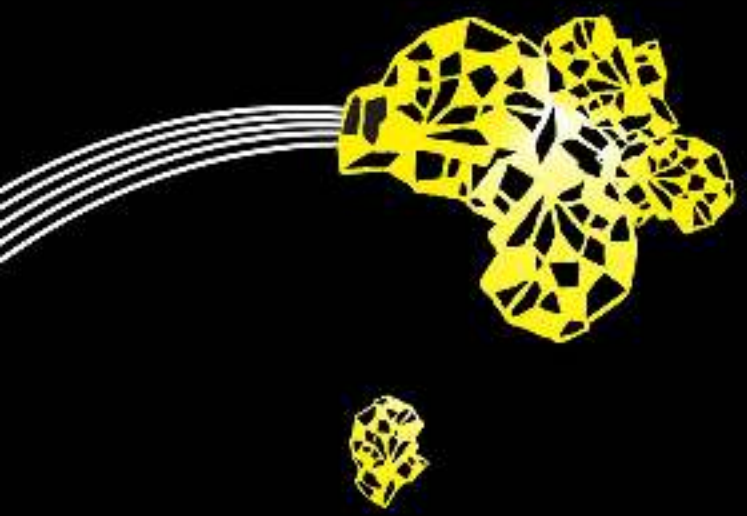
UNIVERSITY OF TWENTE.

RANKING LEARNING TO RANK METHODS

Niek Tax, Sander Bockting, Djoerd Hiemstra

http://www.cs.utwente.nl/~hiemstra



LEARning Next gEneration Rankers (LEARNER 2017)

1 October 2017

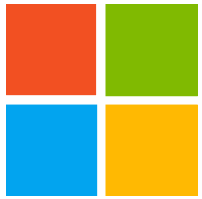
MANY LEARNING-TO-RANK METHODS

- RankSVM, RankBoost, RankCSA, RankDE, RankMGP, RankNet, RankRLS, ...
- AdaRank, CCRank, DirectRank, FenchelRank, FSMRank, GPRank, LambdaRank, PermuRank, SmoothRank, RSRank, ...
- BagBoo, DCMP, GAS-E, LRUF, LARF, MCP, OWPC, VFLR, ...

https://openclipart.org/detail/181678/desperate-man

MANY BENCHMARK DATASETS

- LETOR 2.0
- LETOR 3.0
- LETOR 4.0
- MSLR-web10/30k
- Yahoo! Learning to Rank Challenge
- Yandex Internet Mathematics Competition 2009
- WCL2R
- AOL



Microsoft

YAHOO!

Яндекс

todocl
TODO CHILE EN INTERNET

Aol.

FEATURE REPRESENTATIONS

- Datasets offer **feature set representations** of the to-be-ranked documents instead of the documents themselves.
- Therefore, any difference in ranking performance is due to the ranking algorithm and not the features used.

https://openclipart.org

BENCHMARK DATASETS

Benchmark collection	# of datasets
AOL	1
LETOR 2.0	3
LETOR 3.0	7
LETOR 4.0	2
MSLR	2
WCL2R	2
Yahoo! Learning to Rank Challenge	2
Yandex Internet Mathematics 2009 contest	1
Total	**20**

Table 1: Included learning to rank evaluation benchmark collections

(for instance LETOR 4.0: million query sets 2007 and 2008)

LITERATURE SEARCH

BY FORWARD REFERENCES (BENCHMARKS WITH OVERVIEW PAPER)

Benchmark	Paper	# of forward references
LETOR 1.0 & 2.0	Liu et al. [135]	307
LETOR 3.0	Qin et al. [165]	105
Yahoo! Learning to Rank Challenge	Chapelle et al. [44]	102
AOL dataset	Pass et al. [156]	339
WCL2R	Alcântara et al. [10]	2

LITERATURE SEARCH
BY SEARCH (GOOGLE SCHOLAR)

Benchmark	Google Scholar search results
LETOR 4.0	75 results
MSLR-web10k	16 results
MSLR-web30k	15 results
Yandex Internet Mathematics Competition	1 result

https://openclipart.org

LITERATURE FILTERING

About 150 learning-to-rank papers

***Remove* if:**

- other methodology (e.g. train/test data);
- other task (rank aggregation, transfer ranking, …);
- additional features;
- no exact data (e.g. only graphs);
- other evaluation metric;
- other reported baseline performance;
- no baseline performance mentioned.

https://openclipart.org

Method	Described	Evaluated	Method	Described	Evaluated
AdaRank-MAP	[233]	L2, L3, L4	Linear Regression	[57]	L3, [219, 213]
AdaRank-NDCG	[233]	L2, L3, L4, [36, 199]	ListMLE	[232]	[132, 130, 88]
ADMM	[77]	[77]	ListNet	[39]	L2, L3, L4
ApproxAP	[167]	[167]	ListReg	[229]	[229]
ApproxNDCG	[167]	[167]	LRUT	[205]	[205]
BagBoo	[157]	[86]	MCP	[122]	[122]
Best Single Feature		[95]	MLR	[171]	L2
BL-MART	[86]	[86]	MultiStageBoost	[111]	[111]
BoltzRank-Single	[214]	[214, 216]	NewLoss	[159]	[159]
BoltzRank-Pair	[214]	[214, 86, 216]	OWPC	[209]	[209]
BT	[252]	[252]	PERF-MAP	[154]	[205]
C-CRF	[168]	[168]	PermuRank	[234]	[234]
CA	[141]	[36, 199]	Q.D.KNN	[90]	[226]
CCRank	[223]	[223]	RandomForest		[95]
CoList	[88]	[88]	Rank-PMBGP	[179]	[179]
Consistent-RankCosine	[172]	[199]	RankAggNDCG	[226]	[226]
DCMP	[173]	[173]	RankBoost	[81]	L2, L3, L4, [36, 10]
DirectRank	[199]	[199]	RankCSA	[99]	[99]
EnergyNDCG	[80]	[80]	RankDE	[25]	[179]
FBPCRank	[120]	[120]	RankELM (pairwise)	[256]	[256]
FenchelRank	[118]	[118, 119, 122]	RankELM (pointwise)	[256]	[256]
FocusedBoost	[148]	[148]	RankMGP	[129]	[129]
FocusedNet	[148]	[148]	RankNet	[32]	[36, 155, 148]
FocusedSVM	[148]	[148]	RankRLS	[152]	[151]
FP-Rank	[188]	[188]	RankSVM	[101, 110]	L2, L3, [36, 80, 99, 10]
FRank	[207]	L2, L3, [219]	RankSVM-Primal		L3, [120]
FSMRank	[121]	[121, 122]	RankSVM-Struct		L3, L4
FSM^{SVM}	[121]	[121]	RCP	[78]	[78]
GAS-E	[91]	[121]	RE-QR	[211]	[211]
GP	[66]	[10]	REG-SHF-SDCG	[230]	[230]
GPRank	[187]	[204]	Ridge Regression	[57]	L3
GRankRLS	[151]	[151]	RSRank	[195]	[118]
GroupCE	[130]	[130]	SmoothGrad	[123]	[199]
GroupMLE	[132]	[130]	SmoothRank	[47]	L3, [47]
IntervalRank	[144]	[144, 80]	SoftRank	[200, 97]	[167]
IPRank	[225]	[225, 204]	SortNet	[175]	[175, 80]
KeepRank	[52]	[52]	SparseRank	[119]	[119]
Kernel-PCA RankBoost	[75]	[75, 179]	SVD-RankBoost	[131]	[131]
KL-CRF	[213]	[213]	SVM^{MAP}	[246]	L3, [219, 234, 148]
LAC-MR-OR	[212]	[212]	SwarmRank	[72]	[179]
LambdaMART	[31]	[15, 86]	TGRank	[118]	[118]
LambdaNeuralRank	[155]	[155]	TM	[252]	[252, 155, 199]
LambdaRank	[34]		VFLR	[38]	[38]
LARF	[204]	[204]			

