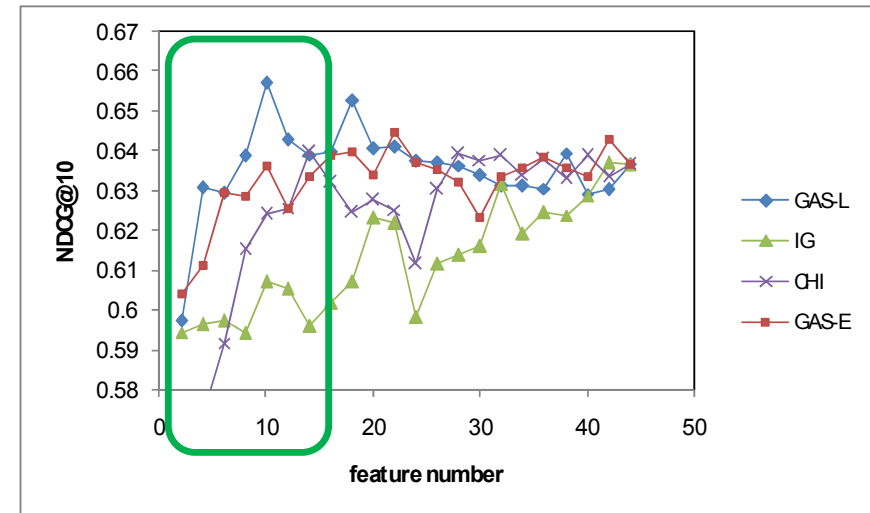
GAS [GLQL07]

- Geng *et al.* are the first proposing feature selection methods for ranking
- Authors propose to exploit ranking information for selecting features
 - They use IR metrics to measure the **importance** of each feature
 - MAP, NDCG: rank instances by feature, evaluate and take the result as importance score
 - They use **similarities** between features to avoid selecting redundant ones
 - By using ranking results of each feature: Kendall's tau, averaged over all queries
- Feature selection as a **multi-objective optimization problem**: maximum importance and minimum similarity
- Greedy Search Algorithm (GAS) performs feature selection **iteratively**
 - Update phase needs the tuning of an hyper-parameter c weighting the impact of the update

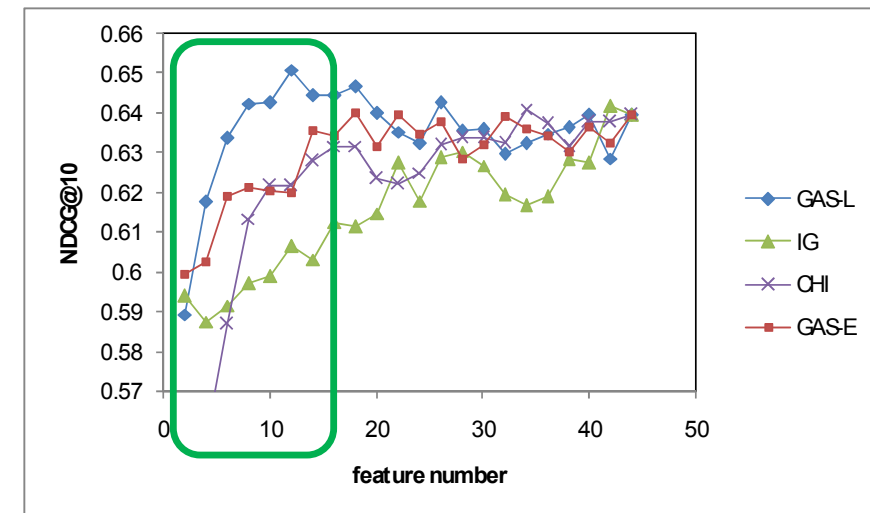
[GLQL07] X. Geng, T. Liu, T. Qin, and H. Li. *Feature selection for ranking*. In Proc. ACM SIGIR, 2007.

GAS [GLQL07]

- Experiments
 - .gov and TREC 2004 Web Track
 - BM25 as first stage
 - 44 features per doc
- Evaluation Measures
 - MAP
 - NDCG
- Applied to second stage ranker
 - Ranking SVM
 - RankNet



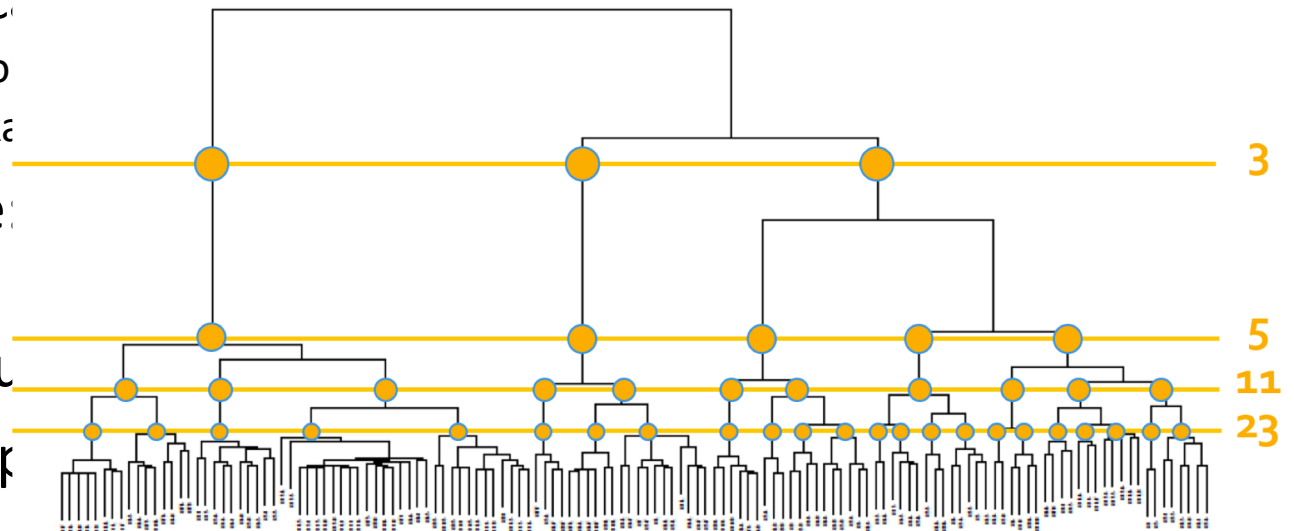
(b) NDCG@10 of Ranking SVM



(b) NDCG@10 of RankNet

Fast Feature Selection for LtR [GLNP16]

- Lucchese *et al.* propose three novel filter methods providing flexible and model-free feature selection
 - Two parameter-free variations of GAS: NGAS and XGAS
 - HCAS exploits hierarchical structure
 - Only one feature per group
 - Two variants: Single-linkage and Complete-linkage
- Importance of a feature: sum of importance of all features in the subtree rooted at the feature
- Similarity between features: distance between features in the hierarchy
- No need to tune hyper-parameters



Fast Feature Selection for LtR

- Experiments
 - MSLR-Web10K (Fold1) and Yahoo LETOR
 - By varying the subset sampled
 - Results confirms Geng *et al.* [GLQL07]
- Evaluation Measures
 - NDCG@10
- For small subsets (5%, 10%, 20%):
 - Best performance by HCAS with “Single Linkage”.
 - Statistically significant w.r.t. GAS
 - Performance against the full model

MSN-1					
Subset	NMI	AGV	S	K	LM-1
5%	0.3548	0.3340	0.3280	0.3313	0.4304
10%	0.3742	0.3416	0.3401	0.3439	0.4310
20%	0.4240	0.3776	0.3526	0.3533	0.4330
30%	0.4625	0.3798	0.4312	0.3556	0.4386
40%	0.4627	0.3850	0.4330	0.3788	0.4513
Full	0.4863	0.4863	0.4863	0.4863	0.4863

MSN-1					
Subset %	NGAS	XGAS p = 0.05	HCAS “single”	HCAS “ward”	GAS c = 0.01
5%	0.4011▼	0.4376▲	0.4423▲	0.4289	0.4294
10%	0.4459	0.4528	0.4643▲	0.4434▼	0.4515
20%	0.4710	0.4577▼	0.4870▲	0.4820	0.4758
30%	0.4739▼	0.4825	0.4854	0.4879	0.4848
40%	0.4813	0.4834	0.4848	0.4853	0.4863
Full	0.4863	0.4863	0.4863	0.4863	0.4863

Further Reading

- Pan *et al.* use boosted regression trees to investigate greedy and randomized wrapper methods [PCA+09].
- Dang and Croft propose a wrapper method that uses best first search and coordinate ascent to greedily partition a set of features into subsets to be selected [DC10].
- Hua *et al.* propose a feature selection method based on clustering: k -means is first used to aggregate similar features, then the most relevant feature in each cluster is chosen to form the final set [HZL+10].
- Laporte *et al.* [LFC+12] and Lai *et al.* [LPTY13] use embedded methods for selecting features and building the ranking model at the same step, by solving a convex optimization problem.
- Naini and Altingovde use greedy diversification methods to solve the feature selection problem [NA14].
- Xu *et al.* solve the feature selection task by modifying the gradient boosting algorithm used to learn forests of regression trees [XHW+14].

