From Dirt to Shovels:
Inferring PADS descriptions from ASCII Data

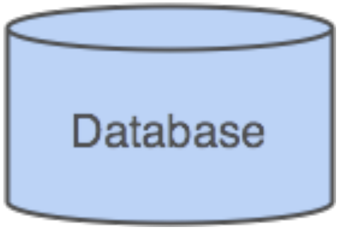
Kathleen Fisher
David Walker
Kenny Zhu
Peter White

March 2008

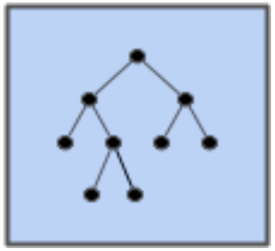
Data,Data,everywhere!

Incredible amounts of data stored in well-behaved formats:

Databases:



XML:

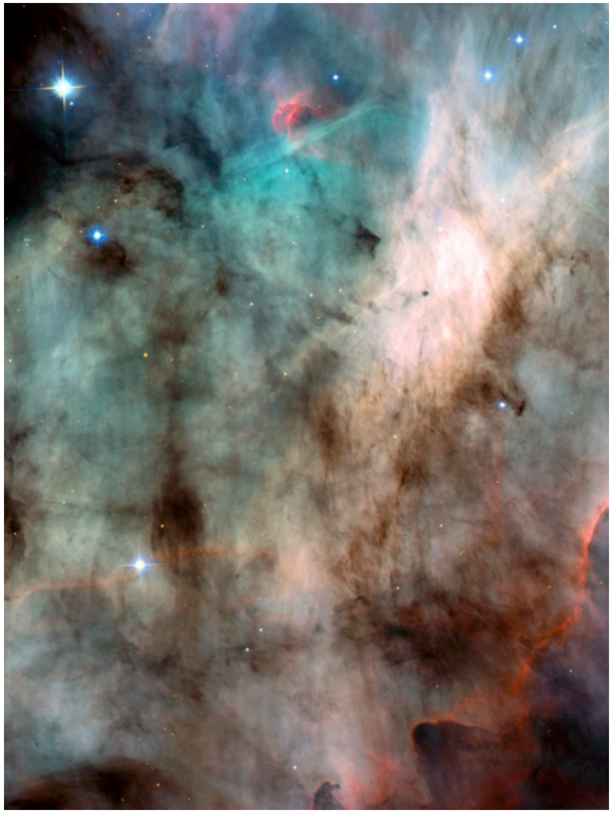


Tools

- Schema
- Browsers
- Query Languages
- Standards
- Libraries
- Books, documentation
- Training courses
- Conversion tools
- Vendor support
- Consultants…

We're not always so lucky!

Vast amounts of chaotic *ad hoc* data:



Tools

- Perl
- Awk
- C
- ...

Government stats

```
"MSN","YYYYMM","Publication Value","Publication Unit","Column Order"
"TEAJBUS",197313,-0.456483,Quadrillion Btu,4
"TEAJBUS",197413,-0.482265,Quadrillion Btu,4
"TEAJBUS",197513,-1.066511,Quadrillion Btu,4
"TEAJBUS",197613,-0.177807,Quadrillion Btu,4
"TEAJBUS",197713,-1.948233,Quadrillion Btu,4
"TEAJBUS",197813,-0.336538,Quadrillion Btu,4
"TEAJBUS",197913,-1.649302,Quadrillion Btu,4
"TEAJBUS",198013,-1.0537,Quadrillion Btu,4
```

Train Stations

```
Southern California Regional Railroad Authority,"Los Angeles, CA",
U,45,46,46,47,49,51,U,45,46,46,47,49,51
Connecticut Department of Transportation ,"New Haven, CT",
U,U,U,U,U,U,8,U,U,U,U,U,U,8
Tri-County Commuter Rail Authority ,"Miami, FL",
U,U,U,U,U,U,18,U,U,U,U,U,U,18
Northeast Illinois Regional Commuter Railroad Corporation,"Chicago, IL",
226,226,226,227,227,227,227,91,104,104,111,115,125,131
Northern Indiana Commuter Transportation District,"Chicago, IL",
18,18,18,18,18,18,20,7,7,7,7,7,7,11
Massachusetts Bay Transportation Authority,"Boston, MA",
U,U,117,119,120,121,124,U,U,67,69,74,75,78
Mass Transit Administration – Maryland DOT ,"Baltimore, MD",
U,U,U,U,U,U,42,U,U,U,U,U,U,22
New Jersey Transit Corporation ,"New York, NY",
158,158,158,162,162,162,167,22,22,41,46,46,46,51
```

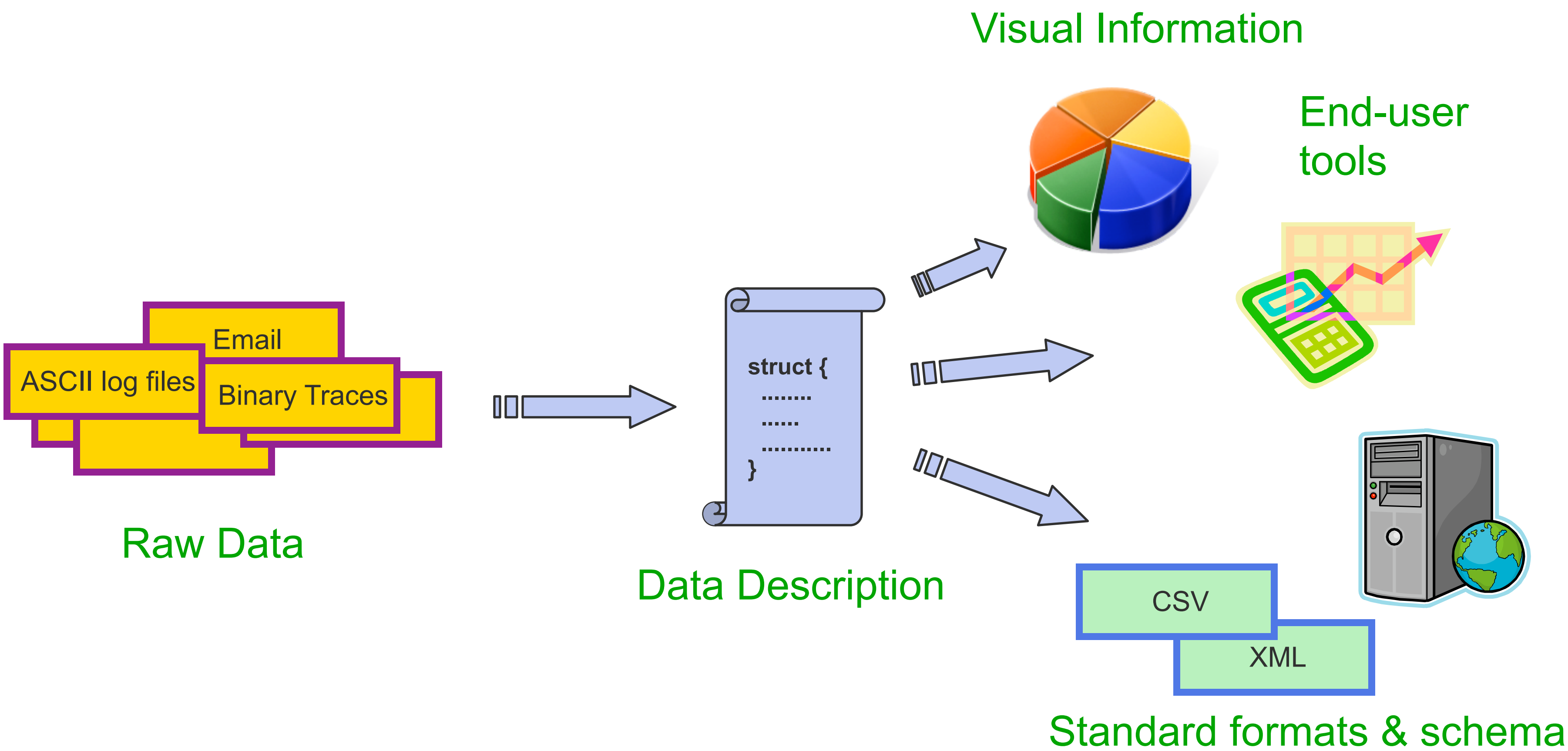
Web logs

```
207.136.97.49 - - [15/Oct/2006:18:46:51 -0700] "GET /turkey/amnty1.gif HTTP/1.0" 200 3013
207.136.97.49 - - [15/Oct/2006:18:46:51 -0700] "GET /turkey/clear.gif HTTP/1.0" 200 76
207.136.97.49 - - [15/Oct/2006:18:46:52 -0700] "GET /turkey/back.gif HTTP/1.0" 200 224
207.136.97.49 - - [15/Oct/2006:18:46:52 -0700] "GET /turkey/women.html HTTP/1.0" 200 17534
208.196.124.26 - Dbuser [15/Oct/2006:18:46:55 -0700] "GET /candatop.html HTTP/1.0" 200 -
208.196.124.26 - - [15/Oct/2006:18:46:57 -0700] "GET /images/done.gif HTTP/1.0" 200 4785
www.att.com - - [15/Oct/2006:18:47:01 -0700] "GET /images/reddash2.gif HTTP/1.0" 200 237
208.196.124.26 - - [15/Oct/2006:18:47:02 -0700] "POST /images/refrun1.gif HTTP/1.0" 200 836
208.196.124.26 - - [15/Oct/2006:18:47:05 -0700] "GET /images/hasene2.gif HTTP/1.0" 200 8833
www.cnn.com - - [15/Oct/2006:18:47:08 -0700] "GET /images/candalog.gif HTTP/1.0" 200 -
208.196.124.26 - - [15/Oct/2006:18:47:09 -0700] "GET /images/nigpost1.gif HTTP/1.0" 200 4429
208.196.124.26 - - [15/Oct/2006:18:47:09 -0700] "GET /images/rally4.jpg HTTP/1.0" 200 7352
128.200.68.71 - - [15/Oct/2006:18:47:11 -0700] "GET /amnesty/usalinks.html HTTP/1.0" 143 10329
208.196.124.26 - - [15/Oct/2006:18:47:11 -0700] "GET /images/reyes.gif HTTP/1.0" 200 10859
```

And many others...