COMPARISON METHODOLOGY

WINNING NUMBER (Liu et al., 2007)

$$\text{Winning Number}_i(M) = \sum_{j=1}^{n} \sum_{k=1}^{m} I_{\{M_i(j) > M_k(j)\}}$$

$$I'_{M_i(j) > M_k(j)} = \begin{cases} 1 & \text{if } M_i(j) \text{ and } M_k(j) \text{ are both defined and } M_i(j) > M_k(j), \\ 0 & \text{otherwise} \end{cases}$$

$M_k(j)$ = performance of method k on dataset j

LIU, T. Y., XU, J., QIN, T., XIONG, W., AND LI, H. LETOR: Benchmark Dataset for Research on Learning to Rank for Information Retrieval. In *Proceedings of the SIGIR 2007 Workshop on Learning to Rank for Information Retrieval* (2007), pp. 3–10.

COMPARISON METHODOLOGY

NORMALIZED WINNING NUMBER

Normalized Winning Number (NWN)

$$\mathrm{NWN}_i(M) = \frac{\mathrm{WN}_i(M)}{\mathrm{IWN}_i(M)}$$

where IWN is the Ideal Winning Number, defined as

$$\mathrm{IWN}_i(M) = \sum_{j=1}^{n} \sum_{k=1}^{m} D_{\{M_i(j), M_k(j)\}}$$

$$D_{\{M_i(j), M_k(j)\}} = \begin{cases} 1 & \text{if } M_i(j) \text{ and } M_k(j) \text{ are both defined,} \\ 0 & \text{otherwise} \end{cases}$$