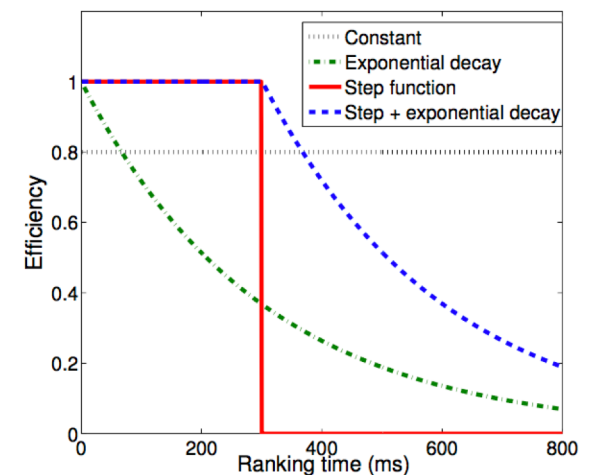
Optimizing Efficiency within the Learning Process

Learning to Efficiently Rank [WLM10]

- Wang *et al.* propose a new cost function for **learning** models that directly optimize the tradeoff metrics: Efficiency-Effectiveness Tradeoff Metric (EET)

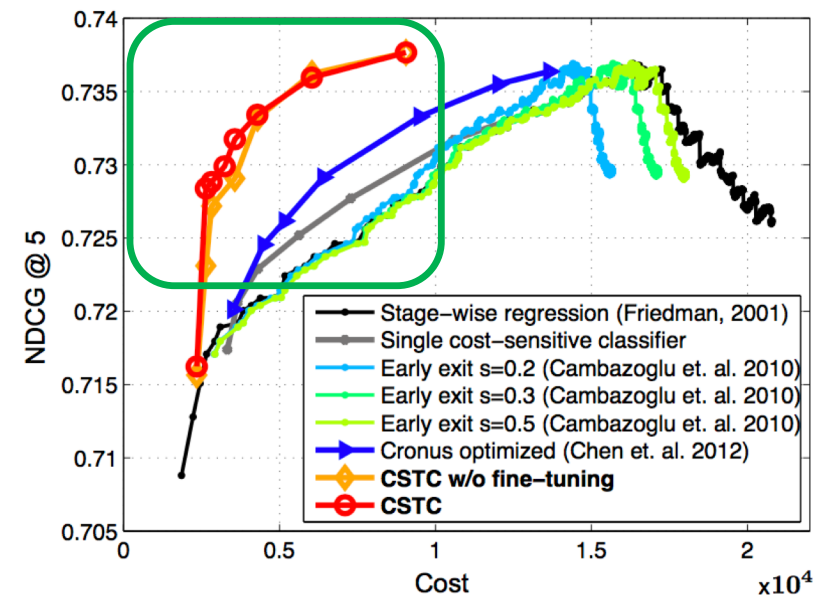
$$\text{EET}(Q) = \frac{(1 + \beta^2) \cdot (\gamma(Q)) \sigma(Q)}{\beta^2 \cdot \sigma(Q) + \gamma(Q)} \longrightarrow \text{MEET}(R) = \frac{1}{N} \sum \text{EET}(Q)$$

- New efficiency metrics: constant, step, exponential
- Focus on linear feature-based ranking functions
- Learned functions show **significant decreased average query execution times**



Cost-Sensitive Tree of Classifiers [XKWC13].

- Xu *et al.* observe that the test-time cost of a classifier is often dominated by the **computation required for feature extraction**
- Tree of classifiers: each path extract different features and is optimized for a specific sub-partition of the data
 - Input-dependent feature selection
 - Dynamic allocation of time budgets: higher budgets for more complex partitions
- Experiments
 - Yahoo LETOR dataset
 - Quality vs Cost budget
- Comparisons against [CZC+10]



Training Efficient Tree-Based Models for Document Ranking [AL13]

- Asadi and Lin propose techniques for training GBRTs that have efficient runtime characteristics.
 - **compact**, **shallow**, and **balanced** trees yield faster predictions
- Cost-sensitive Tree Induction: jointly minimize the loss and the evaluation cost
- Two strategies
 - By directly modifying the node splitting criterion during tree induction
 - Allow split with maximum gain if it does not increase the maximum depth of the tree
 - Find a node closer to the root which, if split, result in a gain larger than the discounted maximum gain
 - Pruning while boosting with focus on **tree depth** and **density**
 - Additional stages compensate for the loss in effectiveness
 - Collapse terminal nodes until the number of internal nodes reach a balanced tree
- Experiments on MSLR-WEB10K show that the **pruning** approach is superior.
 - 40% decrease in prediction latency with minimal reduction in final NDCG.

CLEAVER [LNO+16a]

- Lucchese et al. propose a **pruning & re-weighting** post-processing methodology
- Several pruning strategies
 - random, last, skip, low weights
 - score loss
 - quality loss
- Greedy line search strategy applied to tree weights
- Experiments on MART and LambdaMART
 - MSLR-Web30K and Istella-S LETOR

C. Lucchese, F. M. Nardini, S. Orlando, R. Perego, F. Silvestri, and S. Trani. *Post-learning optimization of tree ensembles for efficient ranking*. In Proc. ACM SIGIR, 2016.

CLEAVER

MART on MSN-1

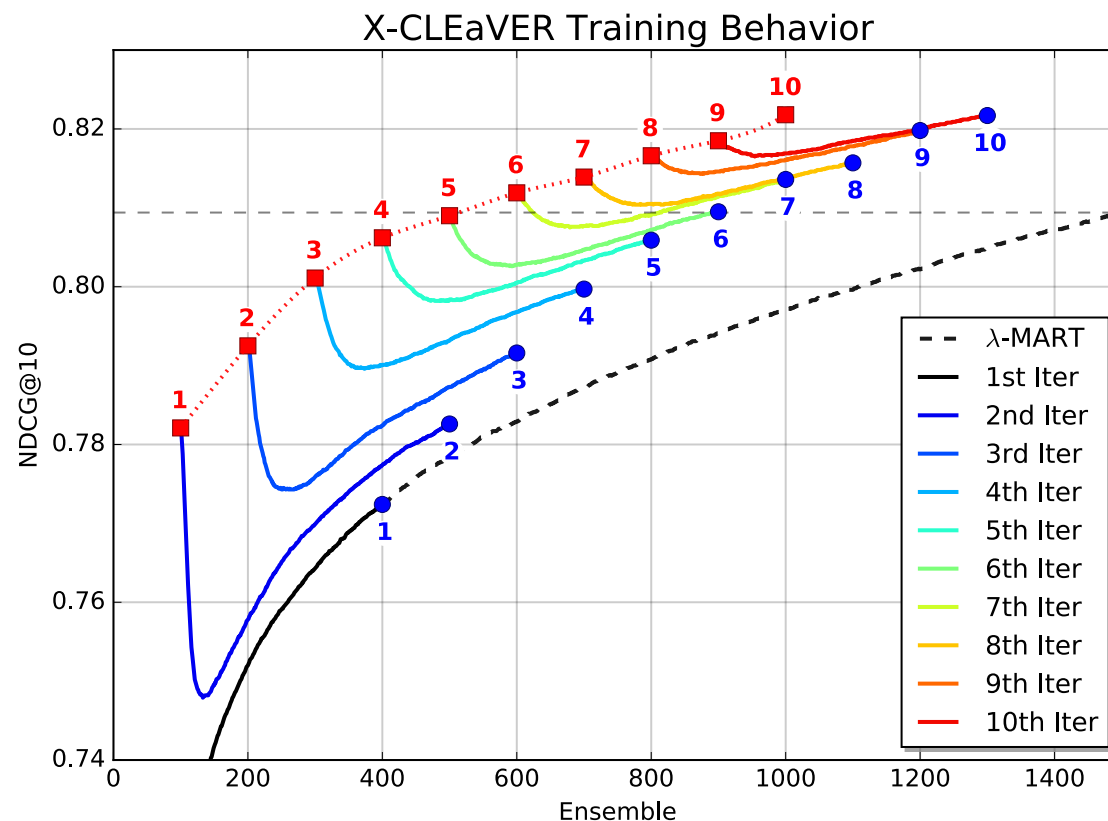
Strategy	100 Trees				500 Trees				737 Trees			
	Trees	Time	Speed-up	<i>NDCG@10</i>	Trees	Time	Speed-up	<i>NDCG@10</i>	Trees	Time	Speed-up	<i>NDCG@10</i>
<i>Reference</i>	100	5.55	–	0.4590	500	19.36	–	0.4749	737	27.46	–	0.4766
RANDOM	40	3.28	1.7x	0.4603	350	14.27	1.4x	0.4756	516	19.64	1.4x	0.4768
LAST	60	3.93	1.4x	0.4601	400	16.17	1.2x	0.4755	590	22.76	1.2x	0.4771
SKIP	30	2.84	2.0x	0.4593	300	12.98	1.5x	0.4749	442	17.25	1.6x	0.4766
LOW-WEIGHTS	40	3.26	1.7x	0.4609	400	15.75	1.2x	0.4753	663	25.28	1.1x	0.4779
QUALITY-LOSS	30	2.89	1.9x	0.4618	150	7.35	2.6x	0.4752	369	14.42	1.9x	0.4771
SCORE-LOSS	50	3.50	1.6x	0.4591	250	11.16	1.7x	0.4751	442	17.47	1.6x	0.4766