- Gene ontology data
- Cosmology data
- Financial trading data
- Telecom billing data
- Router config files
- System logs
- Call detail data
- Netflow packets
- DNS packets
- Java JAR files
- Jazz recording info
- ...

Learning: Goals & Approach



Problem: Producing useful tools for ad hoc data takes a lot of time.
Solution: A learning system to generate data descriptions and tools automatically.

PADS Reminder

Inferred data formats are described using a specialized language of types

- Provides rich base type library; many specialized for systems data.
  - `Pint8, Puint8, …                 // -123, 44`

    `Pstring(:'|':)                   // hello|`

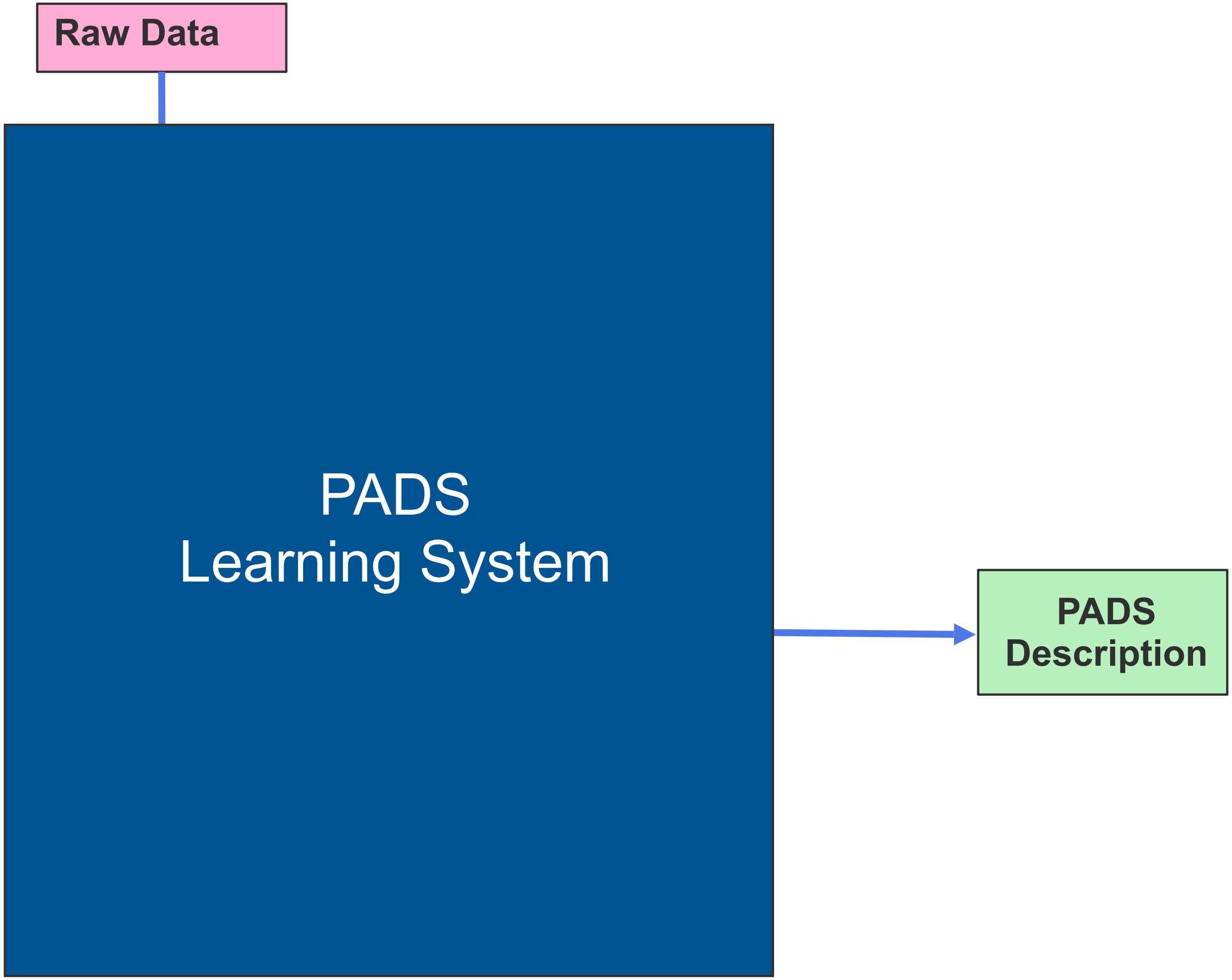
    `Pstring_FW(:3:)                  // catdog`

    `Pdate, Ptime, Pip, …`
- Provides type constructors to describe data source structure:
  - sequences: `Pstruct, Parray,`
  - choices: `Punion, Penum, Pswitch`
  - constraints: allow arbitrary predicates to describe expected properties.

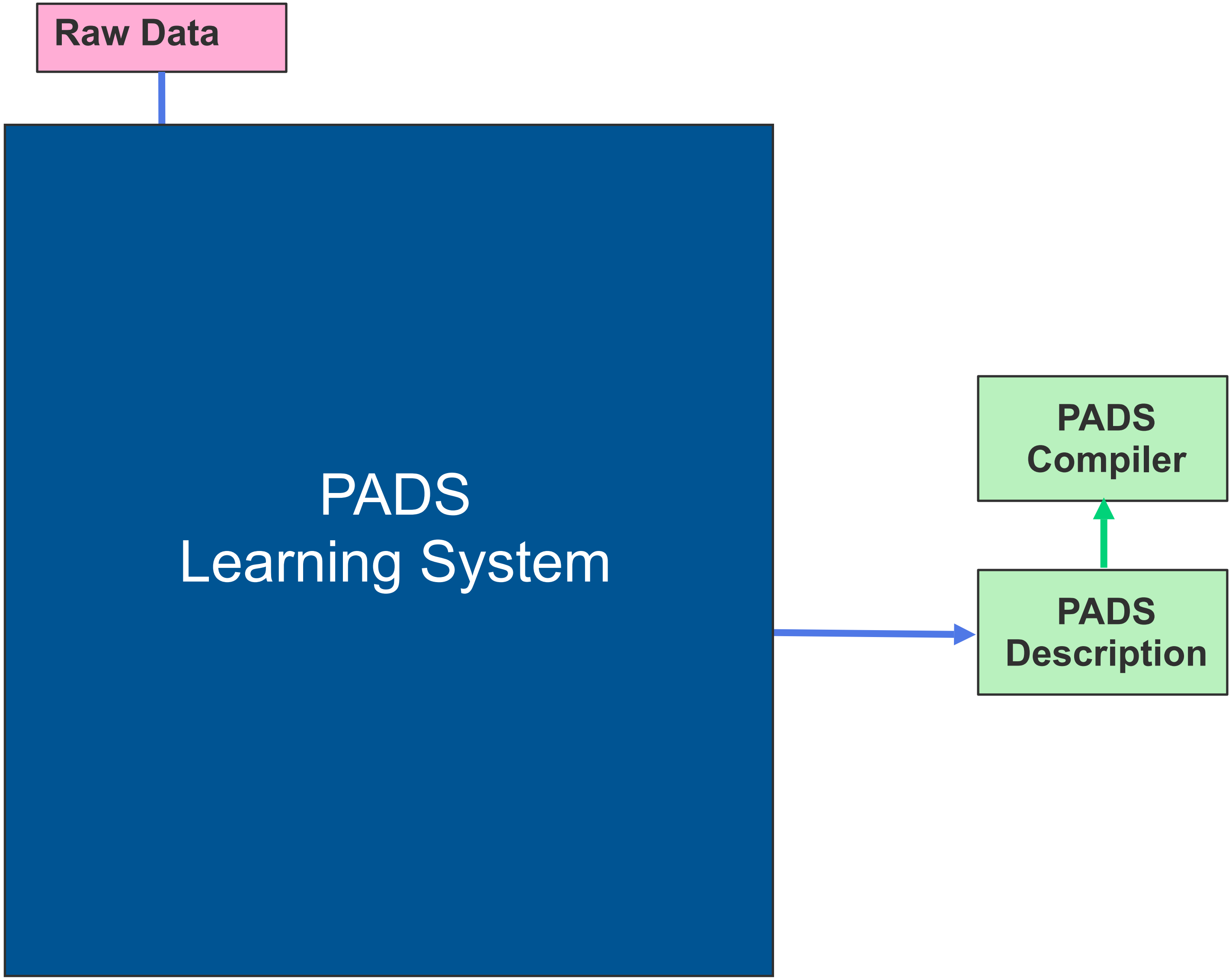
PADS compiler generates stand-alone tools including xml-conversion, Xquery support & statistical analysis directly from data descriptions.

