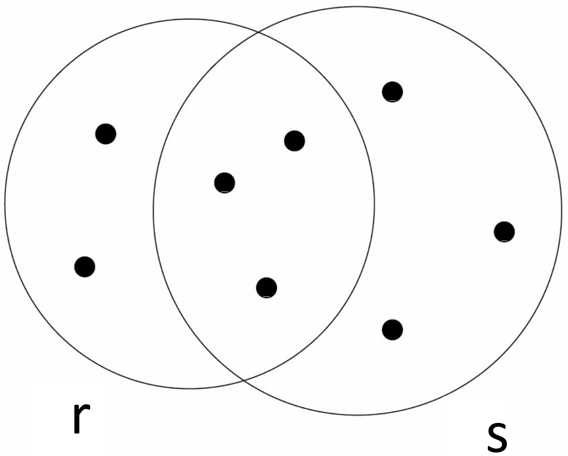
Set Similarity Joins on MapReduce: An Experimental Survey

Fabian Fier

Nikolaus Augsten, Panagiotis Bouros,
Ulf Leser, Johann-Christoph Freytag

Set Similarity Join

- Example (self-join):
  - Input:
    - An input data set of records *R*
    - A similarity function *sim*
    - A similarity threshold *t*
  - Output:
    - All pairs of records from *R* where *sim(r, s)* $\geq$ *t* (*r, s* $\in$ *R*)



Compared Algorithms

• ClusterJoin (CJ):	PVLDB, 2014
• MRGroupJoin (GJ):	PVLDB, 2015
• FullFilteringJoin (FF):	Assoc. for Comp. Linguistics, 2008
• MGJoin (MG):	TKDE, 2013
• MassJoin (MJ):	ICDE, 2014
• MRSimJoin (MR):	SIGMOD, 2012
• SSJ-2R (S2):	ICDM, 2010
• VernicaJoin (VJ):	SIGMOD, 2010
• V-SMART (VS):	PVLDB, 2012
• FS-Join (FS):	ICDE, 2017

Main Contribution: Benchmark

		CJ	GJ	FF	MG	MJ	MR	S2	VJ	VS	FS
ClusterJoin	CJ								=	>	
MRGroupJoin	GJ										
FullFilteringJoin	FF							<	<		
MGJoin	MG								>		
MassJoin	MJ								>		<
MRSimJoin	MR										
SSJ-2R	S2								>		
VernicaJoin	VJ									< >	<
V-SMART	VS										<
FS-Join	FS										