λ -MART on Istella

Strategy	100 Trees				500 Trees				736 Trees			
	Trees	Time	Speed-up	<i>NDCG@10</i>	Trees	Time	Speed-up	<i>NDCG@10</i>	Trees	Time	Speed-up	<i>NDCG@10</i>
<i>Reference</i>	100	5.37	–	0.6923	500	15.74	–	0.7397	736	20.40	–	0.7432
RANDOM	30	2.65	2.0x	0.7003	250	9.10	1.7x	0.7424	515	14.81	1.4x	0.7449
LAST	70	4.22	1.3x	0.6969	400	13.55	1.2x	0.7418	442	14.09	1.4x	0.7437
SKIP	20	2.36	2.3x	0.6976	250	9.09	1.7x	0.7416	368	11.42	1.8x	0.7438
LOW-WEIGHTS	30	2.85	1.9x	0.6986	350	11.07	1.4x	0.7418	589	15.55	1.3x	0.7437
QUALITY-LOSS	20	2.29	2.3x	0.6989	200	7.83	2.0x	0.7412	442	13.22	1.5x	0.7438
SCORE-LOSS	20	2.13	2.5x	0.6976	300	10.68	1.5x	0.7407	368	12.39	1.6x	0.7433

X-CLEAVER [LNO+18]

- Pruning and
 - First, reduce the ensemble size
 - Then the desired ratio
- Same pruning
- Experiment:
 - Pruning at single position
 - X-CLEaVER time.



mizing the

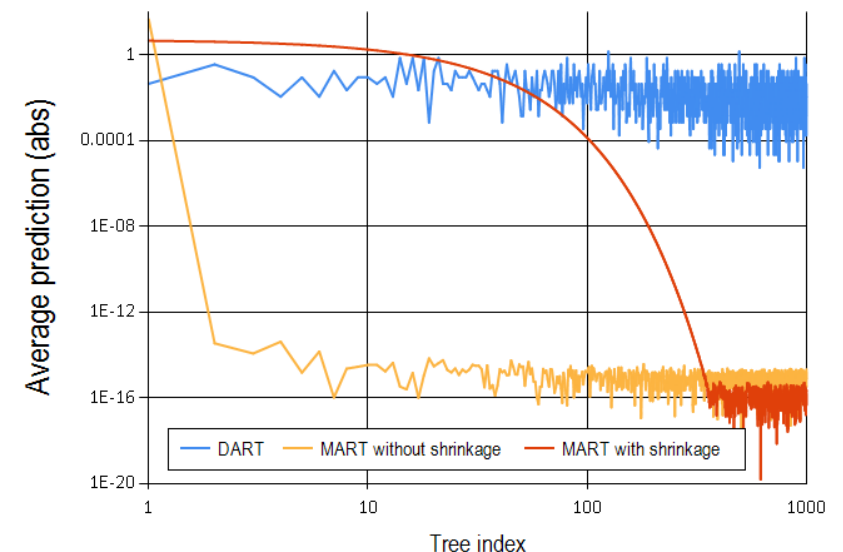
-1, Istella-S
an applying a

cient at scoring

[LNO+18] C. Lucchese, F. M. Nardini, S. Orlando, R. Perego, F. Silvestri, and S. Trani. *X-CLEAVER: Learning Ranking Ensembles by Growing and Pruning Trees*. ACM TIST, 2018.

DART [VGB15]

- Rashmi and Gilad-Bachrach propose to employ *dropouts* from NN while learning a MART: DART
 - Dropouts as a way to fight *over-specialization*
 - Shrinkage helps but does not solve
- DART differs from MART
 - When learning a new tree, a subset of the model is muted (random)
 - Normalization step when adding a new tree to avoid *overshooting*
 - muted trees are still part of the model
 - new tree scaled by a factor of $1/k$
 - $k/(k+1)$ normalization of new and muted



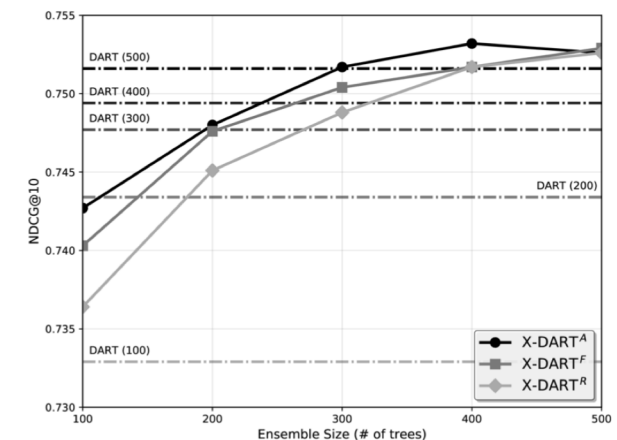
DART

- Experiments on ranking/regression/classification tasks
 - LambdaMART
 - Ranking on the MSLR-Web10K dataset
 - NDCG@3

algorithm	Shrinkage	Dropout	Loss function parameter	Feature fraction	NDCG@3
MART	0.4	0	1.2	0.75	46.31
DART	1	0.03	1.2	0.5	46.70

X-DART [LNO+17]

- Lucchese *et al.* merge DART with pruning while training
 - like DART, some trees are muted and this set is actually removed after fitting
- Two good news
 - X-DART builds even more compact models than DART
 - Smaller models are less prone to overfitting: potential for higher effectiveness
- Three strategies for pruning
 - Ratio, Fixed, Adaptive
- Experiments on MSLR-Web30K and Istella-S
 - X-DART (adaptive) provide statistically significant improvements w.r.t. DART

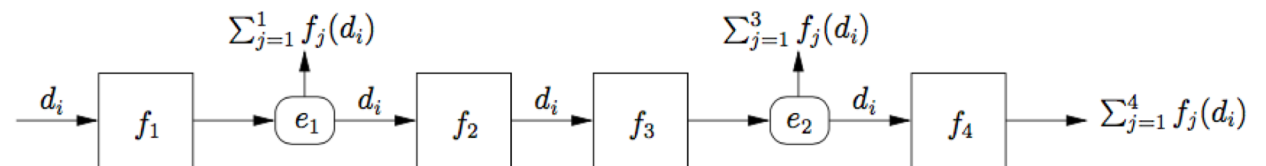


[LNO+17] C. Lucchese, F. M. Nardini, S. Orlando, R. Perego, and S. Trani. *X-DART: Blending dropout and pruning for efficient learning to rank*. In Proc. ACM SIGIR, 2017.

Approximate Score Computation and Efficient Cascades

Early Exit Optimizations for Additive Machine Learned Ranking Systems [CZC+10]

- Why short-circuiting the scoring process in additive ensembles
 - For each query, few highly relevant documents and many irrelevant ones
 - most users view only the first few result pages
- Cambazoglu *et al.* introduce additive examples with early exits



- Four techniques
 - Early exits using {Score, Capacity, Rank, Proximity} thresholds
- Evaluation on a state-of-the-art ML platform with GBRT
- With EPT, up to four times faster without loss in quality

[CZC+10] B. B. Cambazoglu, H. Zaragoza, O. Chapelle, J. Chen, C. Liao, Z. Zheng, and J. Degenhardt. *Early exit optimizations for additive machine learned ranking systems*. In Proc. ACM WSDM, 2010.

Efficient Cost-Aware Cascade Ranking in Multi-Stage Retrieval [CGBC17b]

- Cascade ranking model as a sequence of LtR models (*stages*)
 - ascending order of model complexity, only a fraction of documents in each stage will advance to the next stage
- Chen *et al.* revisit the problem on how best to balance feature importance and feature costs in multi-stage cascade ranking models
 - three cost-aware heuristics to assign features to each stage
 - cost-aware L_1 regularization to learn each stage
 - Automatic feature selection while jointly optimize efficiency and effectiveness
- Experiments
 - Yahoo! Learning to Rank, gov
- Comparisons against [WLM11b]

[CGBC17b] R. Chen, L. Gallagher, R. Blanco, and J. S. Culpepper. *Efficient cost-aware cascade ranking in multi-stage retrieval*. In Proc. ACM SIGIR, 2017.