Go to demo

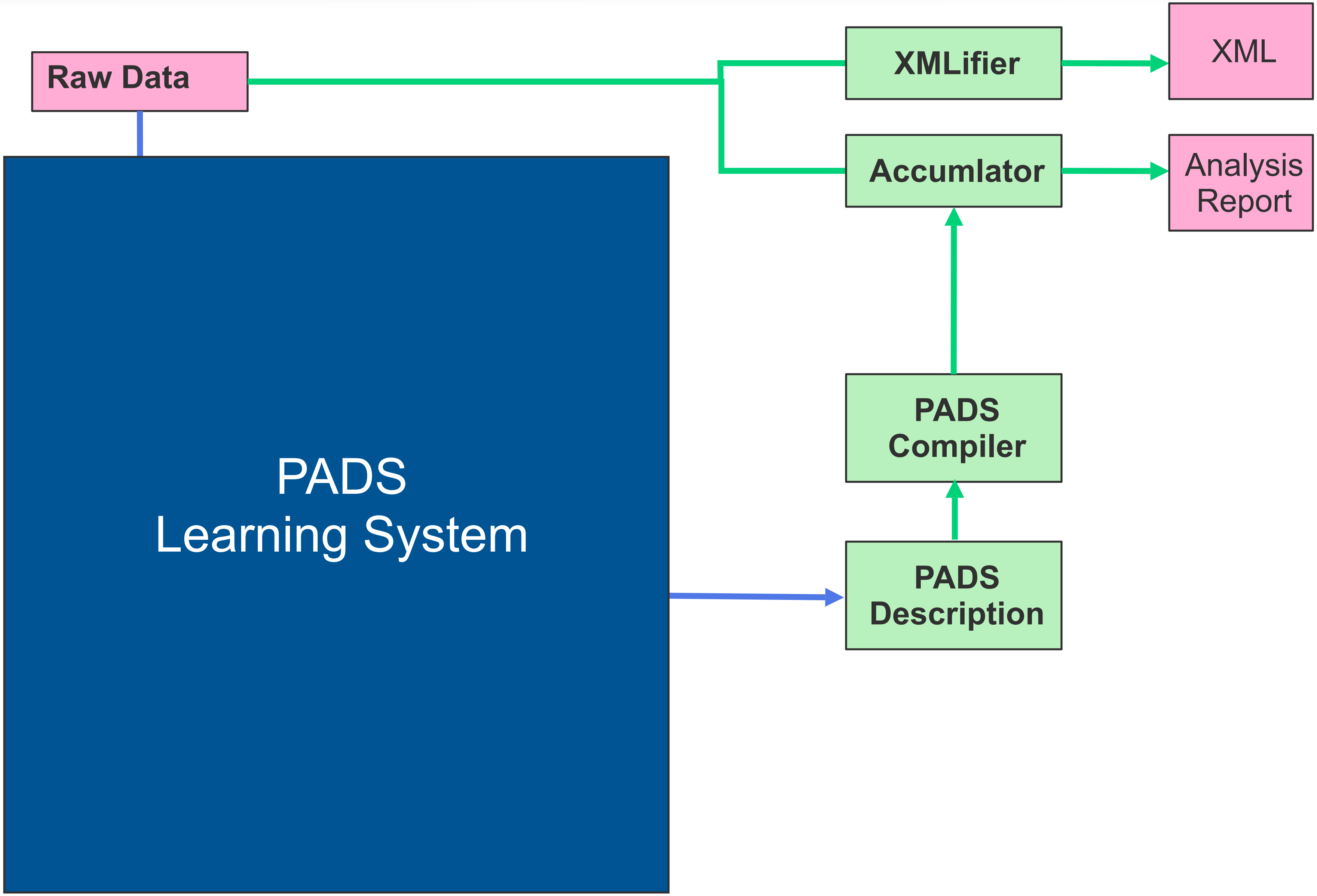
Format inference overview



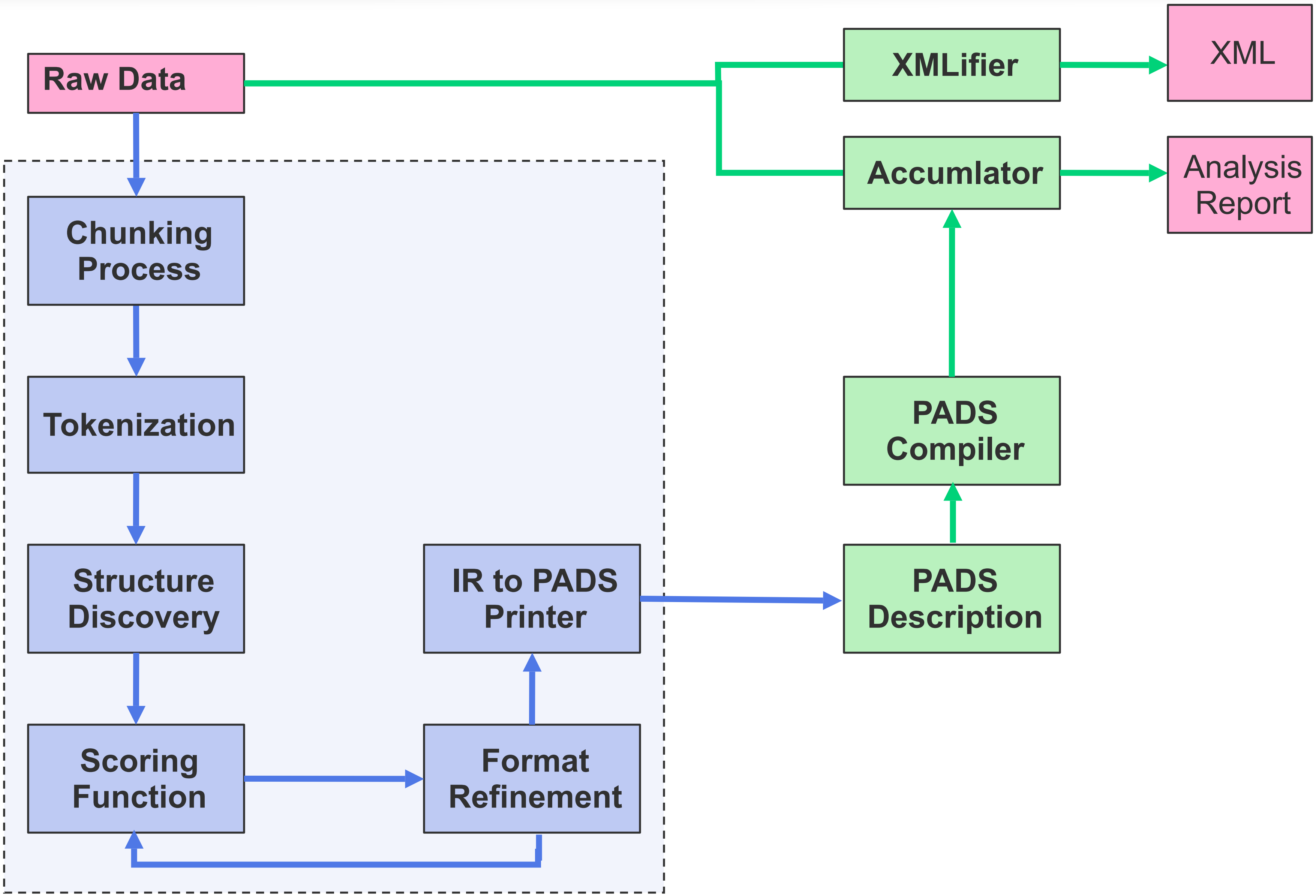
Format inference overview



Format inference overview

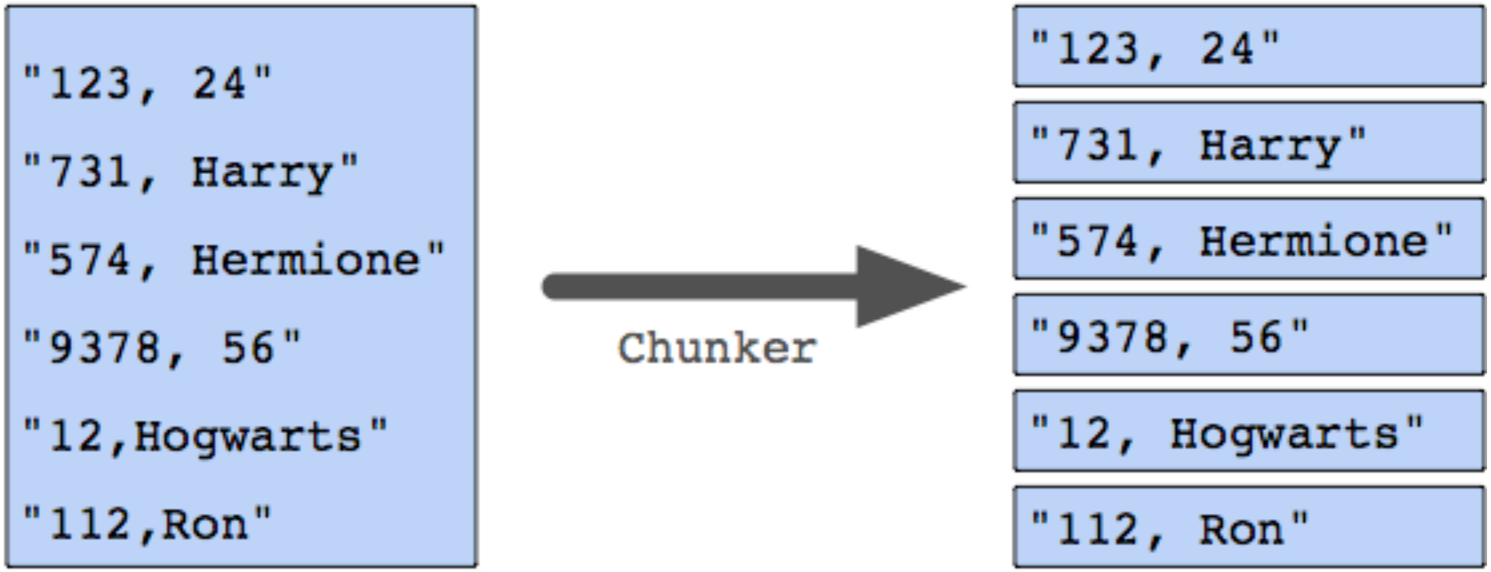


Format inference overview



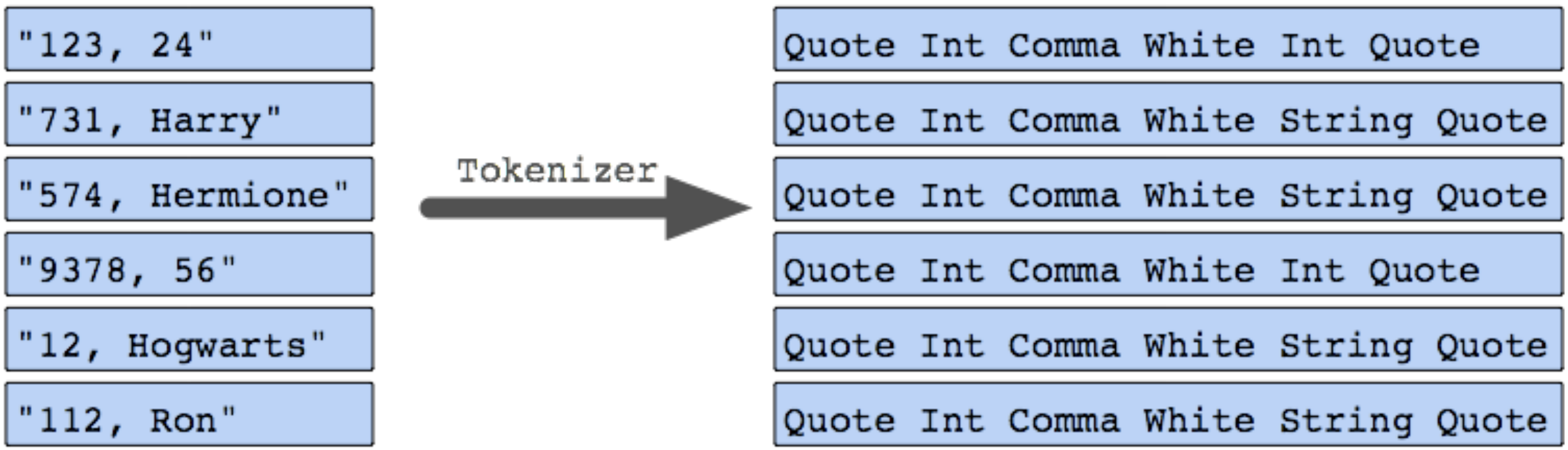
Chunking Process

- Convert raw input into sequence of "chunks."



- Supported divisions:
  - Various forms of "newline"
  - File boundaries
- Also possible: user-defined "paragraphs"

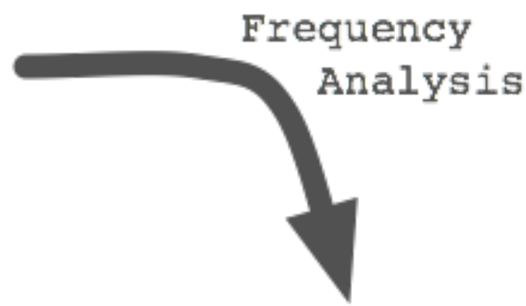