COMPARISON RESULTS

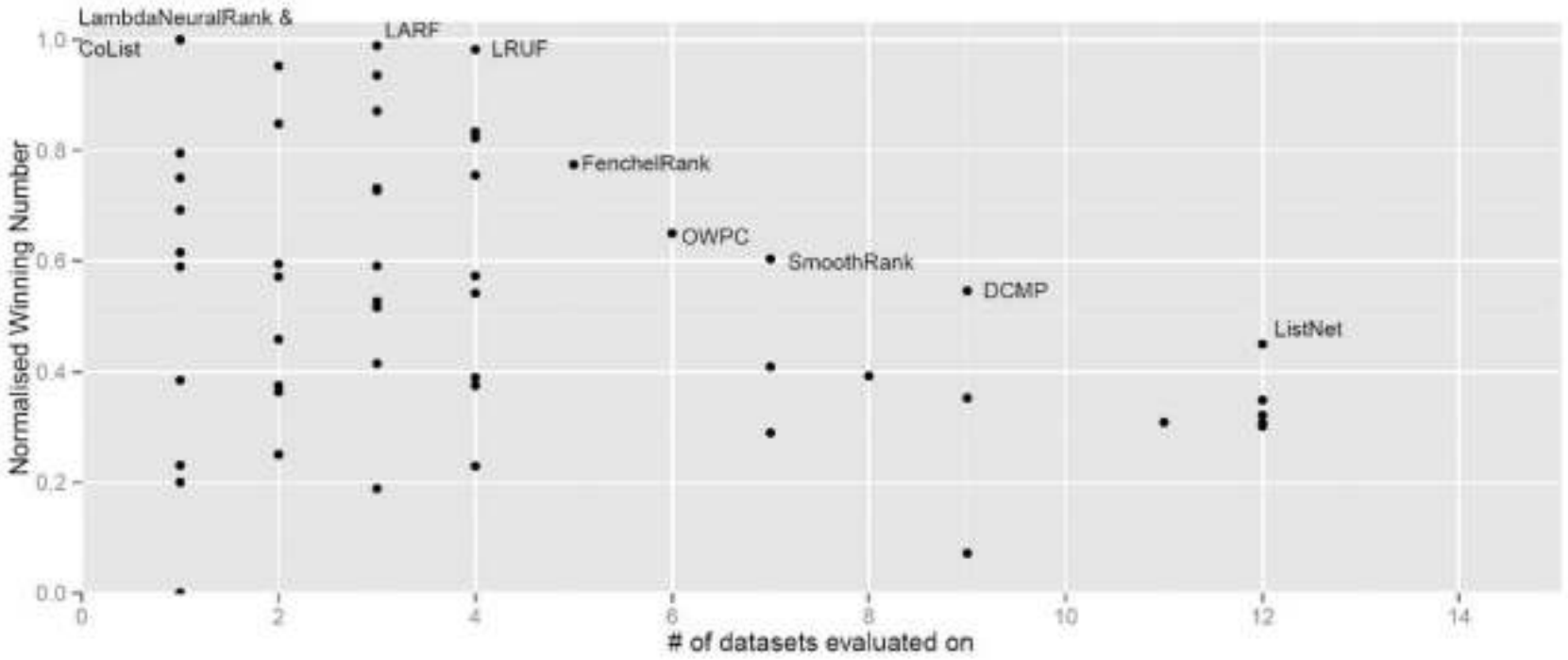


Figure 1: NDCG@3 comparison of 87 learning to rank methods

COMPARISON RESULTS

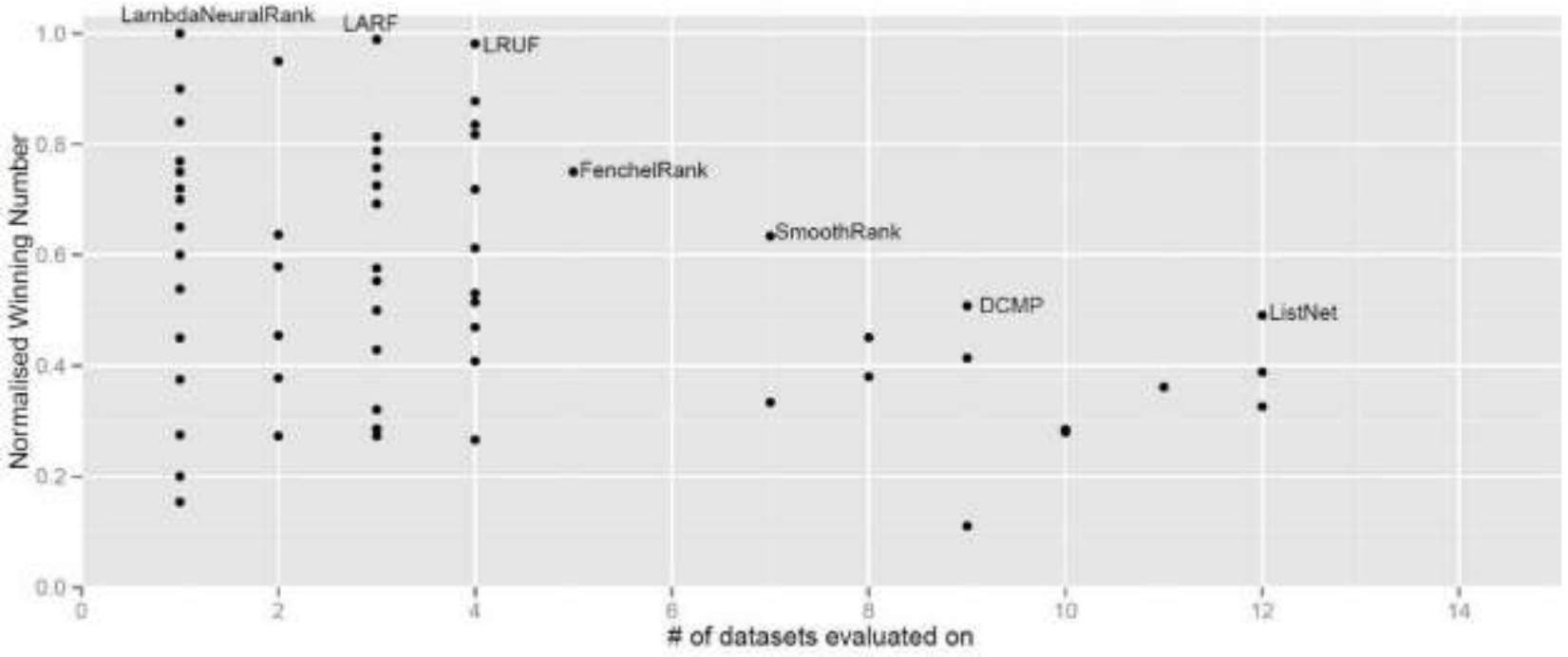


Figure 2: NDCG@5 comparison of 87 learning to rank methods

COMPARISON RESULTS

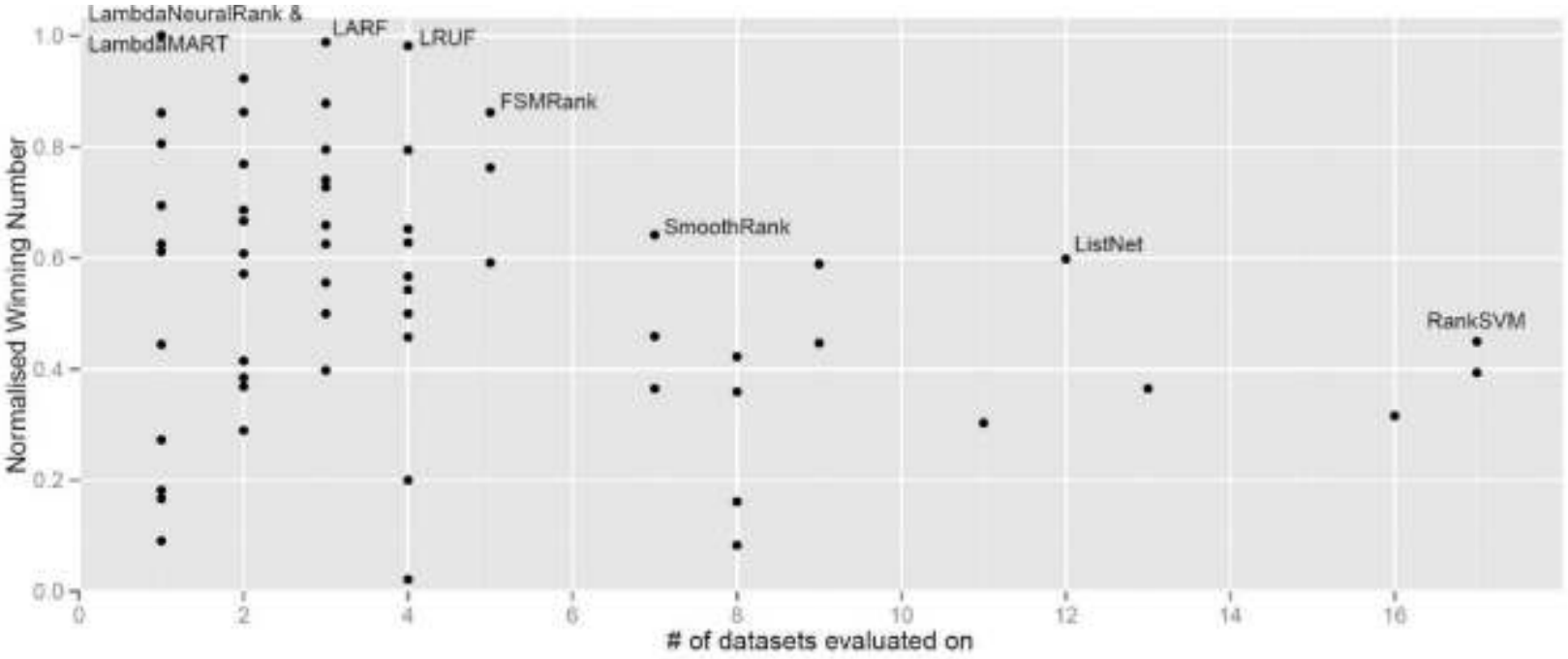


Figure 3: NDCG@10 comparison of 87 learning to rank methods

COMPARISON RESULTS

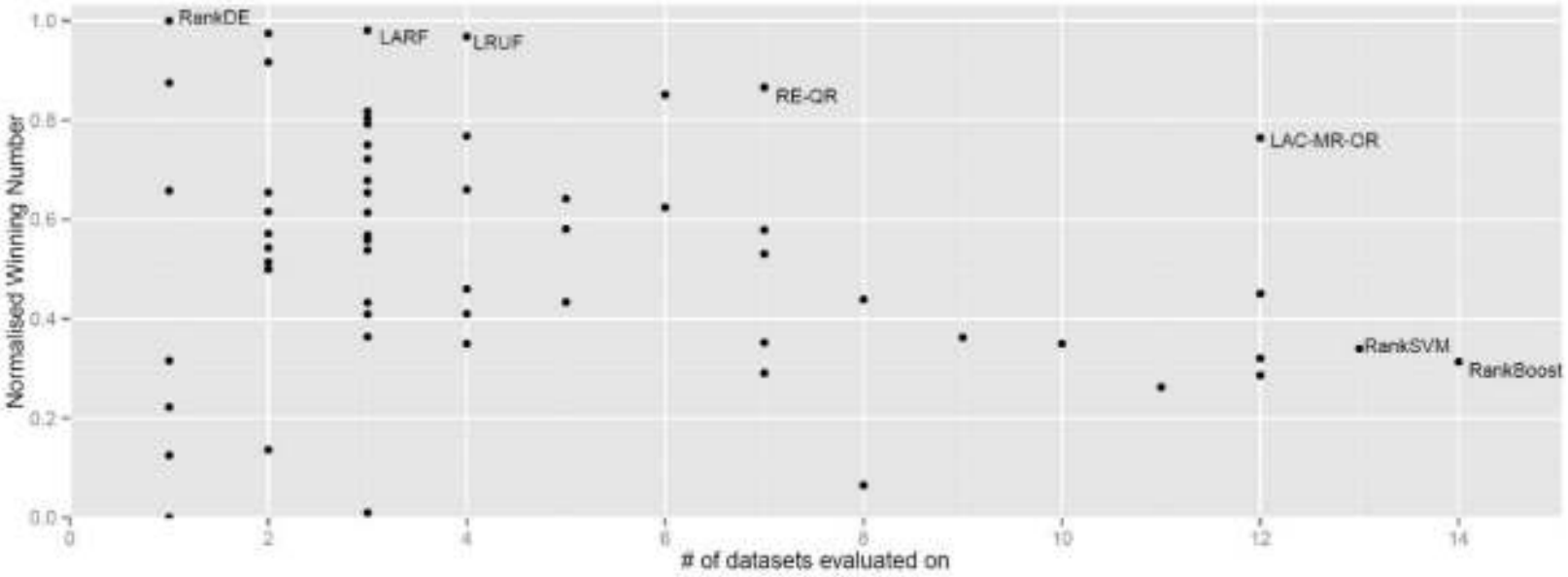


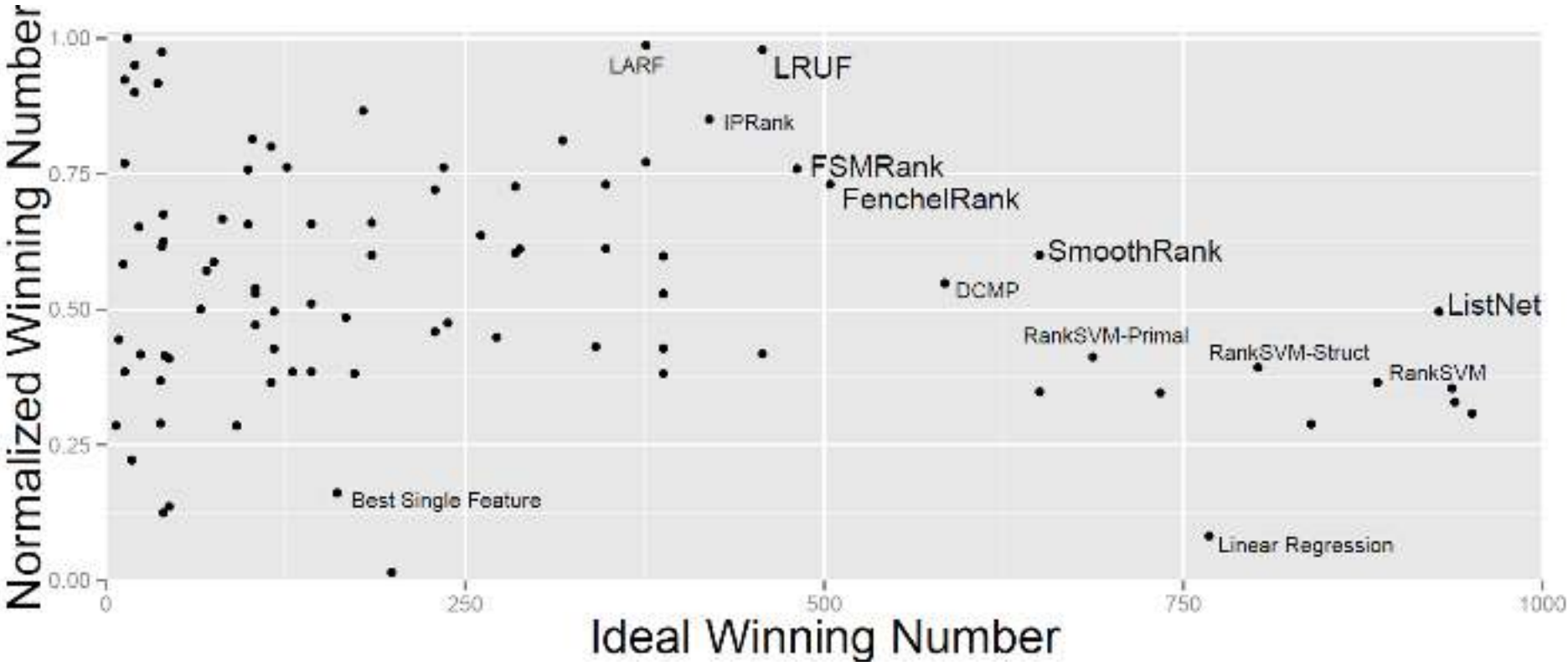
Figure 4: MAP comparison of 87 learning to rank methods

PARETO-OPTIMAL ($NWN > 0.5,\ n > 2$)

- nDCG@3: DCMP, SmoothRank, OWPC, FenchelRank, LRUF, LARF
- nDCG@5: DCMP, SmoothRank, FenchelRank, LRUF, LARF