Efficient Cost-Aware Cascade Ranking in Multi-Stage Retrieval

System	ERR@k			NDCG@k			P@k			Cost
	@5	@10	@20	@5	@10	@20	@5	@10	@20	
Ground Truth Models										
GBDT-BL	0.4605	0.4751	0.4789	0.7448	0.7872	0.8279	0.8323	0.7577	0.5967	15988
GBRT-BL	0.4598	0.4744	0.4782	0.7420	0.7852	0.8264	0.8322	0.7562	0.5962	15876
LambdaMART-BL	0.4526	0.4674	0.4712	0.7314	0.7768	0.8203	0.8330	0.7564	0.5964	15856
Cascade Models (including Baseline) ^a										
WLM-BL	0.3679	0.3876	0.3933	0.5886	0.6506	0.7088	0.7832	0.7171	0.5673	99
LM-C3-C	0.3950	0.4127	0.4175	0.6461	0.7067	0.7638	0.8086**	0.7364**	0.5856**	1871
LM-C3-E	0.3871	0.4039	0.4089	0.6503	0.7033	0.7618	0.8192**	0.7413**	0.5885**	1580
LM-C3-F	0.3876	0.4047	0.4093	0.6541	0.7113	0.7666	0.8226**	0.7483**	0.5915**	5278
GBDT-C3-C	0.4191	0.4357	0.4405	0.6535	0.7100	0.7631	0.7878*	0.7245**	0.5781**	1760
GBDT-C3-E	0.4264	0.4419	0.4466	0.6721	0.7180	0.7703	0.7942**	0.7241**	0.5778**	1535
GBDT-C3-F	0.4178	0.4350	0.4395	0.6554	0.7163	0.7672	0.7866	0.7310**	0.5819**	4953
GBRT-C3-C	0.4025	0.4203	0.4254	0.6304	0.6931	0.7488	0.7743	0.7168	0.5737**	1760
GBRT-C3-E	0.4100	0.4260	0.4313	0.6380	0.6867	0.7431	0.7697	0.7009	0.5637**	1535
GBRT-C3-F	0.4158	0.4332	0.4378	0.6479	0.7094	0.7612	0.7862	0.7294**	0.5802**	4949
LambdaMART-C3-C	0.4163	0.4332	0.4379	0.6577	0.7145	0.7673	0.7994**	0.7328**	0.5820**	1760
LambdaMART-C3-E	0.4183	0.4346	0.4394	0.6629	0.7133	0.7671	0.7968**	0.7268**	0.5786**	1535
LambdaMART-C3-F	0.4353	0.4513	0.4557	0.6847	0.7354	0.7851	0.8060**	0.7379**	0.5847**	4929

Further Reading

- Wang *et al.* [WLM11b] propose a cascade ranking model for efficient ranked retrieval
 - Retrieval as a multi-stage progressive refinement problem, where each stage considers successively richer and more complex ranking models, but over successively smaller candidate document sets
 - Boosting algorithm (modified AdaRank) to jointly learn the model structure and the set of documents to prune at each stage
 - Experiments show the model is able to simultaneously achieve high effectiveness and fast retrieval
- Xu *et al.* [XKW+14a] propose to post-process classifiers to reduce their test time complexity
 - Focus on execution time and feature extraction cost with skewed classes
 - Reduction of the average cost of a classifier during test time by an order of magnitude on real-world Web search ranking data sets

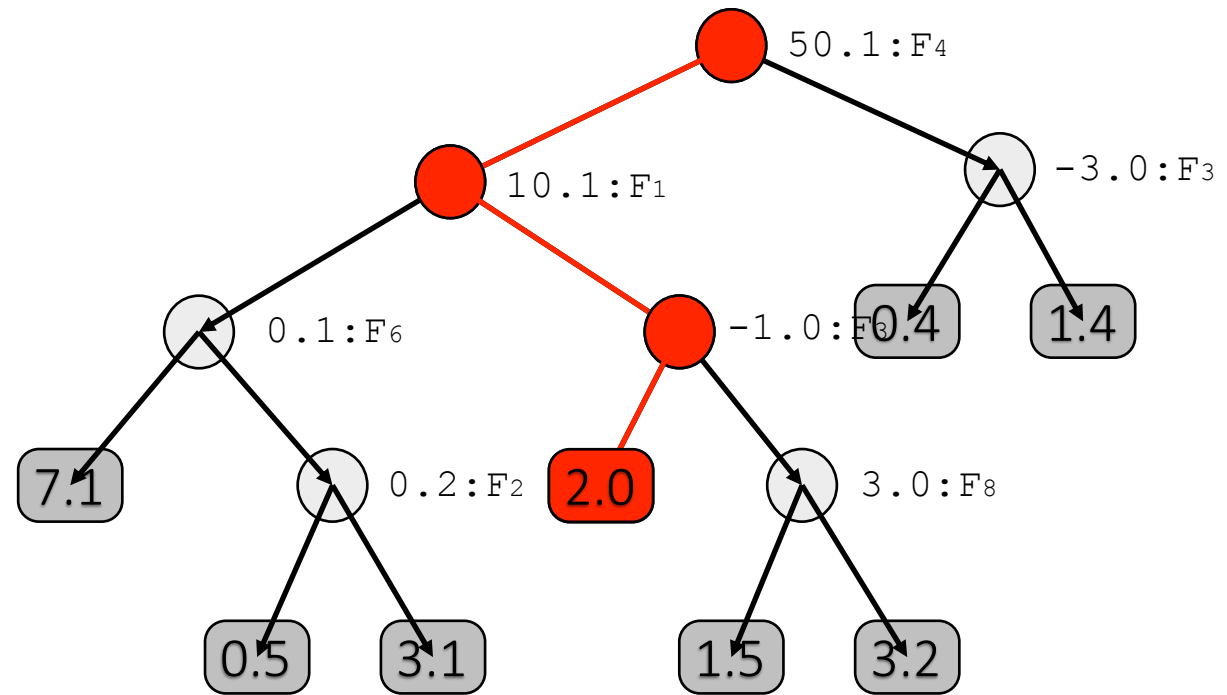
[WLM11b] L. Wang, J. Lin, and D. Metzler. *A cascade ranking model for efficient ranked retrieval*. In Proc. ACM SIGIR, 2011.

[XKW+14a] Z. Xu, M. J. Kusner, K. Q. Weinberger, M. Chen, and O. Chapelle. *Classifier cascades and trees for minimizing feature evaluation cost*. JMLR, 2014.

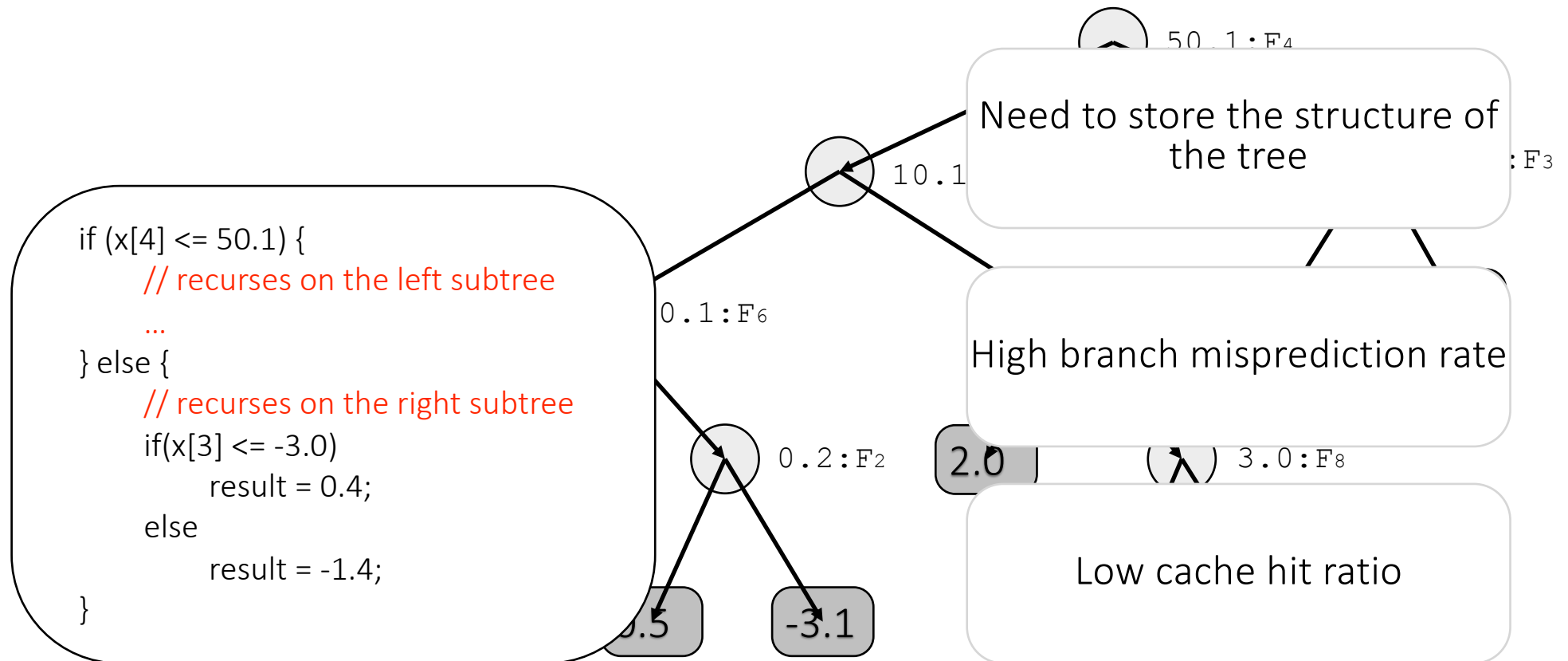
Efficient Traversal of Tree-based Models