Tokenization

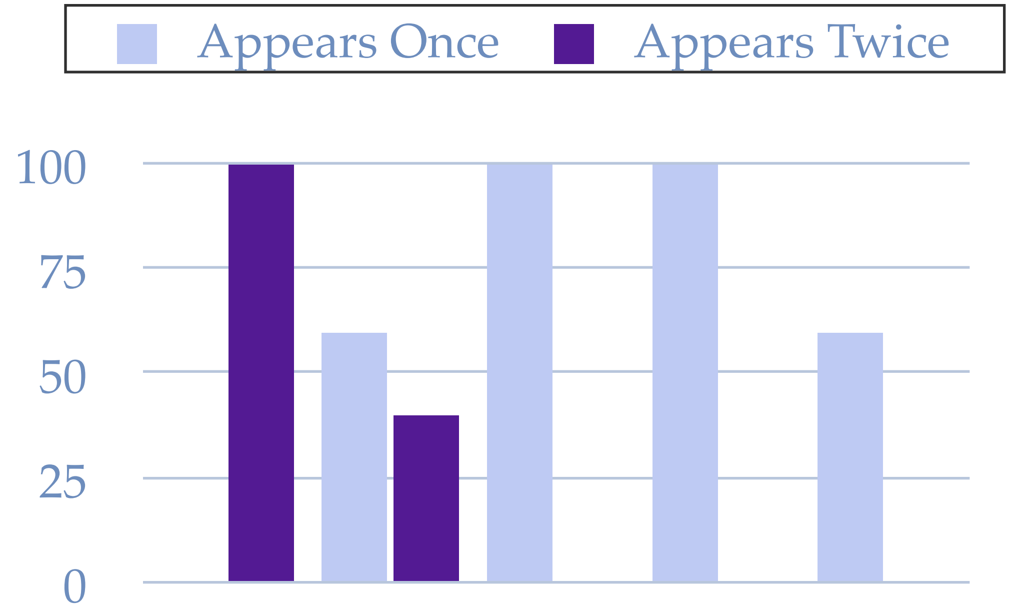


- Tokens expressed as regular expressions.
- **Basic tokens**
    - Integer, white space, punctuation, strings
- **Distinctive tokens**
    - IP addresses, dates, times, MAC addresses, ...

Histograms

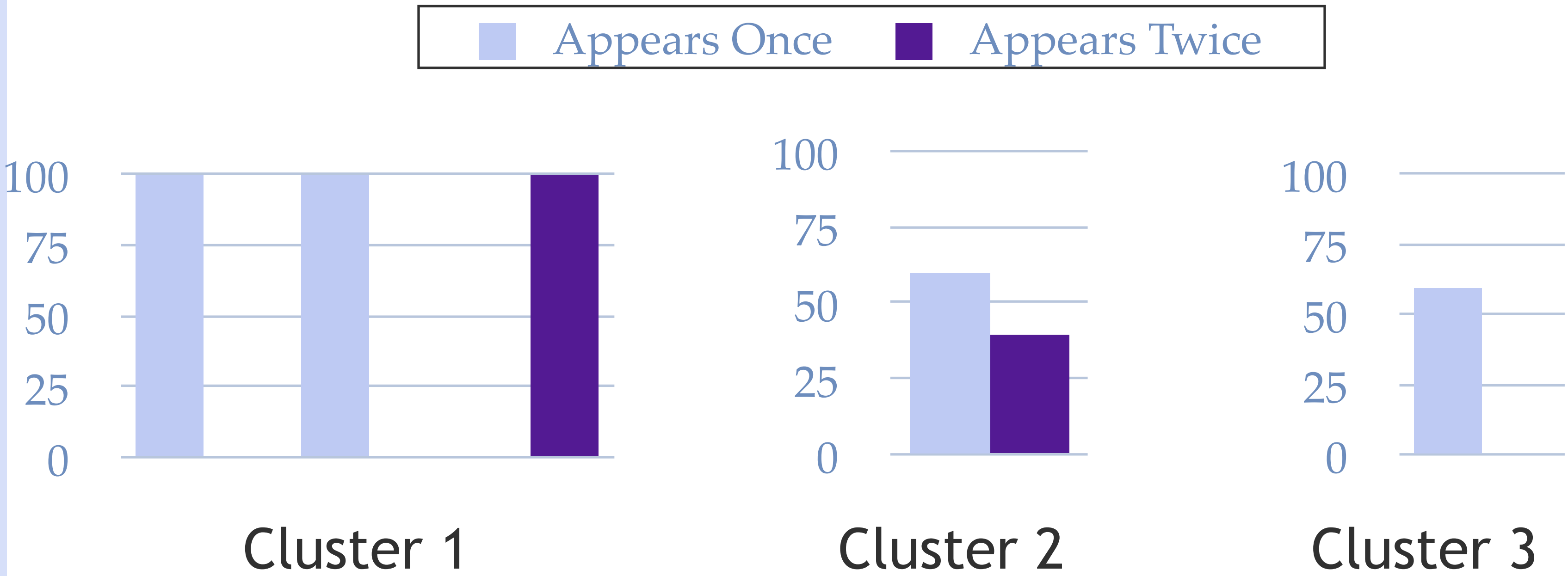
```
Quote Int Comma White Int Quote
Quote Int Comma White String Quote
Quote Int Comma White String Quote
Quote Int Comma White Int Quote
Quote Int Comma White String Quote
Quote Int Comma White String Quote
```