- nDCG@10: ListNet, SmoothRank, FSMRank, LRUF, LARF
- MAP: LAC-MR-OR, RE-QR, LRUF, LARF

(listed in decreasing nr. of datasets n and increasing NWN)

COMPARISON RESULTS



PARETO-OPTIMAL

- Cross-metric:
  - ListNet, SmoothRank, FenchelRank, FSMRank, LRUF , LARF

(listed in decreasing IWN and increasing NWN)

https://openclipart.org

SENSITIVITY

Q: What if we ignore a measure?

A: Pareto-optimal methods remain the same, except:

- FSMRank is not optimal when MAP is ignored;
- FenchelRank is not optimal when nDCG@3 or nDCG@5 are ignored

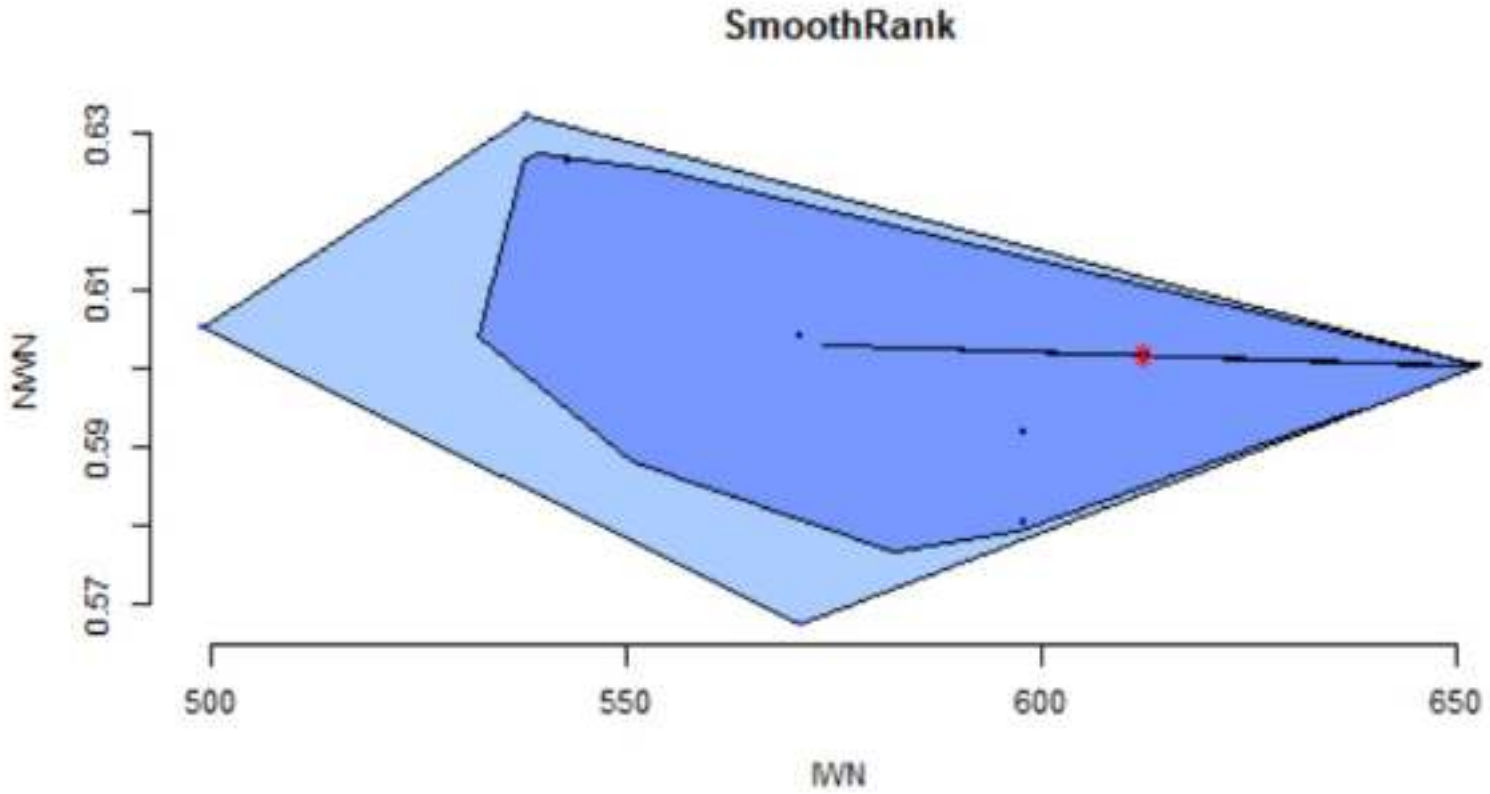
SENSITIVITY

Q: What if we ignore a dataset?