← Previous Comparisons →

Fair Benchmark

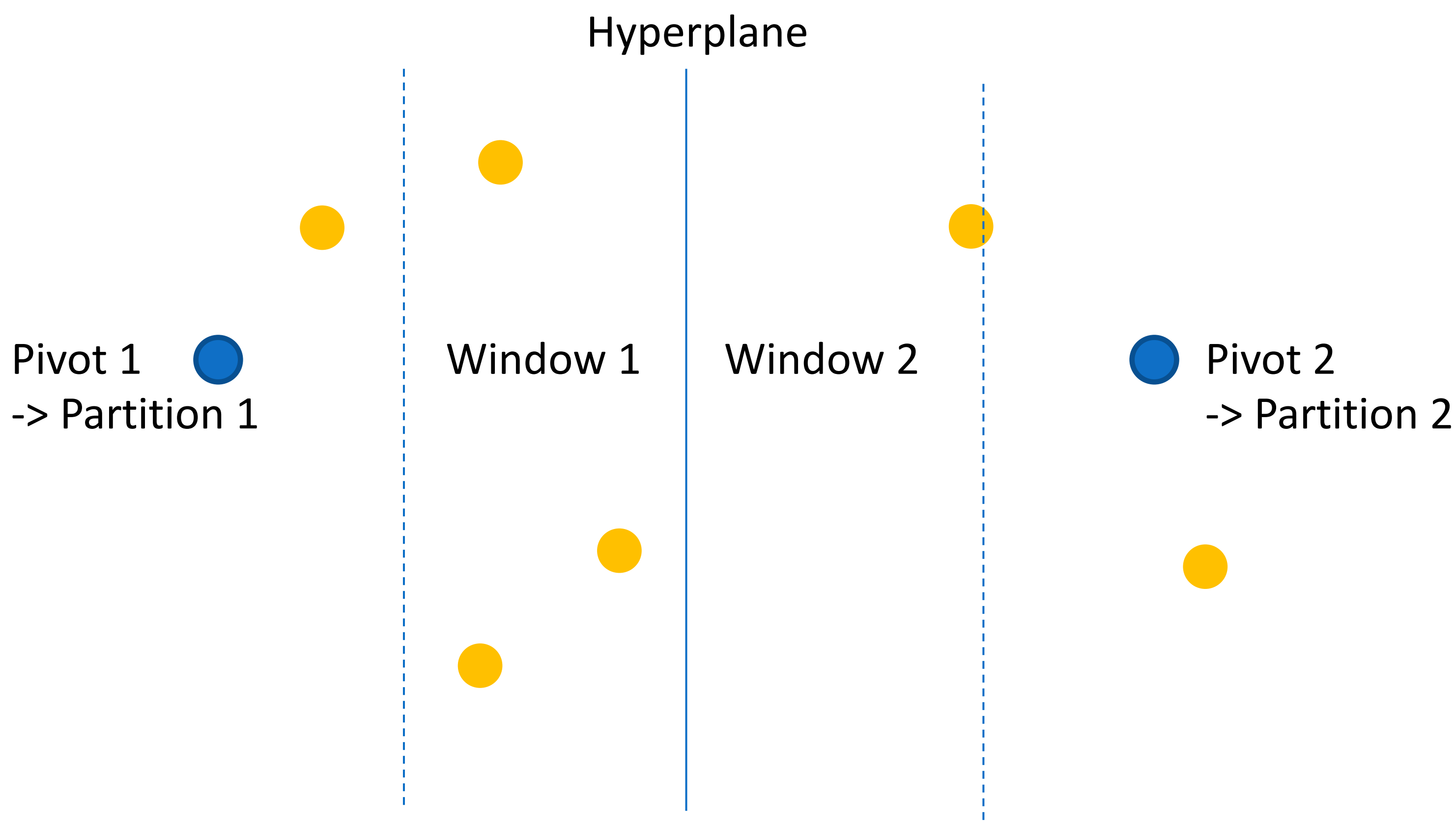
Algorithm	Approach	Preprocessing	Partitioning	Load balancing	number of MR runs
ClusterJoin	Metric Partitioning	No	Based on random pivots and hashing	Sampling-based	2
MRSimJoin	Metric Partitioning	No	Based on iterative random pivots	Memory-threshold-based	Random
MassJoin	Filter-and-verify	No			3
SSJ-2R	Filter-and-verify	Token frequencies	Remainder file	Yes	2
Vernica Join	Filter-and-verify	Token frequencies	Hash-based	Frequency-based	4 [1]
V-SMART Join	Filter-and-verify	No	Hash-based	Cardinality-based	2 [2]

...