Efficient Traversal of Tree-based models

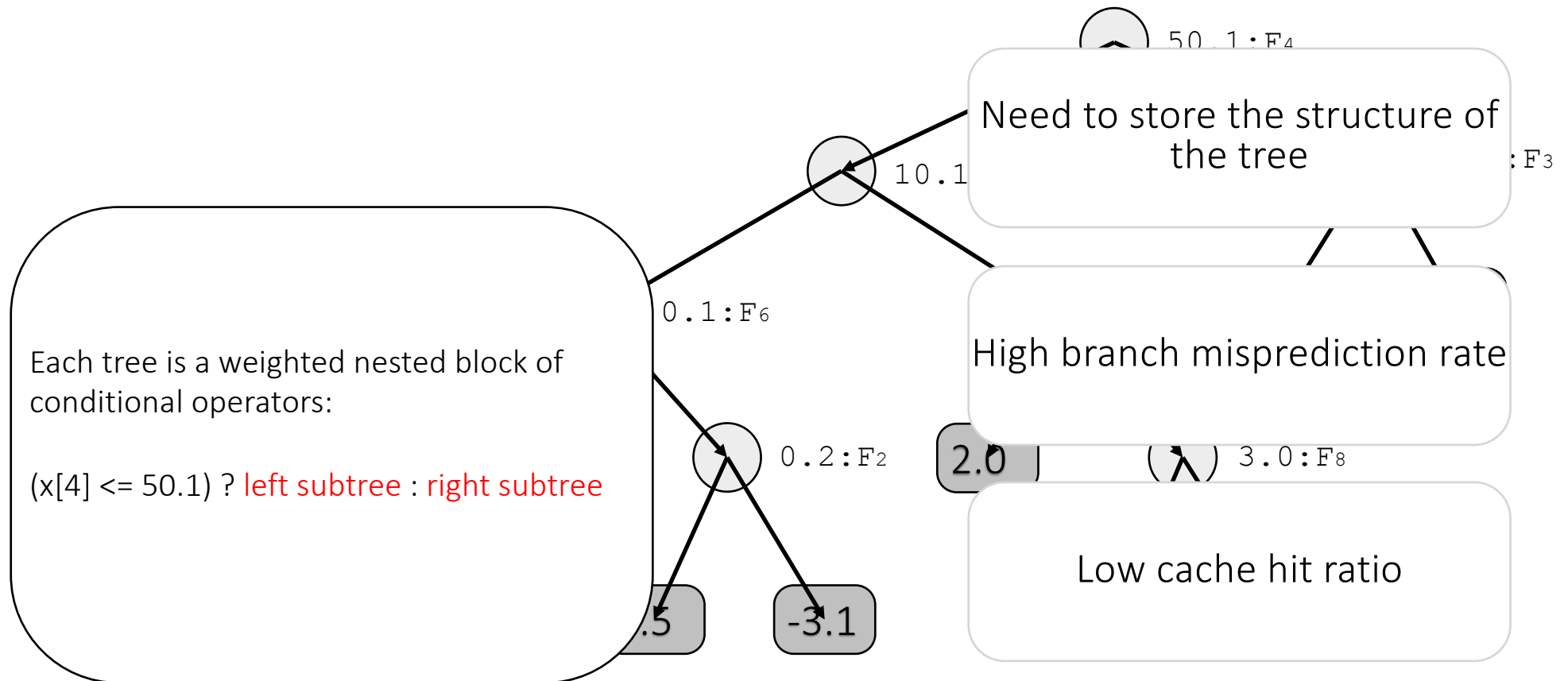
- From Yahoo! Learning to Rank Challenge Overview: *“The winner proposal used a linear combination of 12 ranking models, 8 of which were LambdaMART boosted tree models, having each up to 3,000 trees”* [YLTRC].



If-Then-Else



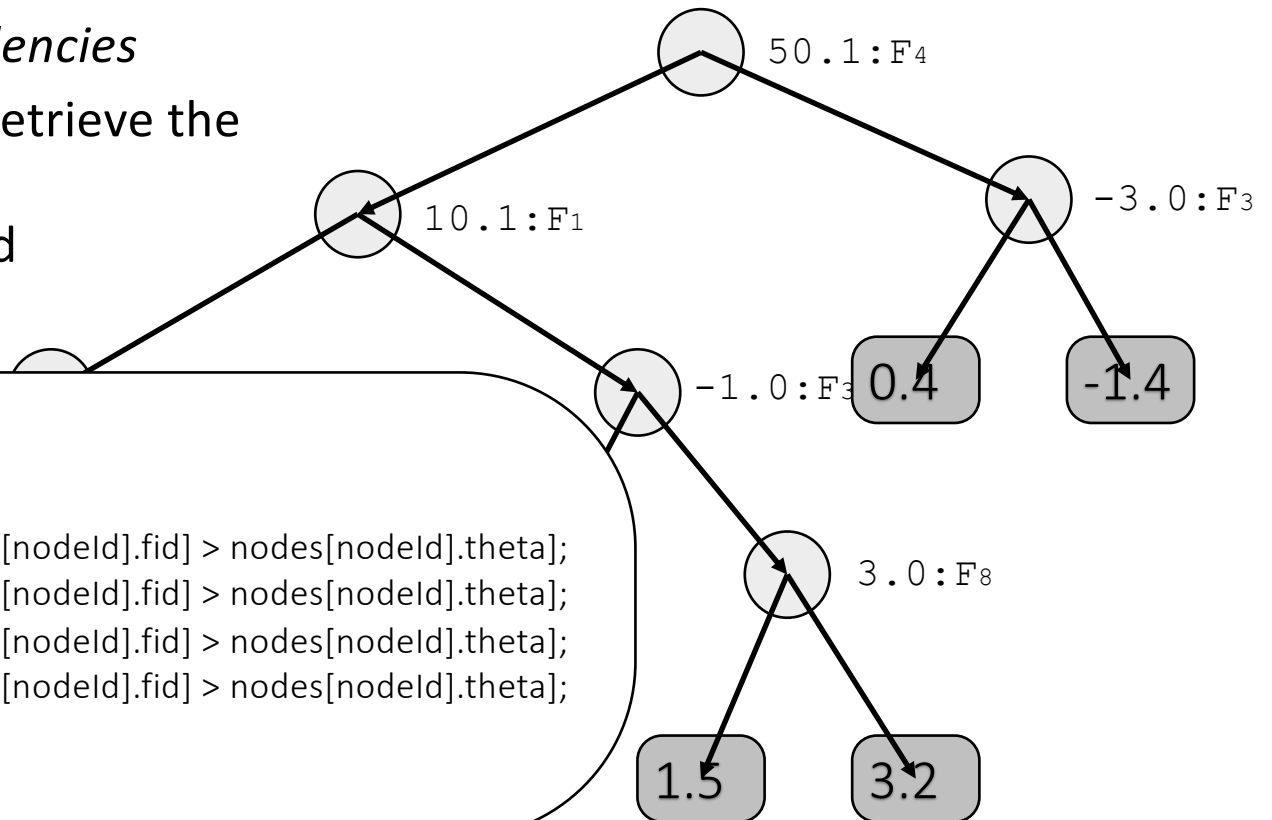
Conditional Operators



VPred [ALdV14]