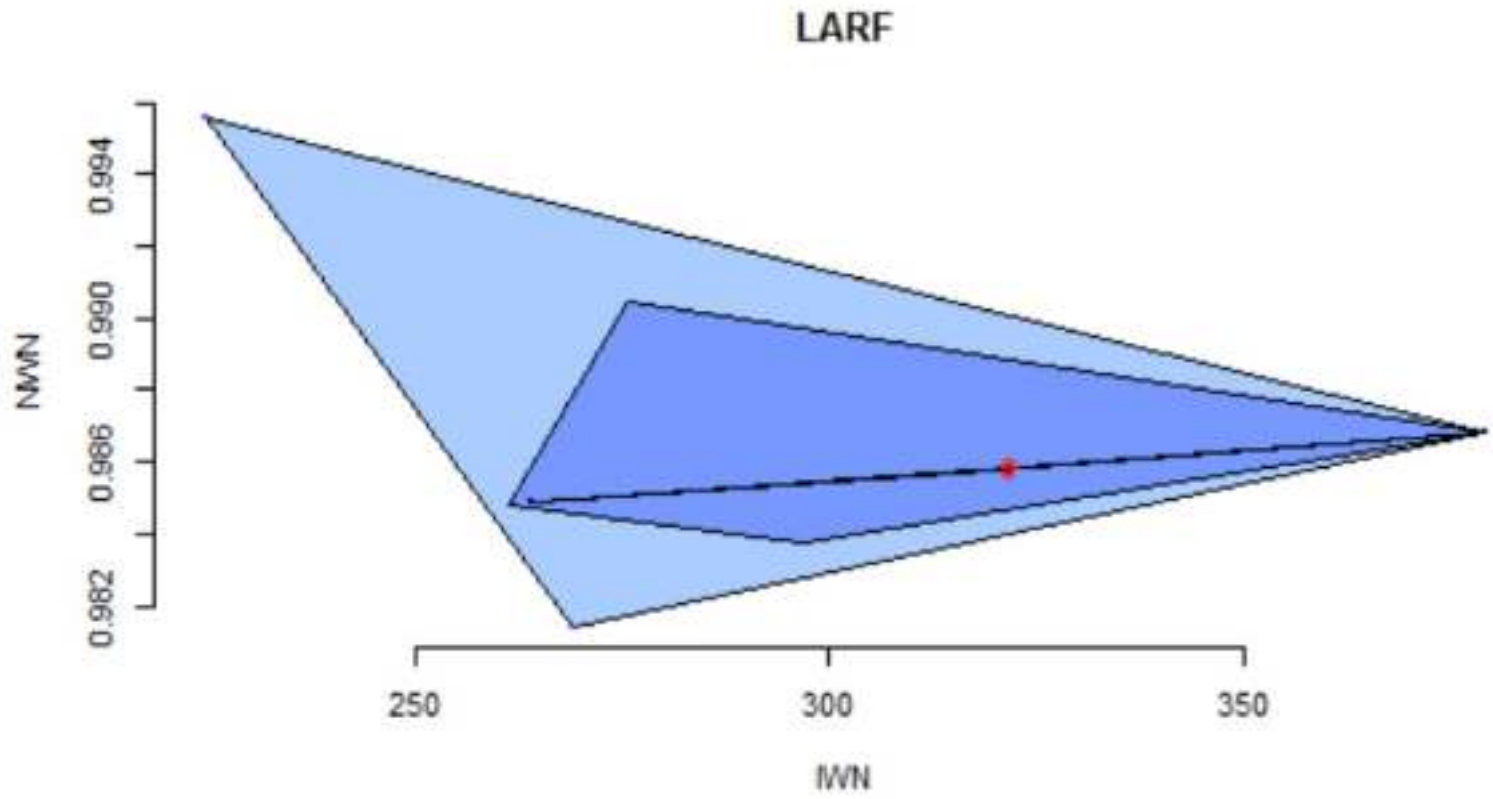
A: Use bagplots (Rousseeuw et al. 1999)

- Bivariate generalization of boxplot
  - Dark Blue: Bag with n/2 observations
  - Light Blue: Magnifies bag by factor 3

SENSITIVITY



SENSITIVITY



LIMITATIONS

- Missing measures
  - (e.g. RE-QR is not evaluated for nDCG)
- Relies on other researchers performing correct evaluations
- Possible bias in published evaluation runs

CONCLUSIONS

- New way of comparing Learning-to-Rank methods using sparse evaluation data
  - Normalized Winning Number (ranking accuracy)
  - Ideal Winning Number (confidence in ranking)
- Best Learning-to-rank methods (2015)
  - ListNet, SmoothRank, FenchelRank, FSMRank, LRUF , LARF
- Fill up the missing combinations of methods/datasets/measures!

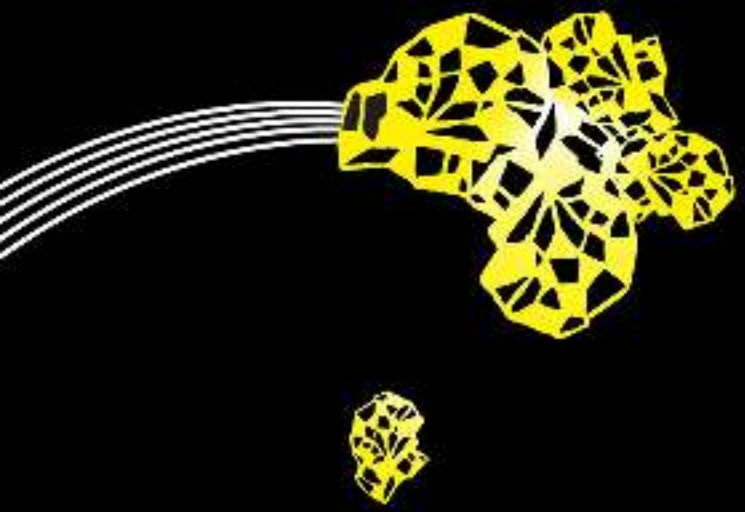
FURTHER READING

- Niek Tax, Sander Bockting, and Djoerd Hiemstra.
  "A cross-benchmark comparison of 87 learning to rank methods",
  Information Processing and Management 51(6), November 2015
  http://www.cs.utwente.nl/~hiemstra/papers/ipm2015.pdf

  (IPM 2015 Best Paper Award)

UNIVERSITY OF TWENTE.