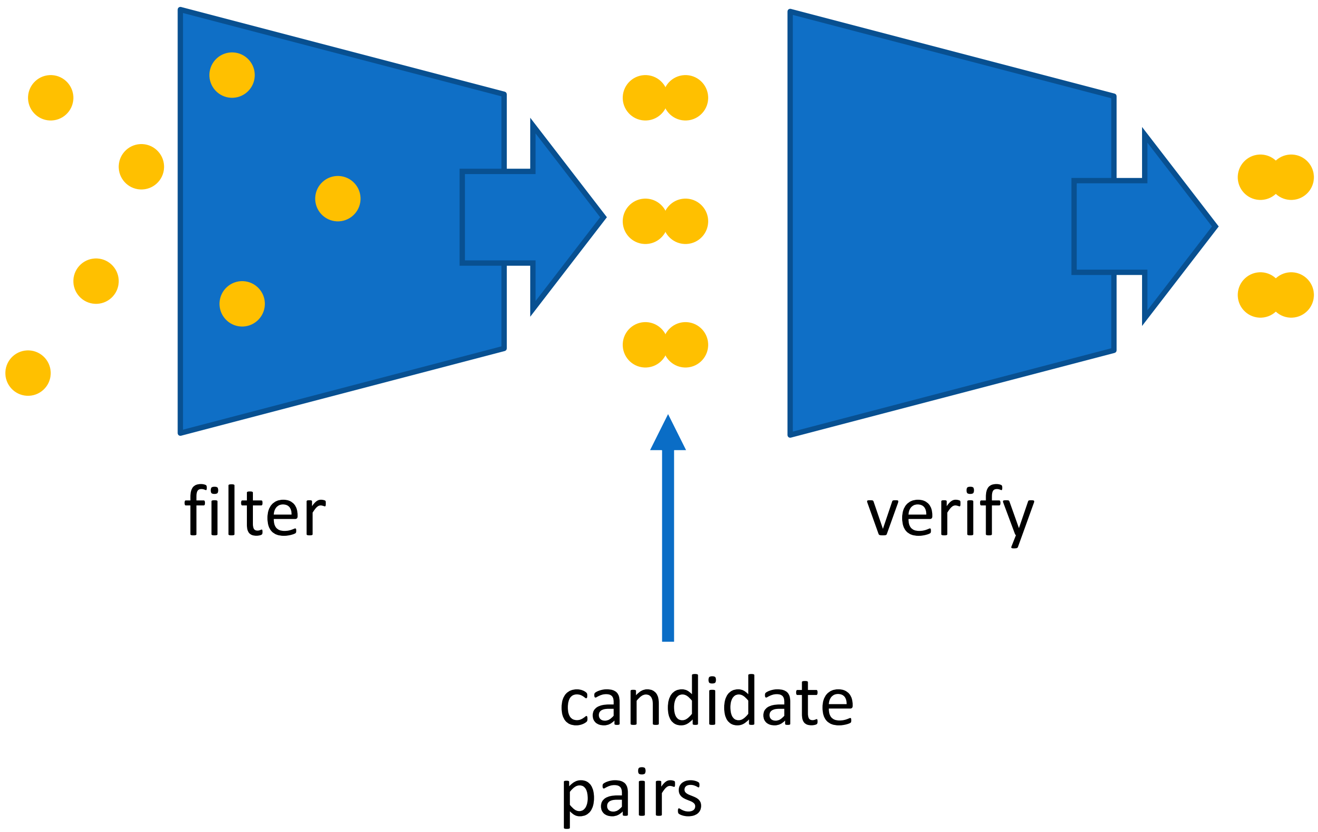
...

- SSJ ≠ SSJ
- Hadoop
- Similarity Measure
- Reimplemented Algorithms
- ...