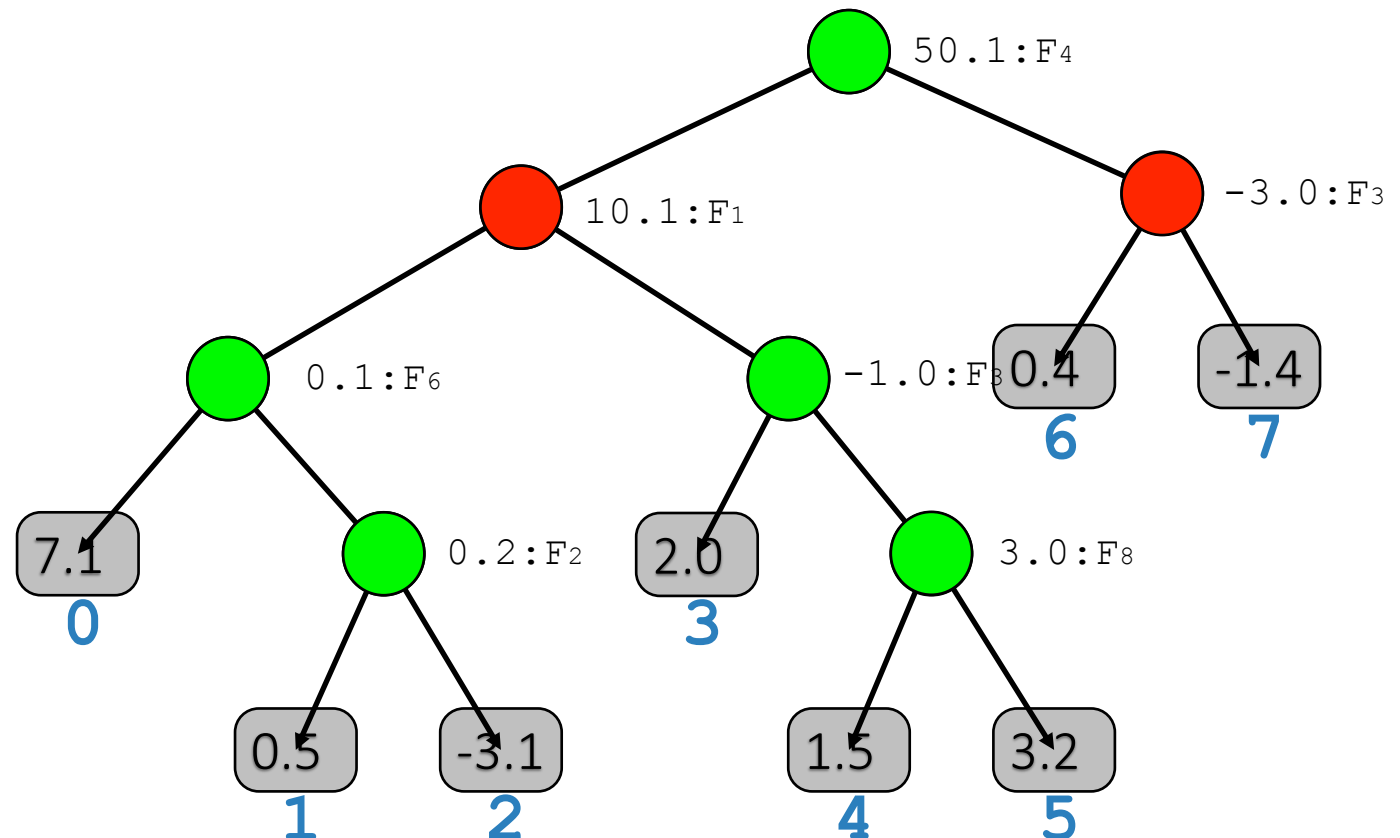
- From *control* to *data dependencies*
- Output of a test as index to retrieve the next node to process
- The visit is statically un-rolled
- 16 docs at the same time

```
double depth4(float* x, Node* nodes) {  
    int nodeId = 0;  
    nodeId = nodes->children[x[nodes[nodeId].fid] > nodes[nodeId].theta];  
    nodeId = nodes->children[x[nodes[nodeId].fid] > nodes[nodeId].theta];  
    nodeId = nodes->children[x[nodes[nodeId].fid] > nodes[nodeId].theta];  
    nodeId = nodes->children[x[nodes[nodeId].fid] > nodes[nodeId].theta];  
    return scores[nodeId];  
}
```



QuickScorer [LNO+15]

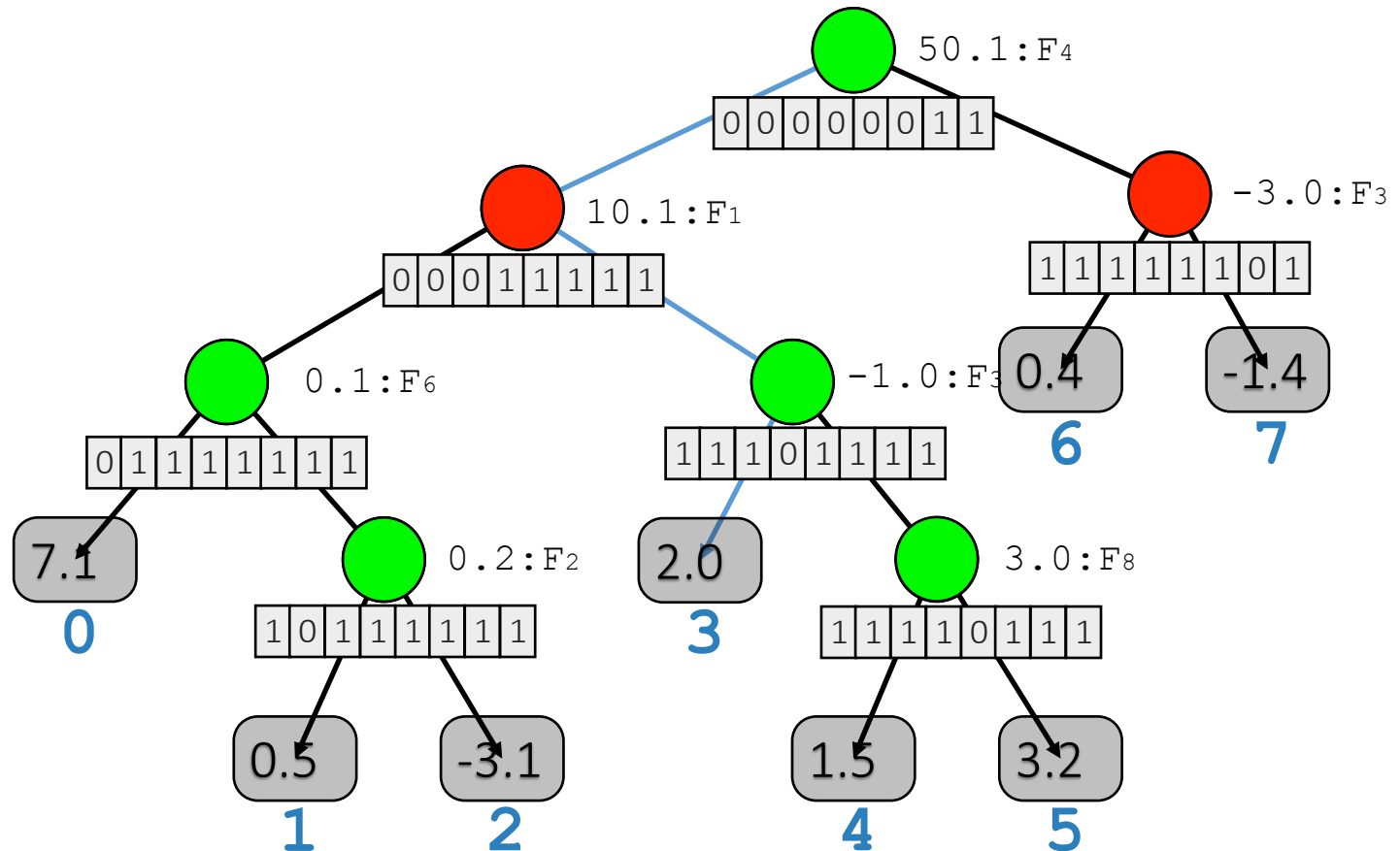
- Given a document, each node of a tree can be classified as **True** or **False**
- The exit leaf can be identified by knowing **all (and only) false nodes of a tree**
- From per-tree scoring to per-feature scoring
 - Per-feature linear scan of thresholds in the forest



[LNO+15] C. Lucchese, F. M. Nardini, S. Orlando, R. Perego, N. Tonellotto, and R. Venturini. *Quickscorer: A fast algorithm to rank documents with additive ensembles of regression trees*. In Proc. ACM SIGIR, 2015.

QuickScorer [LNO+15]

- Bitmasks storing leafs “disappearing” if the node is False.
- ANDing masks of false nodes lead to the identification of the exit leaf.
 - Leftmost bit set to 1 in the resulting mask.
- Few operations insensitive to node processing order.



QuickScorer [LNO+15]

- Public datasets: MSLR-Web10K and Yahoo LETOR
- Experiments on LambdaMART models: 1K, 5K, 10K, 20K trees and 8, 16, 32, 64 leaves.
 - trained with QuickRank [QR].