Clustering

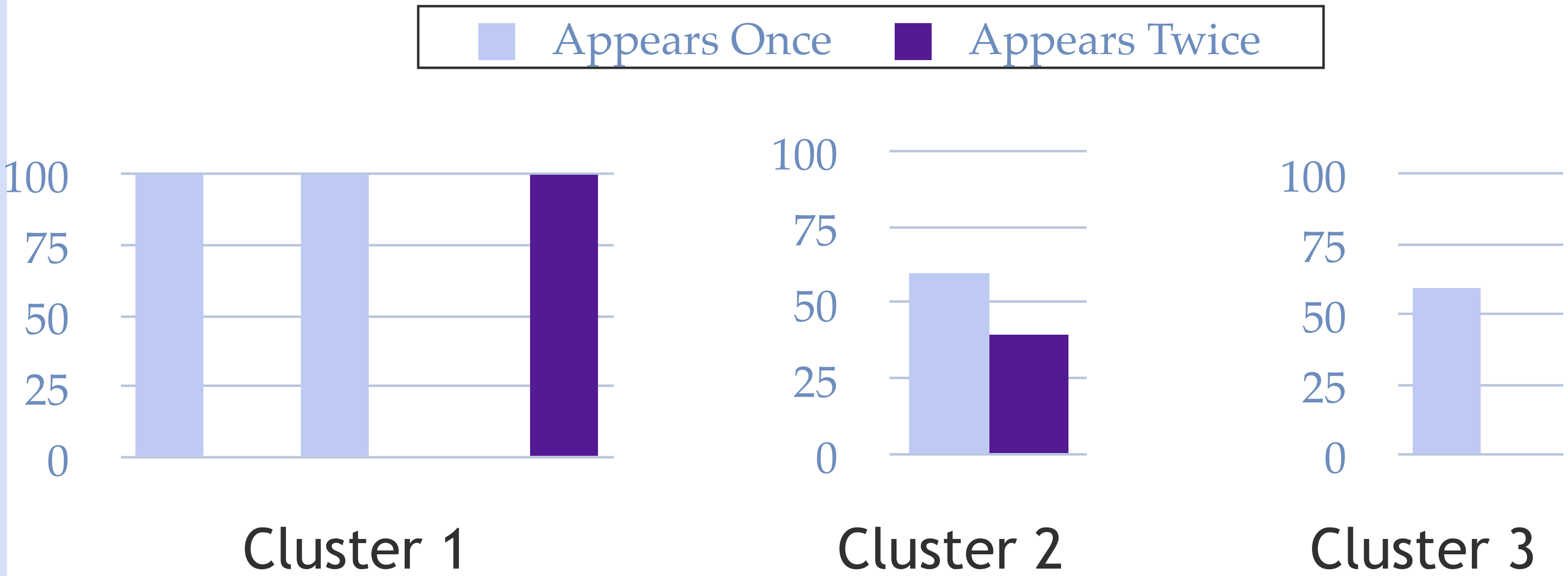
Group clusters with similar frequency distributions



Two frequency distributions are *similar* if they have the same shape (within some error tolerance) when the columns are sorted by height.

Clustering

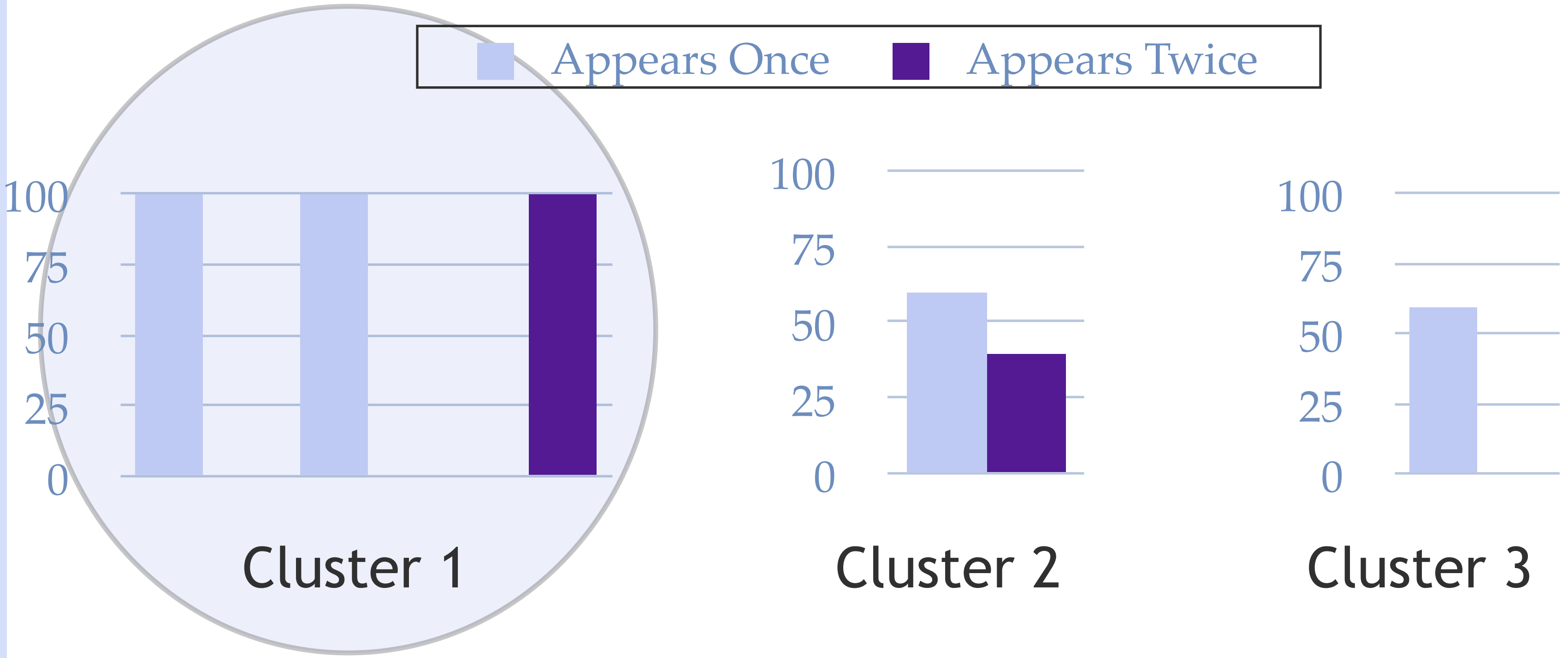
Group clusters with similar frequency distributions



Rank clusters by metric that rewards *high coverage* and *narrower* distributions. Chose cluster with highest score.

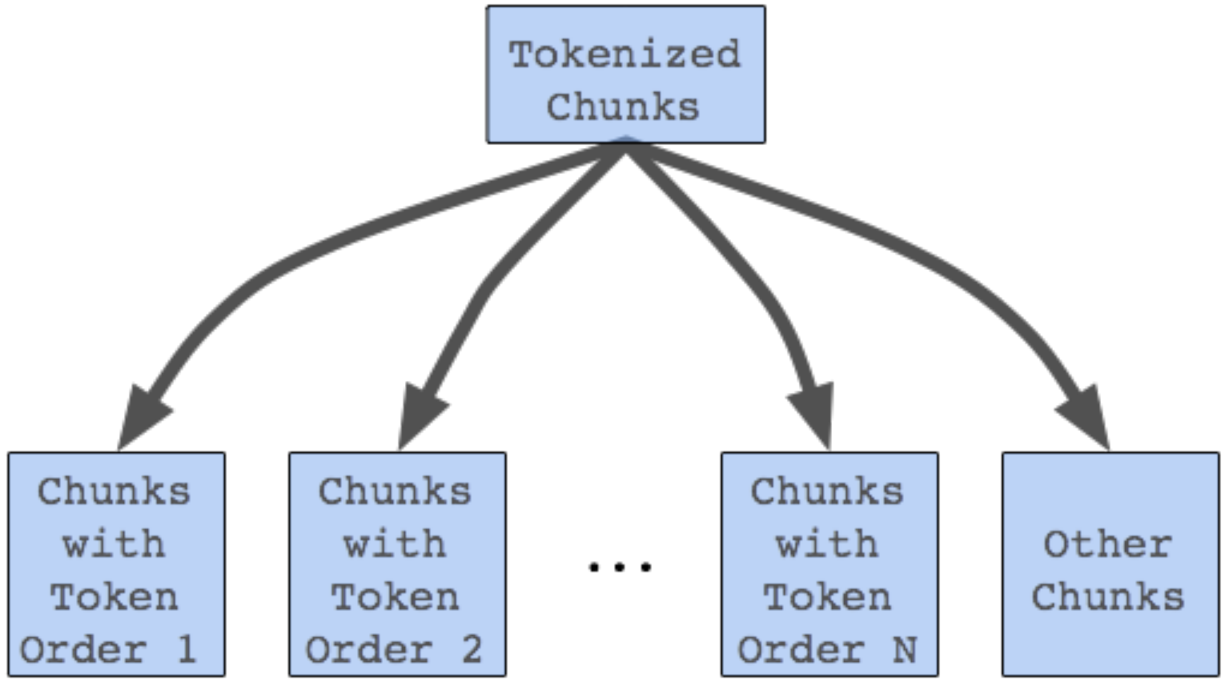
Clustering

Group clusters with similar frequency distributions



Rank clusters by metric that rewards *high coverage* and *narrower* distributions. Chose cluster with highest score.