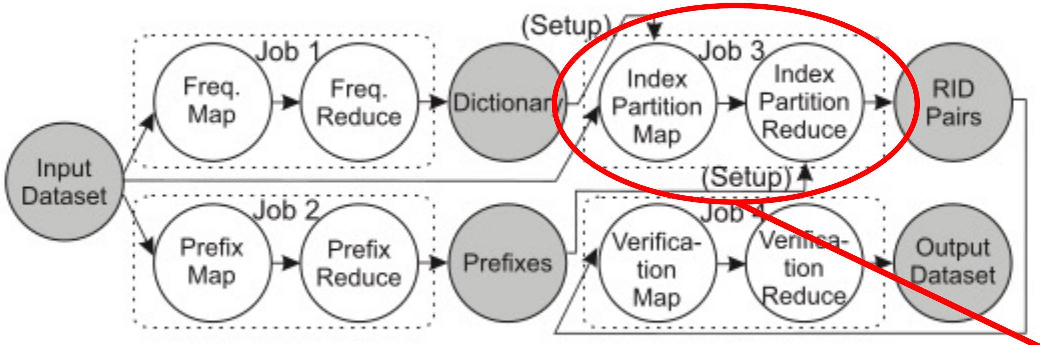
Metric-based Approach



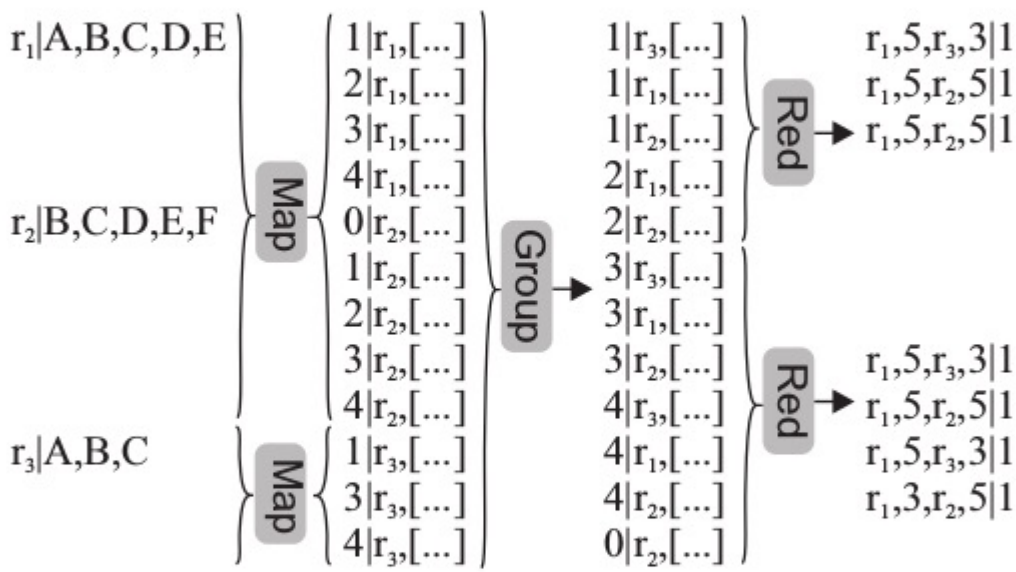
Filter-and-verification Approach



Analysis and Experiments



- Critical Jobs
- Theoretical analysis
- Experiments

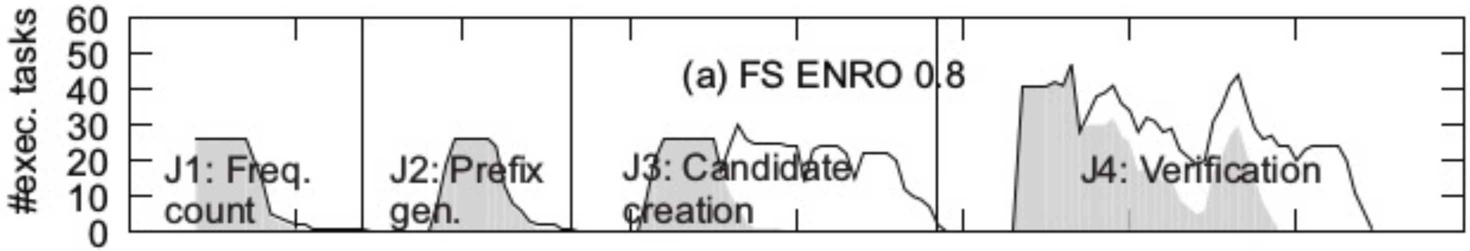


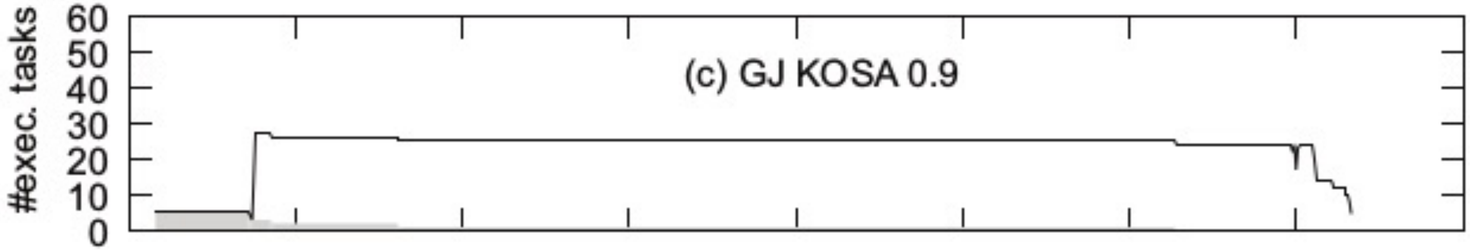
Insight 1: Benchmark

- Winners (average rank): VJ, GJ, FS