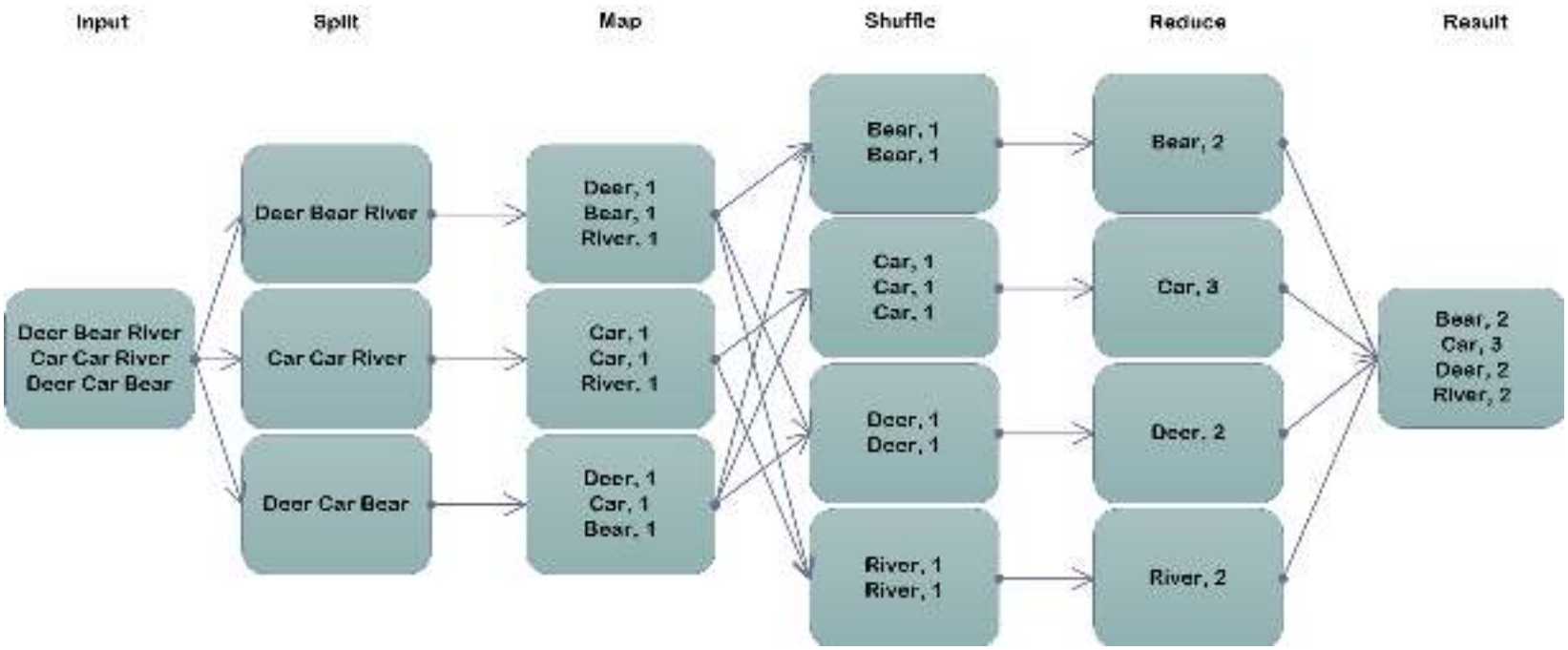
LEARNING TO RANK ON MAP/REDUCE



HOW TO FURTHER INCREASE PERFORMANCE ?

- Deep Neural Networks
  - Weak supervision: Dehghani et al. SIGIR 2017
- MOOOORE training data!
- “Artificial” training data from:
  - Google :-)
  - Standard rankers (BM25)
  - Knowledge databases

MAP/REDUCE

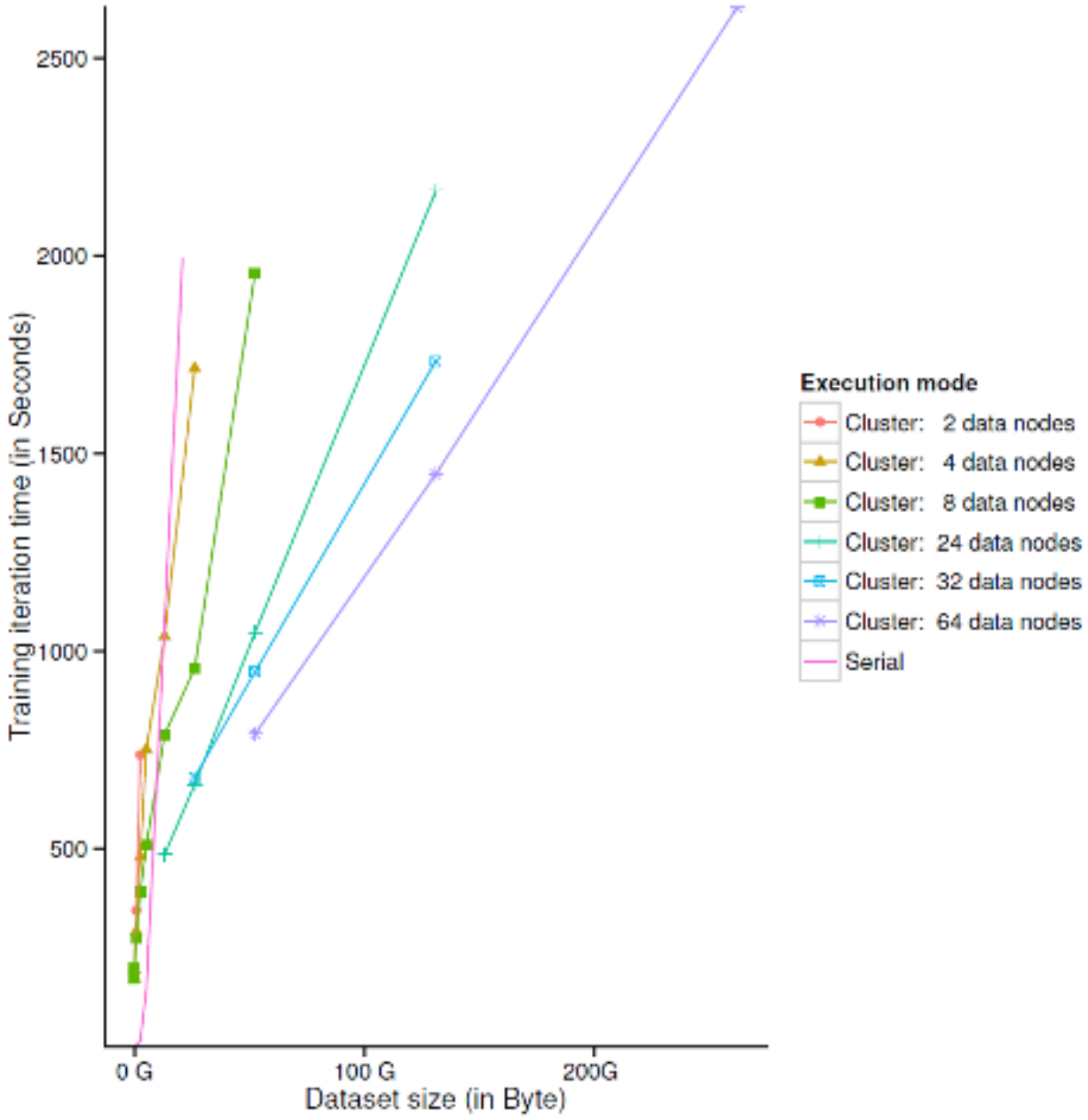

Input
Split
Map
Shuffle
Reduce
Result
Deer Bear River
Car Car River
Deer Car Bear
Deer Bear River
Car Car River
Deer Car Bear
Deer, 1
Bear, 1
River, 1
Car, 1
Car, 1
River, 1
Deer, 1
Car, 1
Bear, 1
Bear, 1
Bear, 1
Car, 1
Car, 1
Car, 1
Deer, 1
Deer, 1
River, 1
River, 1
Bear, 2
Car, 3
Deer, 2
River, 2
Bear, 2
Car, 3
Deer, 2
River, 2