Partition chunks



$$\tau_1 + \tau_2 \ldots + \ldots \tau_N + \tau_{other}$$

In our example, all the tokens appear in the same order in all chunks, so the union is degenerate.

Find subcontexts

Quote Int **Comma** **White** Int **Quote**

Quote Int **Comma** **White** String **Quote**

Quote Int **Comma** **White** String **Quote**

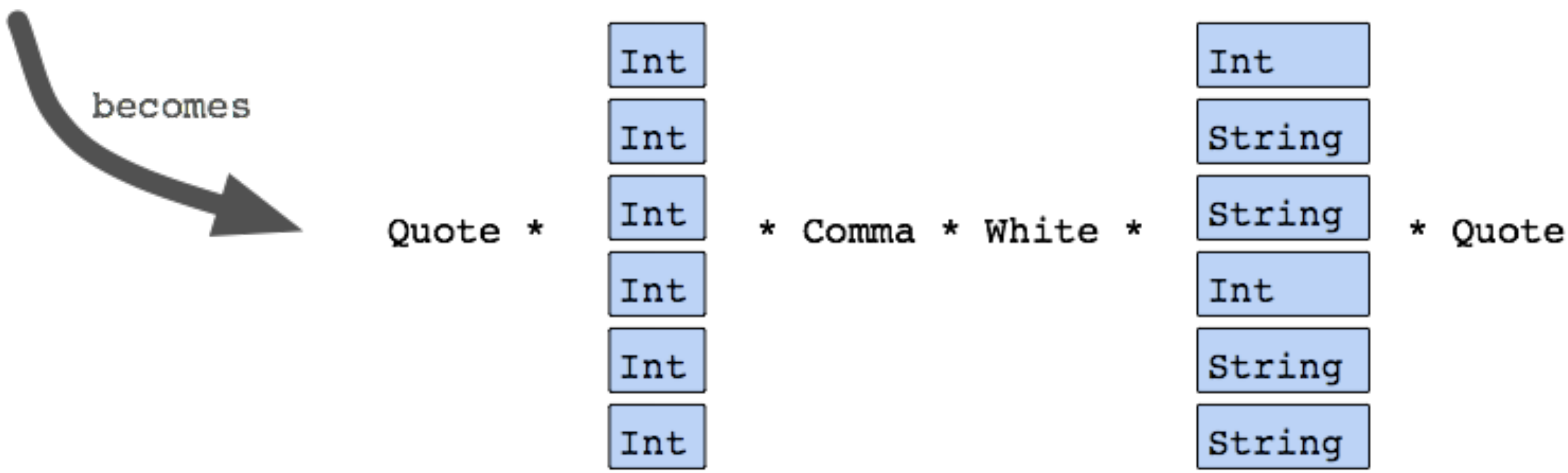
Quote Int **Comma** **White** Int **Quote**

Quote Int **Comma** **White** String **Quote**

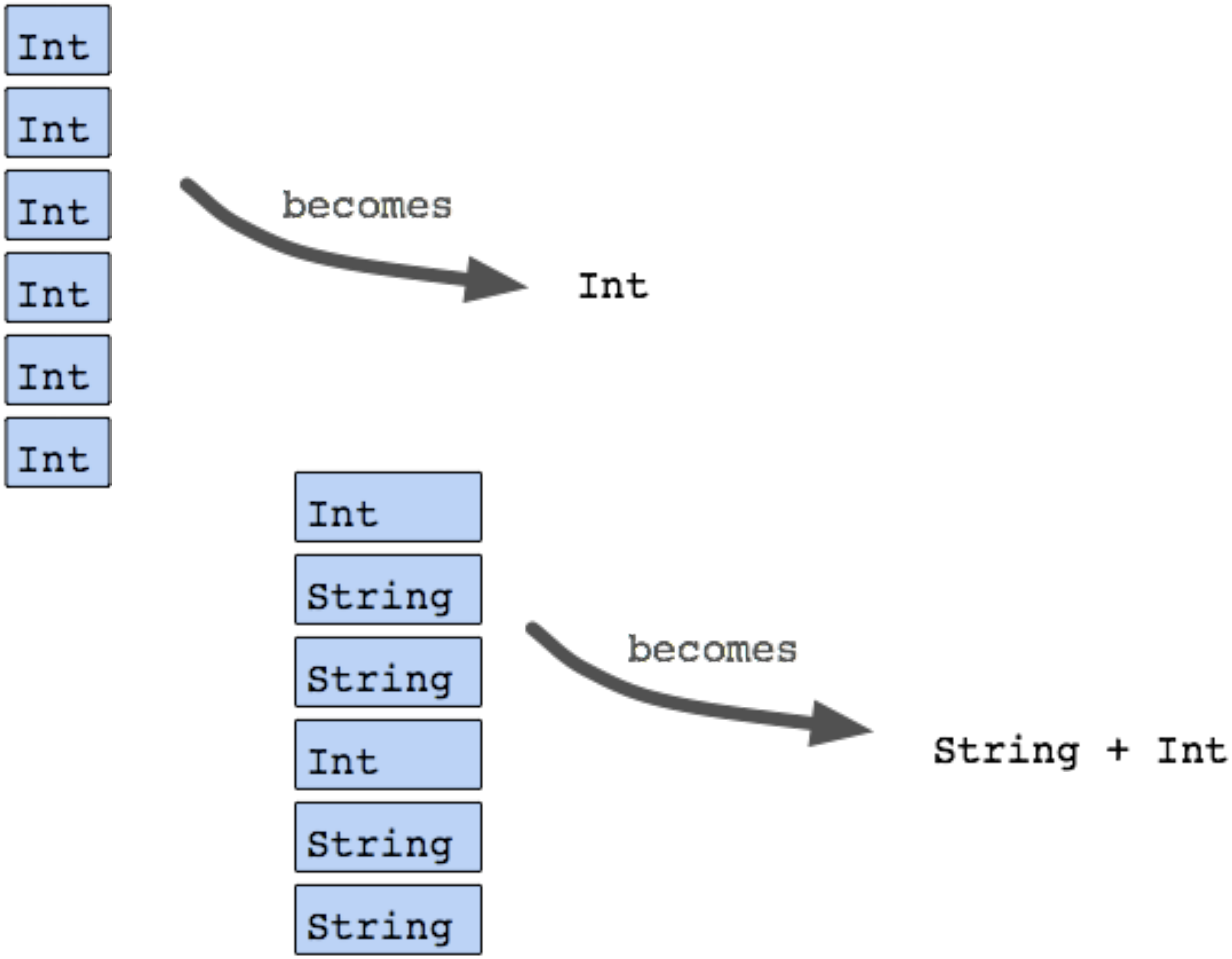
Quote Int **Comma** **White** String **Quote**

Tokens in selected cluster:

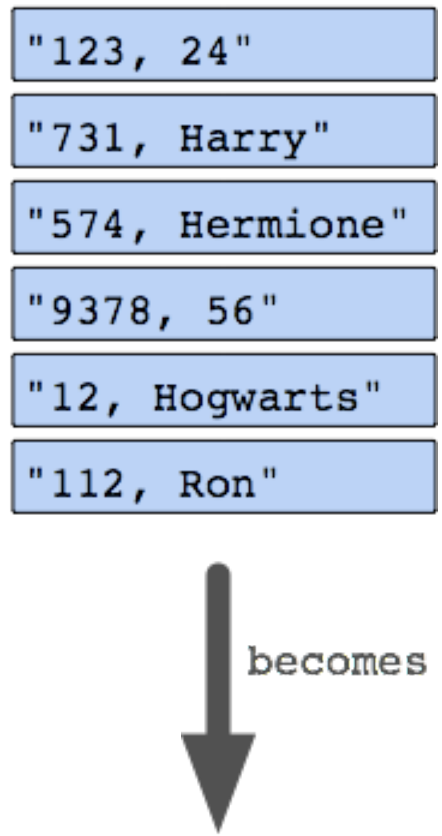
Quote(2) Comma White



Then Recurse...