- 5 algorithms without best runtimes

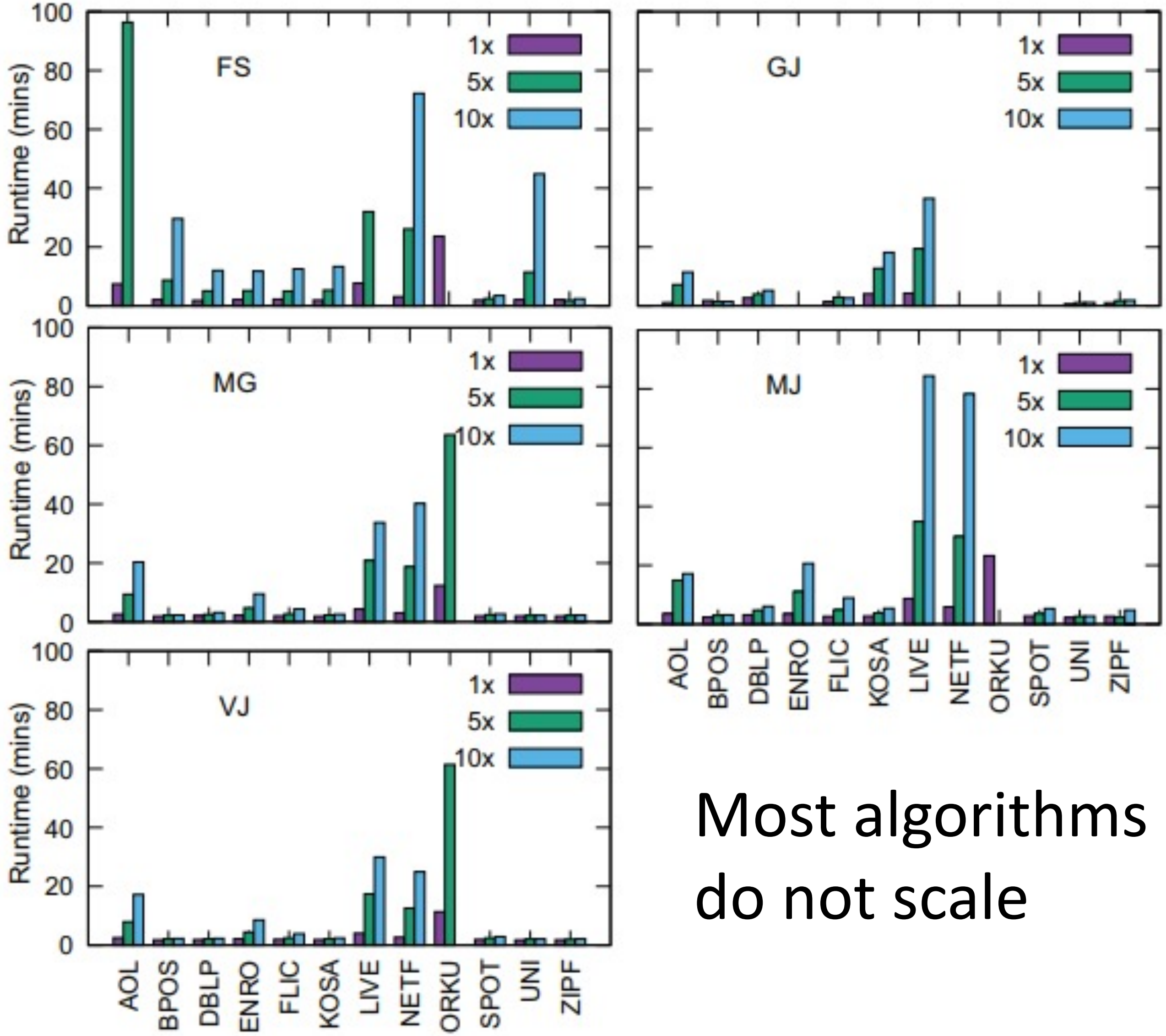
Dataset	Jaccard threshold				
	0.6	0.7	0.8	0.9	0.95
AOL	166 VJ MG	155 VJ MG	84 GJ	68 GJ	64 GJ
BPOS	123 VJ MG	116 VJ MG	101 GJ	101 GJ	106 VJ GJ, MJ
DBLP	342 VJ	174 VJ	129 VJ	112 VJ MG	111 FS VJ
ENRO	323 VJ	230 MG VJ	161 MG FS	130 FS MG VJ	127 FS MG, VJ
FLIC	234 MG VJ	163 MG VJ	119 GJ MG	86 GJ	85 GJ
KOSA	138	121	117	113	112