Method	Λ	Number of trees/dataset							
		1,000		5,000		10,000		20,000	
		MSN-1	Y!S1	MSN-1	Y!S1	MSN-1	Y!S1	MSN-1	Y!S1
QS	8	2.2 (-)	4.3 (-)	10.5 (-)	14.3 (-)	20.0 (-)	25.4 (-)	40.5 (-)	48.1 (-)
VPRED		7.9 (3.6x)	8.5 (2.0x)	40.2 (3.8x)	41.6 (2.9x)	80.5 (4.0x)	82.7 (3.3)	161.4 (4.0x)	164.8 (3.4x)
IF-THEN-ELSE		8.2 (3.7x)	10.3 (2.4x)	81.0 (7.7x)	85.8 (6.0x)	185.1 (9.3x)	185.8 (7.3x)	709.0 (17.5x)	772.2 (16.0x)
STRUCT+		21.2 (9.6x)	23.1 (5.4x)	107.7 (10.3x)	112.6 (7.9x)	373.7 (18.7x)	390.8 (15.4x)	1150.4 (28.4x)	1141.6 (23.7x)
QS	16	2.9 (-)	6.1 (-)	16.2 (-)	22.2 (-)	32.4 (-)	41.2 (-)	67.8 (-)	81.0 (-)
VPRED		16.0 (5.5x)	16.5 (2.7x)	82.4 (5.0x)	82.8 (3.7x)	165.5 (5.1x)	165.2 (4.0x)	336.4 (4.9x)	336.1 (4.1x)
IF-THEN-ELSE		18.0 (6.2x)	21.8 (3.6x)	126.9 (7.8x)	130.0 (5.8x)	617.8 (19.0x)	406.6 (9.9x)	1767.3 (26.0x)	1711.4 (21.1x)
STRUCT+		42.6 (14.7x)	41.0 (6.7x)	424.3 (26.2x)	403.9 (18.2x)	1218.6 (37.6x)	1191.3 (28.9x)	2590.8 (38.2x)	2621.2 (32.4x)
QS	32	5.2 (-)	9.7 (-)	27.1 (-)	34.3 (-)	59.6 (-)	70.3 (-)	155.8 (-)	160.1 (-)
VPRED		31.9 (6.1x)	31.6 (3.2x)	165.2 (6.0x)	162.2 (4.7x)	343.4 (5.7x)	336.6 (4.8x)	711.9 (4.5x)	694.8 (4.3x)
IF-THEN-ELSE		34.5 (6.6x)	36.2 (3.7x)	300.9 (11.1x)	277.7 (8.0x)	1396.8 (23.4x)	1389.8 (19.8x)	3179.4 (20.4x)	3105.2 (19.4x)
STRUCT+		69.1 (13.3x)	67.4 (6.9x)	928.6 (34.2x)	834.6 (24.3x)	1806.7 (30.3x)	1774.3 (25.2x)	4610.8 (29.6x)	4332.3 (27.0x)
QS	64	9.5 (-)	15.1 (-)	56.3 (-)	66.9 (-)	157.5 (-)	159.4 (-)	425.1 (-)	343.7 (-)
VPRED		62.2 (6.5x)	57.6 (3.8x)	355.2 (6.3x)	334.9 (5.0x)	734.4 (4.7x)	706.8 (4.4x)	1309.7 (3.0x)	1420.7 (4.1x)
IF-THEN-ELSE		55.9 (5.9x)	55.1 (3.6x)	933.1 (16.6x)	935.3 (14.0x)	2496.5 (15.9x)	2428.6 (15.2x)	4662.0 (11.0x)	4809.6 (14.0x)
STRUCT+		109.8 (11.6x)	116.8 (7.7x)	1661.7 (29.5x)	1554.6 (23.2x)	3040.7 (19.3x)	2937.3 (18.4x)	5437.0 (12.8x)	5456.4 (15.9x)

Vectorized QuickScorer [LNO+16b]

- Extends QuickScorer by exploiting SIMD capabilities of modern CPUs (SSE 4.2 and AVX 2).
- V-QuickScorer (vQS) exploits 128 bit registers (SSE 4.2) and 256 bit registers (AVX 2) to:
 - perform mask computation: with 32 leaves models, 4 docs in parallel (SSE 4.2), 8 docs in parallel (AVX 2).
 - perform score computation: with 32 leaves models, 2 docs in parallel (SSE 4.2), 4 docs in parallel (AVX 2).
- Tests on MSN10K, Yahoo LETOR and istella datasets.
- Experiments on LambdaMART models: 1K, 10K trees and 32, 64 leaves.
 - trained with QuickRank [QR].

Λ	Method	Number of trees/dataset					
		1,000			10,000		
		MSN-1	Y!S1	istella	MSN-1	Y!S1	istella
32	QS	6.3 (-)	12.5 (-)	8.9 (-)	73.7 (-)	88.7 (-)	69.9 (-)
	vQS (SSE 4.2)	3.2 (2.0x)	5.2 (2.4x)	4.2 (2.1x)	46.2 (1.6x)	53.7 (1.7x)	38.6 (1.8x)
	vQS (AVX-2)	2.6 (2.4x)	3.9 (3.2x)	3.1 (2.9x)	39.6 (1.9x)	43.7 (2.0x)	30.7 (2.3x)
64	QS	11.9 (-)	18.8 (-)	14.3 (-)	183.7 (-)	182.7 (-)	162.2 (-)
	vQS (SSE 4.2)	10.2 (1.2x)	13.9 (1.4x)	11.0 (1.3x)	173.1 (1.1x)	164.3 (1.1x)	132.2 (1.2x)
	vQS (AVX-2)	7.9 (1.5x)	10.5 (1.8x)	8.0 (1.8x)	138.2 (1.3x)	140.0 (1.3x)	104.2 (1.6x)

[LNO+16b] C. Lucchese, F. M. Nardini, S. Orlando, R. Perego, N. Tonellotto, and R. Venturini. *Exploiting CPU SIMD extensions to speed-up document scoring with tree ensembles*. . In Proc. ACM SIGIR, 2016.



Multi-thread QuickScorer [LLN+18]

- Extends QuickScorer by exploiting thread-level parallelism
 - OpenMP to distribute vQS among several processing cores
- Speedup
 - 20.7x – 35x over QS
 - 6.3x – 14x over vQS

Method	Λ	Number of trees/Dataset											
		1,000						5,000					
		MSN-1		Y!S1		Istella		MSN-1		Y!S1		Istella	
QS	32	7.0	(-)	12.4	(-)	8.9	(-)	33.7	(-)	43.8	(-)	34.5	(-)
vQS		2.8	(2.5x)	3.9	(3.2x)	3.1	(2.9x)	17.4	(1.9x)	20.8	(2.1x)	14.3	(2.4x)
vQS-MT		0.2	(35.0) [14.0x]	0.4	(31.0) [9.8x]	0.3	(29.6x) [10.3x]	1.4	(24.1x) [12.4x]	1.9	(23.1x) [10.9x]	1.2	(28.8x) [11.9x]
QS	64	12.4	(-)	19.2	(-)	13.5	(-)	70.7	(-)	83.3	(-)	69.8	(-)
vQS		8.3	(1.5x)	10.4	(1.8x)	7.9	(1.7x)	60.8	(1.2x)	64.6	(1.3x)	46.3	(1.5x)
vQS-MT		0.7	(17.7x) [11.9x]	1.0	(19.2x) [10.4x]	0.7	(19.3x) [11.3x]	4.9	(14.4x) [12.4x]	10.2	(8.2x) [6.3x]	3.7	(18.9x) [12.5x]

Method	Λ	Number of trees/Dataset											
		10,000						20,000					
		MSN-1		Y!S1		Istella		MSN-1		Y!S1		Istella	
QS	32	74.6	(-)	88.7	(-)	71.4	(-)	183.7	(-)	185.1	(-)	157.2	(-)
vQS		39.6	(1.9x)	44.2	(2.0x)	31.1	(2.3x)	87.8	(2.1x)	88.5	(2.1x)	64.8	(2.4x)
vQS-MT		3.2	(23.3x) [11.5x]	4.1	(21.6x) [10.8x]	2.5	(28.6x) [12.4x]	7.3	(25.2x) [12.0x]	8.2	(22.6x) [10.8x]	7.6	(20.7x) [8.5x]
QS	64	194.8	(-)	186.9	(-)	167.4	(-)	470.5	(-)	377.2	(-)	326.1	(-)
vQS		146.9	(1.3x)	136.8	(1.4x)	105.9	(1.6x)	321.7	(1.5x)	274.1	(1.4x)	236.6	(1.4x)
vQS-MT		12.5	(15.6x) [11.8x]	14.6	(12.8x) [9.4x]	8.8	(19.0x) [12.0x]	46.2	(10.2x) [7.0x]	35.1	(10.7x) [7.8x]	26.1	(12.5x) [9.1x]

[LLN+18] F. Lettich, C. Lucchese, F. M. Nardini, S. Orlando, R. Perego, N. Tonellotto, and R. Venturini. *Parallel Traversal of Large Ensembles of Decision Trees*. IEEE TPDS, 2018.



GPU-QuickScorer [LLN+18]

- QuickScorer on GPU
- The GPU performs both
 - mask computation
 - score computation
- Speedup
 - Up to 102.6x over QS
- NVIDIA GTX 1080
- Several low-level optimizations on GPU to achieve the best speedup

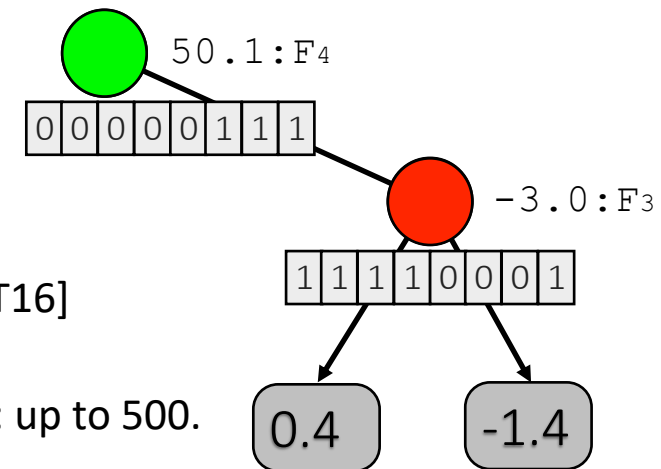
$ \mathcal{T} $	Λ	Method	Threads per block	τ	Time	
MSN-1						
1,000	32	QS	–	–	7.0	(–)
		QS _{GPU}	128	1,000	0.19	(36.8x)
	64	QS	–	–	12.4	(–)
		QS _{GPU}	256	1,000	0.25	(49.6x)
5,000	32	QS	–	–	33.7	(–)
		QS _{GPU}	512	5,000	0.44	(76.6x)
	64	QS	–	–	70.7	(–)
		QS _{GPU}	256	1,500	1.08	(65.5x)
10,000	32	QS	–	–	74.6	(–)
		QS _{GPU}	512	5,000	0.86	(86.7x)
	64	QS	–	–	194.8	(–)
		QS _{GPU}	256	1,500	2.29	(85.1x)
20,000	32	QS	–	–	183.7	(–)
		QS _{GPU}	512	4,000	1.79	(102.6x)
	64	QS	–	–	470.5	(–)
		QS _{GPU}	256	1,500	4.67	(100.8x)

[LLN+18] F. Lettich, C. Lucchese, F. M. Nardini, S. Orlando, R. Perego, N. Tonellotto, and R. Venturini. *Parallel Traversal of Large Ensembles of Decision Trees*. IEEE TPDS, 2018.