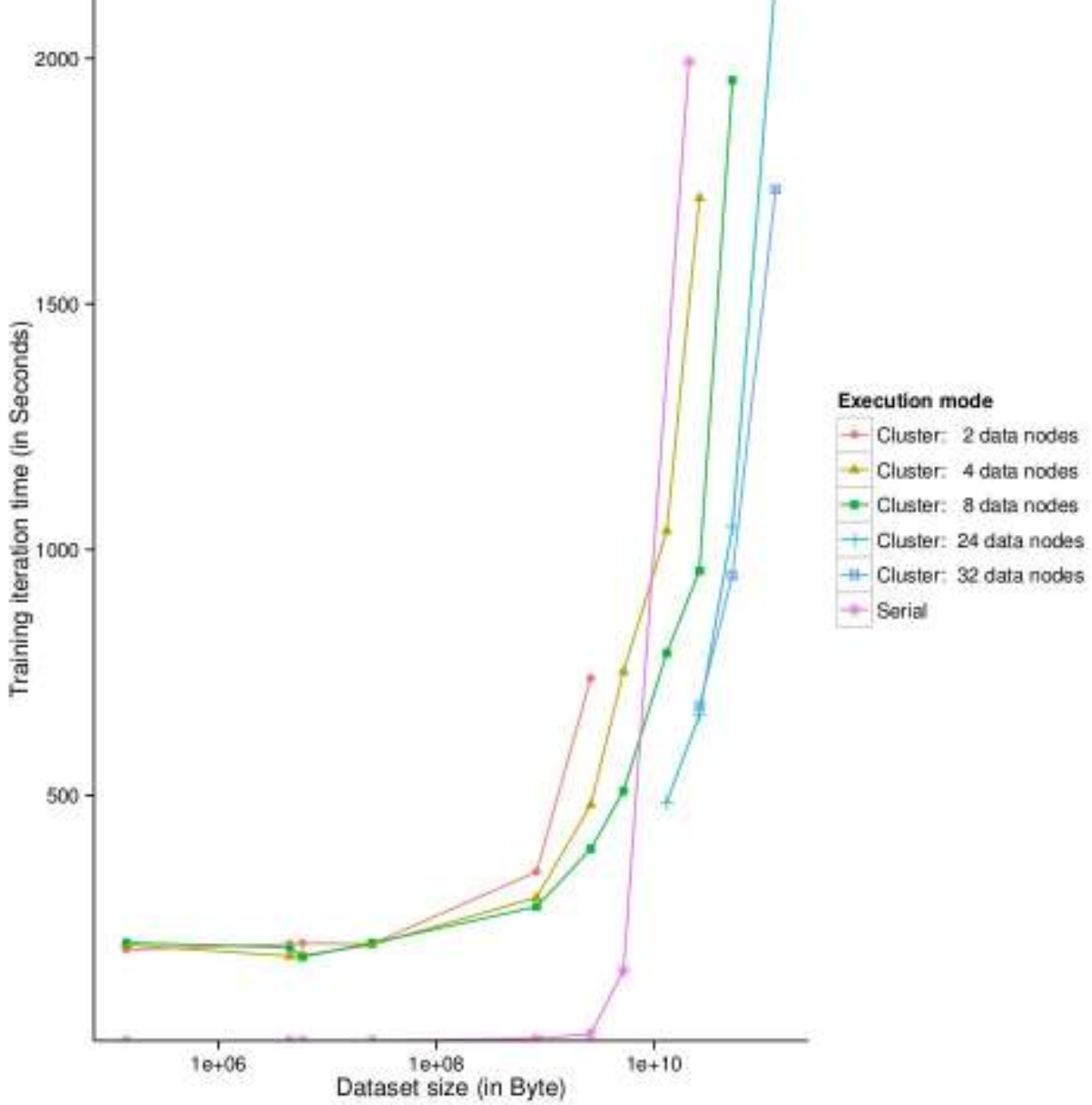
DATA SETS

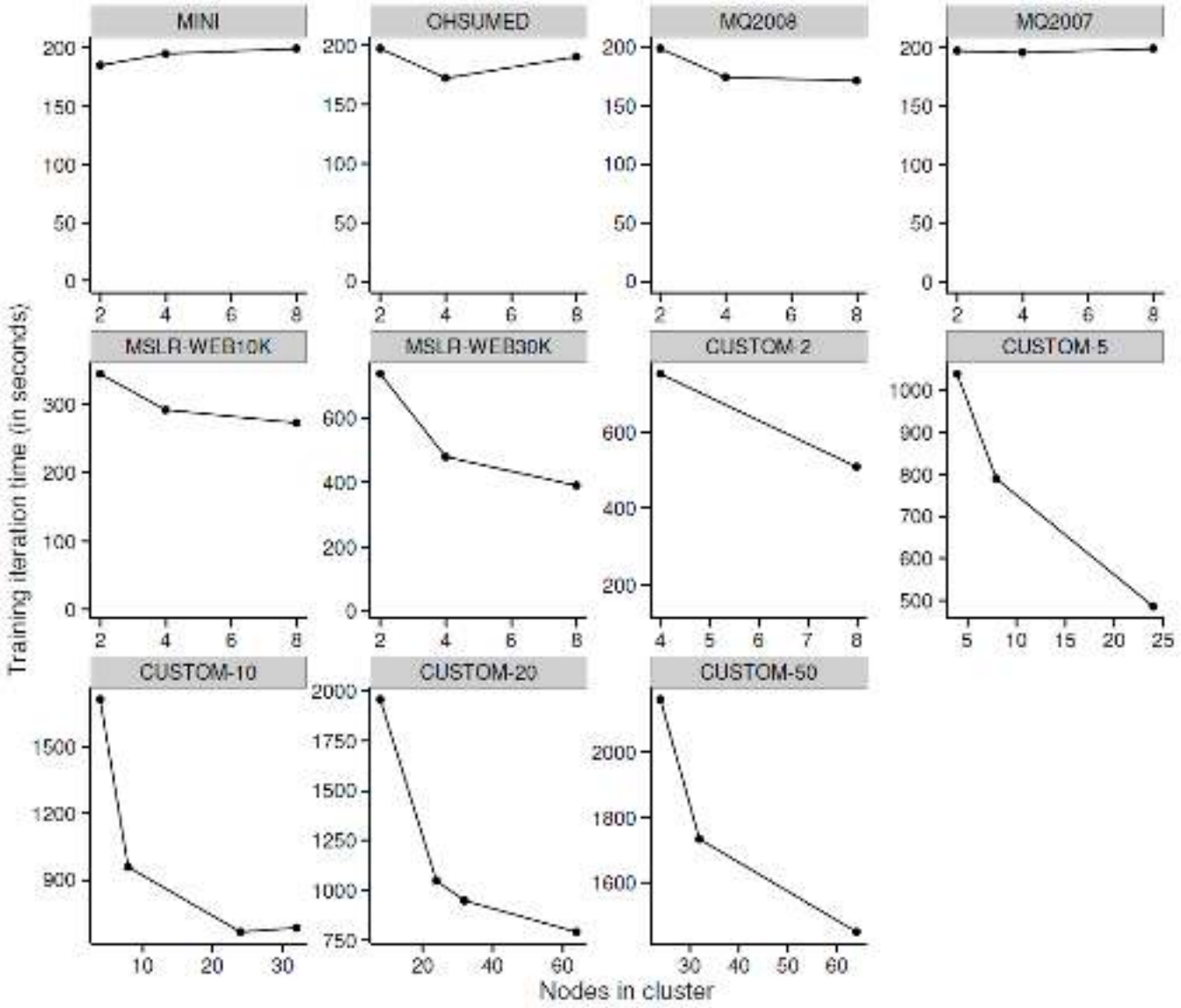
Data set	Collection	Single fold training size
MINI	GENERATED	143.38 KB
OHSUMED	LETOR 3.0	4.55 MB
MQ2008	LETOR 4.0	5.93 MB
MQ2007	LETOR 4.0	25.52 MB
MSLR-WEB10K	MSLR-WEB10K	938.01 MB
MSLR-WEB30K	MSLR-WEB30K	2.62 GB
CUSTOM-2	GENERATED	5.25 GB
CUSTOM-5	GENERATED	13.12 GB
CUSTOM-10	GENERATED	26.24 GB
CUSTOM-20	GENERATED	52.42 GB
CUSTOM-50	GENERATED	131.21 GB
CUSTOM-100	GENERATED	262.41 GB