Insight 2: Bottlenecks





Insight 3: Scalability



Most algorithms do not scale

Summary

- Benchmark with surprising winners
- Scalability issue with all algorithms
- Main reason: straggling reducers
  - High/skewed data replication between map and reduce.
  - Specific characteristics of input data.
  - Adding more nodes does not solve this problem.
- Future:
  - Spark, Flink, ...
  - Remove data-dependency of replication and distribution. Regard restrictions (memory!) of nodes.
- Code at: https://github.com/fabiyon/ssj-dist

Questions?

Datasets

Table 2: Characteristics of the experimental datasets.

Dataset	# recs $\cdot 10^5$	Record length		Universe $\cdot 10^3$		Size (B)
		max	avg	size	maxFreq	
AOL	100	245	3	3900	420	396M
BPOS	3.2	164	9	1.7	240	17M
DBLP	1.0	869	83	6.9	84	41M
ENRO	2.5	3162	135	1100	200	254M
FLIC	12	102	10	810	550	92M
KOSA	6.1	2497	12	41	410	46M
LIVE	31	300	36	7500	1000	873M
NETF	4.8	18000	210	18	230	576M
ORKU	27	40000	120	8700	320	2.5G
SPOT	4.4	12000	13	760	9.7	41M
UNI	1.0	25	10	0.21	18	4.5M
ZIPF	4.4	84	50	100	98	33M