RapidScorer [YZZ+18]



- The data structure of QS and V-QS
 - linearly depends from the number of leaves of the trees (bitmasks)
 - for large trees, is redundant and inefficient
 - for large trees, impacts the performance (cache misses)
 - bad performance of QS and V-QS with a large number of leaves [JYT16]
- RapidScorer (RS)
 - industrial scenarios can exploit trees with a large number of leaves: up to 500.
- Two solutions:
 - *Epitome*: 0's count not 1's while ANDing.
 - Four bytes to store 0's (position byte first zero, byte, position byte last zero, byte)
 - Collapse nodes of trees with same feature, threshold
- Experiments on two datasets: MSN and AdsCTR (not public)
 - Against QS and V-QS
 - Speedup of RS on QS ranging from 5.8x to 25x when using trees with more than 64 leaves.



[YZZ+18] T. Ye, H. Zhou, W. Y. Zou, B. Gao, R. Zhang. *RapidScorer: Fast Tree Ensemble Evaluation by Maximizing Compactness in Data Level Parallelization*. In. Proc. ACM SIGKDD, 2018.

Further Reading

- Tang *et al.* [TJY14] investigates data traversal methods for fast score calculation with large ensembles of regression trees. Authors propose a 2D blocking scheme for better cache utilization.
 - Introduction of document and tree blocking for a better exploitation of cache layers of modern CPUs. The technique is used by Lucchese *et al.* [LNO+15].
- Jin *et al.* [JYT16] provide an analytical comparison of cache blocking methods. Moreover, they propose a technique to select a traversal method and its optimal blocking parameters for effective use of memory hierarchy.

[TJY14] X. Tang, X. Jin, and T. Yang. *Cache-conscious runtime optimization for ranking ensembles*. In Proc. ACM SIGIR, 2014.

[JYT16] X. Jin, T. Yang, and X. Tang. *A comparison of cache blocking methods for fast execution of ensemble-based score computation*. In Proc. ACM SIGIR, 2016.

Thanks a lot for your attention!



References

Feature Selection

- [DC10] V. Dang and B. Croft. *Feature selection for document ranking using best first search and coordinate ascent*. In ACM SIGIR workshop on feature generation and selection for information retrieval, 2010.
- [GE03] I. Guyon and A. Elisseeff. *An introduction to variable and feature selection*. JMLR, 2003.
- [GLNP16] A. Gigli, C. Lucchese, F. M. Nardini, and R. Perego. *Fast feature selection for learning to rank*. In Proc. ACM ICTIR, 2016.
- [HZL+10] G. Hua, M. Zhang, Y. Liu, S. Ma, and L. Ru. *Hierarchical feature selection for ranking*. In Proc. WWW 2010.
- [LFC+12] L. Laporte, R. Flamary, S. Canu, S. D'ejean, and J. Mothe. *Non-convex regularizations for feature selection in ranking with sparse SVM*. Transactions on Neural Networks and Learning Systems, 10(10), 2012.
- [LPTY13] H.J. Lai, Y. Pan, Y. Tang, and R. Yu. *FSMRank: Feature selection algorithm for learning to rank*. Transactions on Neural Networks and Learning Systems, 24(6), 2013.
- [NA14] K. D. Naini and I. S. Altingovde. *Exploiting result diversification methods for feature selection in learning to rank*. In Proc. ECIR, 2014.
- [PCA+09] F. Pan, T. Converse, D. Ahn, F. Salvetti, and G. Donato. *Feature selection for ranking using boosted trees*. In Proc. ACM CIKM, 2009.
- [XHW+14] Z. Xu, G. Huang, K.Q. Weinberger, A.X. Zheng. *Gradient boosted feature selection*. In Proc. ACM SIGKDD 2014

References

Optimizing Efficiency within the Learning Process

- [AL13] N. Asadi and J. Lin. *Training efficient tree-based models for document ranking*. In Proc. ECIR, 2013.
- [LNO+16a] C. Lucchese, F. M. Nardini, S. Orlando, R. Perego, F. Silvestri, and S. Trani. *Post-learning optimization of tree ensembles for efficient ranking*. In Proc. ACM SIGIR, 2016.
- [LNO+17] C. Lucchese, F. M. Nardini, S. Orlando, R. Perego, and S. Trani. *X-DART: Blending dropout and pruning for efficient learning to rank*. In Proc. ACM SIGIR, 2017.
- [LNO+18] C. Lucchese, F. M. Nardini, S. Orlando, R. Perego, F. Silvestri, and S. Trani. *X-CLEAVER: Learning Ranking Ensembles by Growing and Pruning Trees*. ACM TIST, 2018.
- [VGB15] R. Korlakai Vinayak and R. Gilad-Bachrach. *DART: Dropouts meet Multiple Additive Regression Trees*. In PMLR, 2015.
- [WLM10] L. Wang, J. Lin, and D. Metzler. *Learning to efficiently rank*. In Proc. ACM SIGIR 2010.
- [XKWC13] Z. Xu, M. J. Kusner, K. Q. Weinberger, and M. Chen. *Cost-sensitive tree of classifiers*. In Proc. ICML, 2013.

References

Approximate Score Computation and Efficient Cascades

- [CGBC17b] R. Chen, L. Gallagher, R. Blanco, and J. S. Culpepper. *Efficient cost-aware cascade ranking in multi-stage retrieval*. In Proc. ACM SIGIR, 2017.
- [CZC+10] B. B. Cambazoglu, H. Zaragoza, O. Chapelle, J. Chen, C. Liao, Z. Zheng, and J. Degenhardt. *Early exit optimizations for additive machine learned ranking systems*. In Proc. ACM WSDM, 2010.
- [WLM11b] L. Wang, J. Lin, and D. Metzler. *A cascade ranking model for efficient ranked retrieval*. In Proc. ACM SIGIR, 2011.
- [XKW+14a] Z. Xu, M. J. Kusner, K. Q. Weinberger, M. Chen, and O. Chapelle. *Classifier cascades and trees for minimizing feature evaluation cost*. JMLR, 2014.

References

Efficient Traversal of Tree-based Models

- [ALdV14] N. Asadi, J. Lin, and A. P. de Vries. *Runtime optimizations for tree-based machine learning models*. IEEE TKDE, 2014.
- [TJY14] X. Tang, X. Jin, and T. Yang. *Cache-conscious runtime optimization for ranking ensembles*. In Proc. ACM SIGIR, 2014.
- [DLN+16] D. Dato, C. Lucchese, F. M. Nardini, S. Orlando, R. Perego, N. Tonellotto, and R. Venturini. *Fast ranking with additive ensembles of oblivious and non-oblivious regression trees*. ACM TOIS, 2016.
- [JYT16] X. Jin, T. Yang, and X. Tang. *A comparison of cache blocking methods for fast execution of ensemble-based score computation*. In Proc. ACM SIGIR, 2016.
- [LNO+15] C. Lucchese, F. M. Nardini, S. Orlando, R. Perego, N. Tonellotto, and R. Venturini. *Quickscore: A fast algorithm to rank documents with additive ensembles of regression trees*. In Proc. ACM SIGIR, 2015.
- [LNO+16b] C. Lucchese, F. M. Nardini, S. Orlando, R. Perego, N. Tonellotto, and R. Venturini. *Exploiting CPU SIMD extensions to speed-up document scoring with tree ensembles*. . In Proc. ACM SIGIR, 2016.
- [LLN+18] F. Lettich, C. Lucchese, F. M. Nardini, S. Orlando, R. Perego, N. Tonellotto, and R. Venturini. *Parallel Traversal of Large Ensembles of Decision Trees*. IEEE TPDS, 2018.
- [JYT16] X. Jin, T. Yang, and X. Tang. *A comparison of cache blocking methods for fast execution of ensemble-based score computation*. In Proc. ACM SIGIR, 2016.
- [YZZ+18] T. Ye, H. Zhou, W. Y. Zou, B. Gao, R. Zhang. *RapidScorer: Fast Tree Ensemble Evaluation by Maximizing Compactness in Data Level Parallelization*. In. Proc. ACM SIGKDD, 2018.

References

Other

[QR] *QuickRank, A C++ suite of Learning to Rank algorithms.* <http://quickrank.isti.cnr.it>

[YLTRC] O. Chapelle and Y. Chang. *Yahoo! learning to rank challenge overview.* JMLR, 2011.

[CBY15] B. B. Cambazoglu and R. Baeza-Yates. *Scalability Challenges in Web Search Engines.* Morgan & Claypool Publishers, 2015.