Inferred type

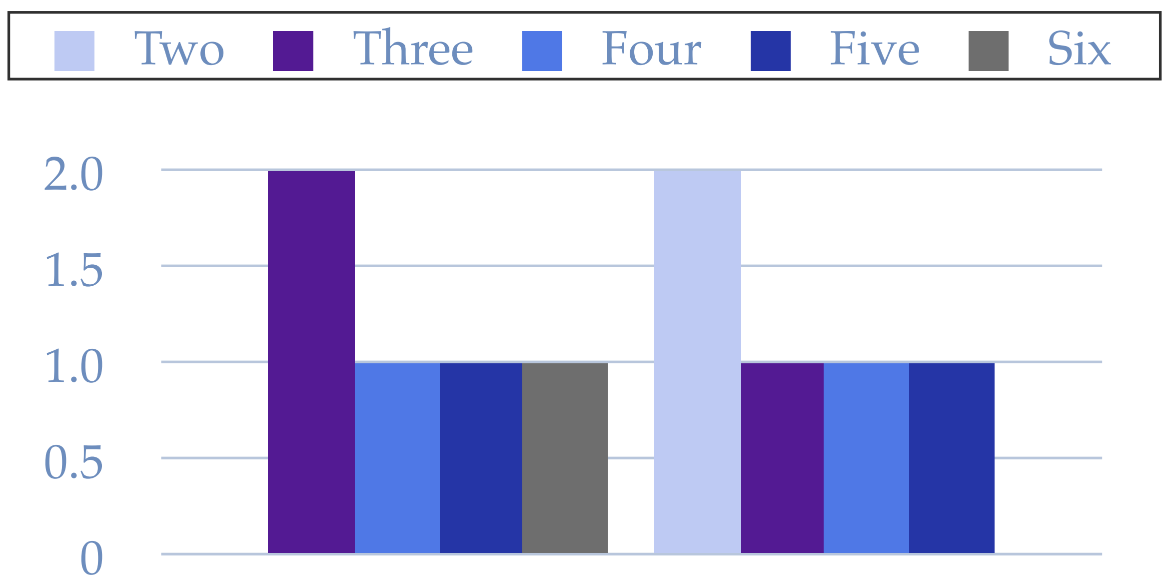



```
Quote * Int * Comma * White * (String + Int) * Quote
```

Finding arrays

```
hermione|ginny|lavender
malfoy|crabbe|goyle|parkinson|bulstrode|greengrass|nott|zabini
harry|ron|nevil|george|fred
trevor|ginger|hedwig
flitwick|mcgonagall|snape|sprout
quirrell|lockhart|lupin|moody|umbridge|snape
```

Single cluster with high coverage, but *wide* distribution.



Partitioning

Selected tokens for array cluster: `String Pipe`

hermione| ginny| lavender

malfoy| crabbe| goyle| parkinson| bulstrode| … zabini

harry| ron| nevil| george| fred

trevor| ginger| hedwig

flitwick| mcgonagall| snape| sprout

quirrell| lockhart| lupin| moody| umbridge| snape

Partitioning

Selected tokens for array cluster: `String Pipe`

```
hermione| ginny| lavender
malfoy| crabbe| goyle| parkinson| bulstrode| ... zabini
harry| ron| nevil| george| fred
trevor| ginger| hedwig
flitwick| mcgonagall| snape| sprout
quirrell| lockhart| lupin| moody| umbridge| snape
```

Context 1,2:

`String * Pipe`

Partitioning

Selected tokens for array cluster: `String Pipe`

```
hermione|  ginny|  lavender
malfoy|  crabbe|  goyle|  parkinson|  bulstrode|  ...  zabini
harry|  ron|  nevil|  george|  fred
trevor|  ginger|  hedwig
flitwick|  mcgonagall|  snape|  sprout
quirrell|  lockhart|  lupin|  moody|  umbridge|  snape
```

Context 1,2:

`String * Pipe`

Context 3: `String`

Partitioning

Selected tokens for array cluster: `String Pipe`

```
hermione| ginny| lavender
malfoy| crabbe| goyle| parkinson| bulstrode| ... zabini
harry| ron| nevil| george| fred
trevor| ginger| hedwig
flitwick| mcgonagall| snape| sprout
quirrell| lockhart| lupin| moody| umbridge| snape
```