Compared Algorithms

- ClusterJoin (CJ): A. D. Sarma et al.: A similarity joins framework using map-reduce. PVLDB, 2014.
- MRGroupJoin (GJ): D. Deng et al.: An efficient partition based method for exact set similarity joins. PVLDB, 2015.
- FullFilteringJoin (FF): T. Elsayed et al.: Pairwise document similarity in large collections with mapreduce. Assoc. for Comp. Linguistics, 2008.
- MGJoin (MG): C. Rong et al.: Efficient and scalable processing of string similarity join. IEEE TKDE, 2013.
- MassJoin (MJ): D. Deng et al.: Massjoin: A mapreduce-based method for scalable string similarity joins. IEEE ICDE, 2014.
- MRSimJoin (MR): Y. N. Silva et al.: Exploiting mapreduce-based similarity joins. ACM SIGMOD, 2012.
- SSJ-2R (S2): R. Baraglia et al.: Document similarity self-join with mapreduce. IEEE ICDM, 2010.
- VernicaJoin (VJ): R. Vernica et al.: Efficient parallel set-similarity joins using mapreduce. ACM SIGMOD, 2010.
- V-SMART (VS): A. Metwally et al.: V-smart-join: A scalable mapreduce framework for all-pair similarity joins of multisets and vectors. PVLDB, 2012.
- FS-Join (FS): C. Rong et al.: Fast and scalable distributed set similarity joins for big data analytics. IEEE ICDE, 2017.

Experiments Overview

- Hadoop 2.7 on a 12-node cluster
- 10 real-world, 2 synthetic datasets

Experiments:

1. Self-join: all algorithms, all datasets, varying similarity thresholds.
2. Scalability: artificially increase dataset sizes.
3. Varied system parameters, especially memory settings, which determine the number of YARN containers.

Experiment 1: Small Data

		CJ	GJ	FF	MG	MJ	MR	S2	VJ	VS	FS
ClusterJoin	CJ								'=	>	
MRGroupJoin	GJ	>									
FullFilteringJoin	FF		<					<	<		
MGJoin	MG	>	<	>					>		
MassJoin	MJ	>	<	>	<				>		<
MRSimJoin	MR		<		<	<					
SSJ-2R	S2		<		<				>		
VernicaJoin	VJ	>	>	>	>	>	>	>		< >	<
V-SMART	VS		<		<	<			<		<
FS-Join	FS	>	<	>	<	>	>	>	<	>	

← Previous Comparisons →

← Our Comparisons →

Experiment 1: Small Data

Distributed survey:

Dataset	Jaccard threshold 0.6	0.7	0.8	0.9	0.95
AOL	166 VJ MG	155 VJ MG	84 GJ	68 GJ	64 GJ
BPOS	123 VJ MG	116 VJ MG	101 GJ	101 GJ	106 VJ GJ, MJ
DBLP	342 VJ	174 VJ	129 VJ	112 VJ	111 FS

Non-distributed survey:

Dataset	Jaccard threshold 0.5	0.6	0.7	0.75	0.8	0.85	0.9	0.95
AOL	PEL 333 PPJ	PEL 83.7 PPJ GRP	GRP 13.2 ALL PPJ	ALL 8.58 GRP	GRP 4.20 ALL	GRP 1.77	GRP 1.46	GRP 1.43
BMS-POS	PPJ 44.9	PPJ 15.6	PPJ 4.78 ADP GRP	PPJ 2.74 GRP ADP ALL	ALL 1.27 GRP PPJ	ALL 0.447 GRP	GRP 0.170	GRP 0.068

Experiment 3: Varying Parameters

- Findings:
  - Compression used: only good for small datasets; creates overhead for all algorithms on larger datasets
  - Number of reducers: only good for small datasets. On larger datasets, only few algorithms profit from more reducers

VernicaJoin: Basic Ideas

$$\text{prefix}_{\text{Jaccard}}(r, \theta) = |r| - floor(\theta * |r|) + 1$$

Example:

r = {Lorem ipsum dolor sit consetetur sadipscing elitr}

s = {magna aliquyam erat consetetur sadipscing elitr}

$\text{prefix}_{\text{Jaccard}}(r, 0.5) =$ 7 – 3 + 1 = 5

$\text{prefix}_{\text{Jaccard}}(s, 0.5) =$ 6 – 3 + 1 = 4

Build inverted index only over the prefix: result stays complete.

Optimization: use global token order. Sort each set/document in ascending order (already true for this example).