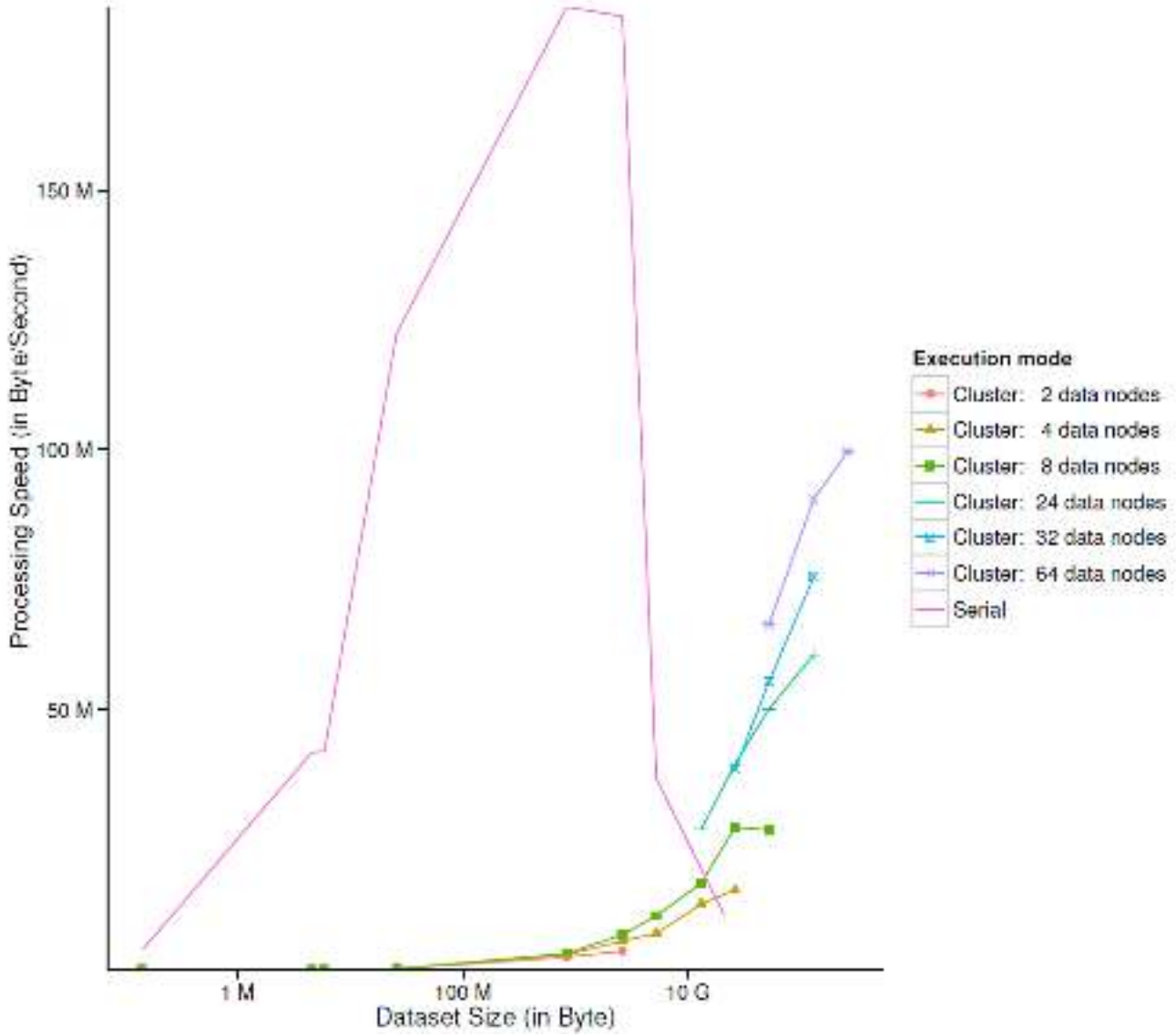
CONVERGENCE GRAPHS

- Single node:
  - ListNet, RankLib implementation
- 2 – 64 nodes Hadoop cluster
  - ListNet, MapReduce implementation


2500
2000
1500
1000
500
Training iteration time (in Seconds)
0 G
100 G
200G
Dataset size (in Byte)
Execution mode
Cluster: 2 data nodes
Cluster: 4 data nodes
Cluster: 8 data nodes
Cluster: 24 data nodes
Cluster: 32 data nodes
Cluster: 64 data nodes
Serial


2000
1500
1000
500
Training iteration time (in Seconds)
1e+06
1e+08
1e+10
Dataset size (in Byte)
Execution mode
Cluster: 2 data nodes
Cluster: 4 data nodes
Cluster: 8 data nodes
Cluster: 24 data nodes
Cluster: 32 data nodes
Serial


MINI
OHSUMED
MQ2008
MQ2007
MSLR-WEB10K
MSLR-WEB30K
CUSTOM-2
CUSTOM-5
CUSTOM-10
CUSTOM-20
CUSTOM-50
Training iteration time (in seconds)
Nodes in cluster


150 M
100 M
50 M
Processing Speed (in Byte/Second)
1 M
100 M
10 G
Dataset Size (in Byte)
Execution mode
Cluster: 2 data nodes
Cluster: 4 data nodes
Cluster: 8 data nodes
Cluster: 24 data nodes
Cluster: 32 data nodes
Cluster: 64 data nodes
Serial

CONCLUSIONS & CONTRIBUTIONS

- Single-machine ListNet does not scale well to data sets larger than physical memory
- MapReduce can increase processing speed of data sets larger than physical memory

- High MapReduce job scheduling overhead:
  - 150-200 seconds per iteration
  - Job scheduling overhead is independent of data set size

FUTURE WORK

- Generating artificial training data;
- Other distributed computing models:
  - GPUs
  - Spark
- Clever optimization algorithms:
  - Feature normalization speeds up convergence of gradient descent (see: Ng, Leung & Luk. *Neural Processing Letters 9*, 1999)