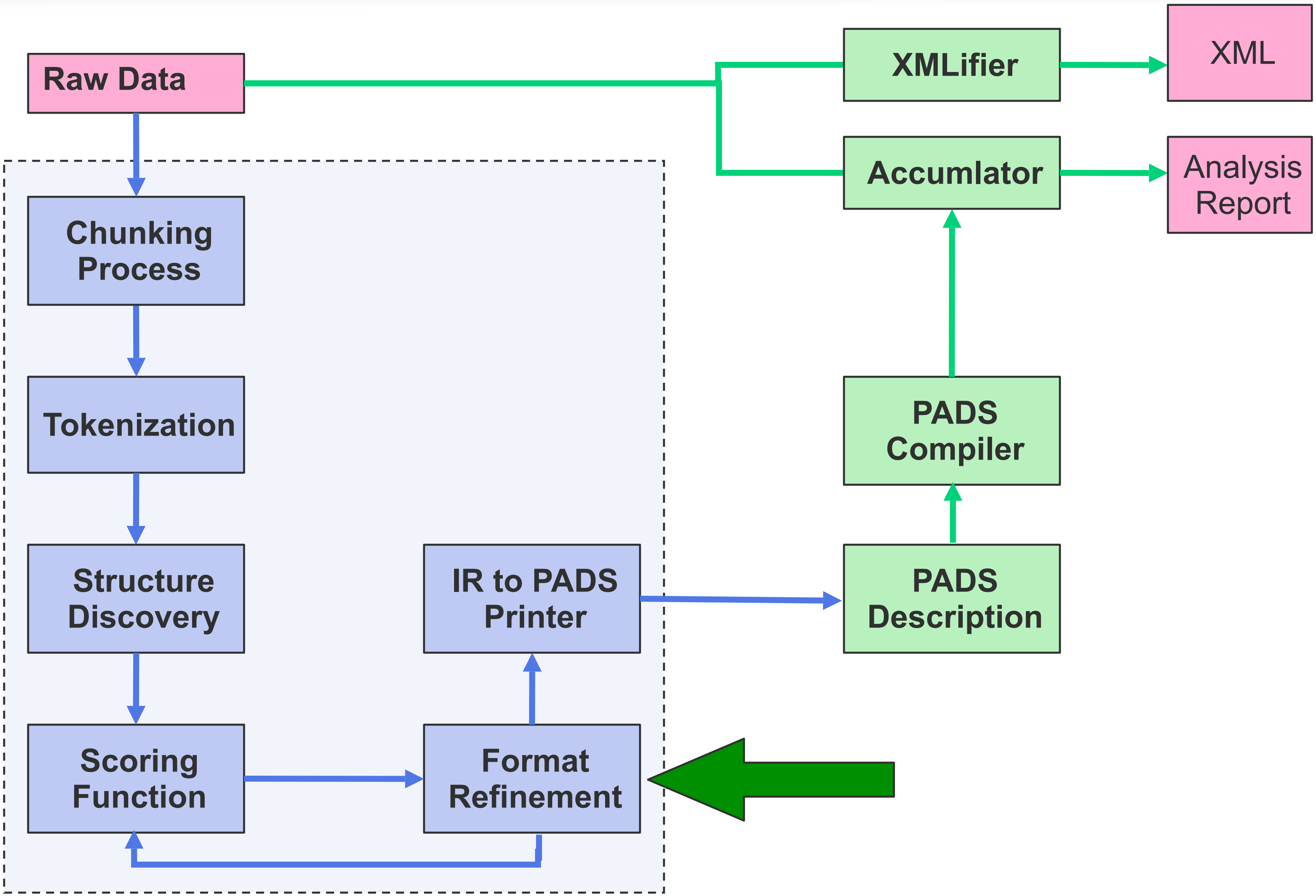
Context 1,2:
`String * Pipe`

becomes

Context 3: `String`

`String [] sep('|')`

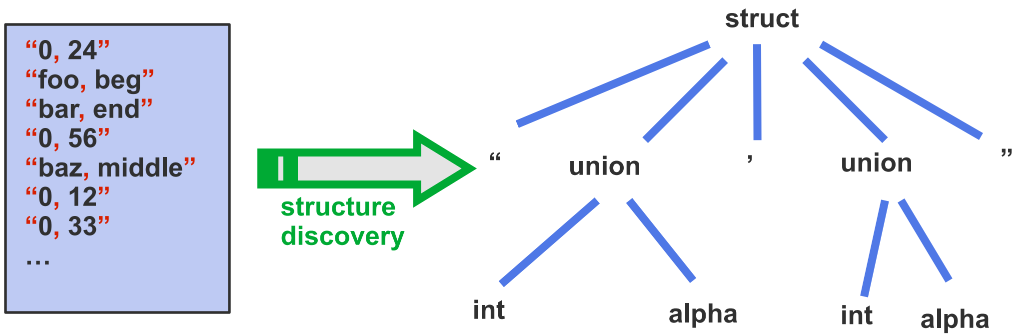
Format inference overview

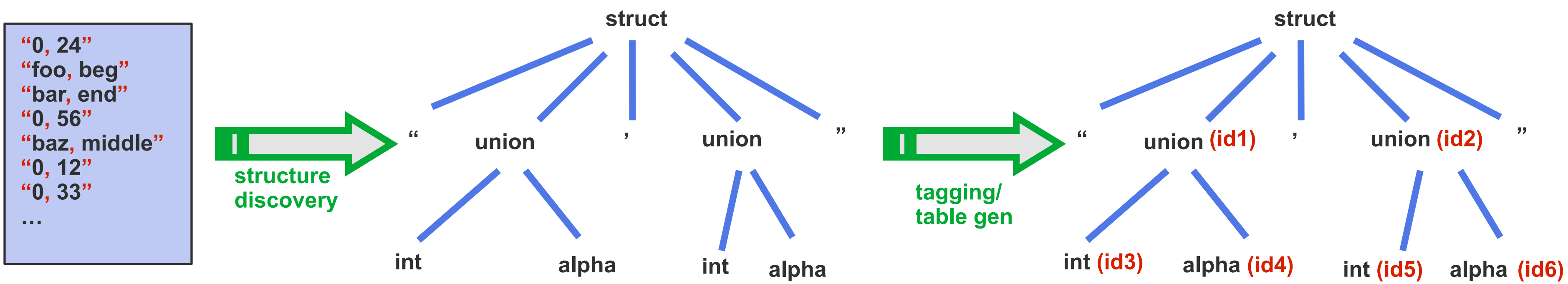


Format Refinement: Rewriting

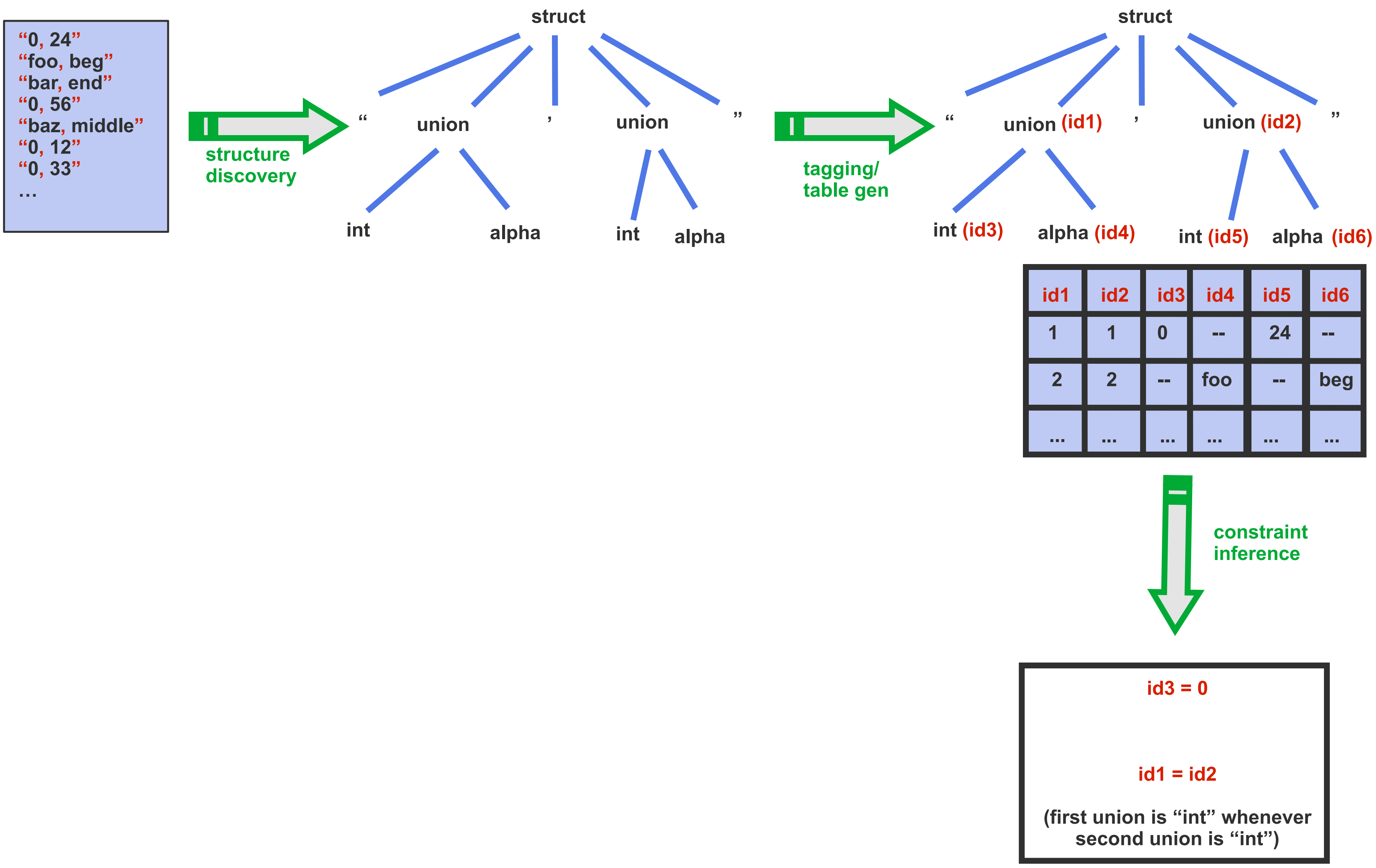
- Optimize information-theoretic complexity
  - Simplify presentation
    - Merge adjacent structures and unions
  - Improve precision
    - Identify constant values
    - Introduce enumerations and dependencies
- Refine types
  - Termination conditions for strings
  - Integer sizes
  - Identify array element separators & terminators

“0, 24”
“foo, beg”
“bar, end”
“0, 56”
“baz, middle”
“0, 12”
“0, 33”
…

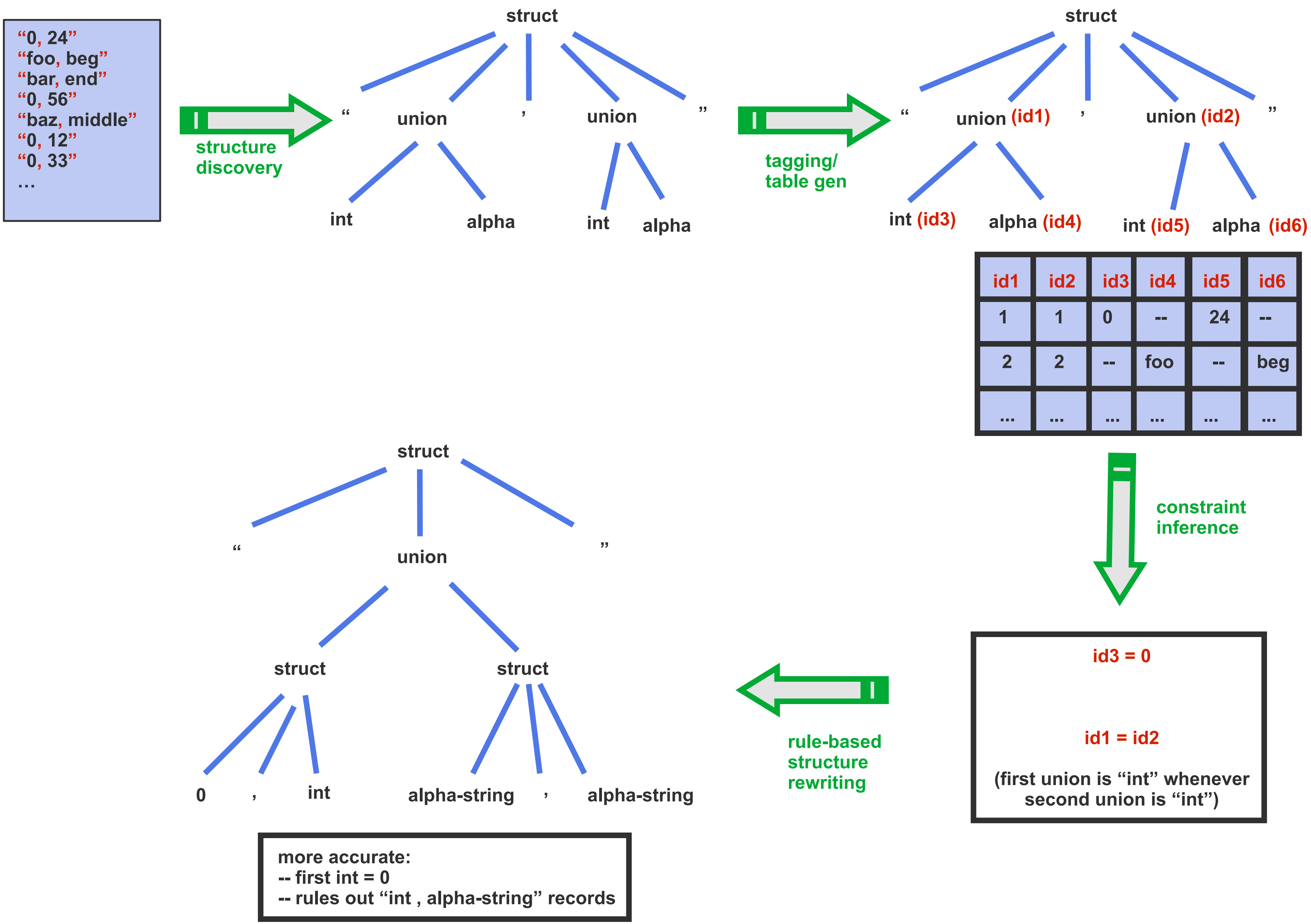

"0, 24"
"foo, beg"
"bar, end"
"0, 56"
"baz, middle"
"0, 12"
"0, 33"
...
structure
discovery
struct
“
union
,
union
”
int
alpha
int
alpha



id1	id2	id3	id4	id5	id6
1	1	0	--	24	--
2	2	--	foo	--	beg
...	...	...	...	...	...

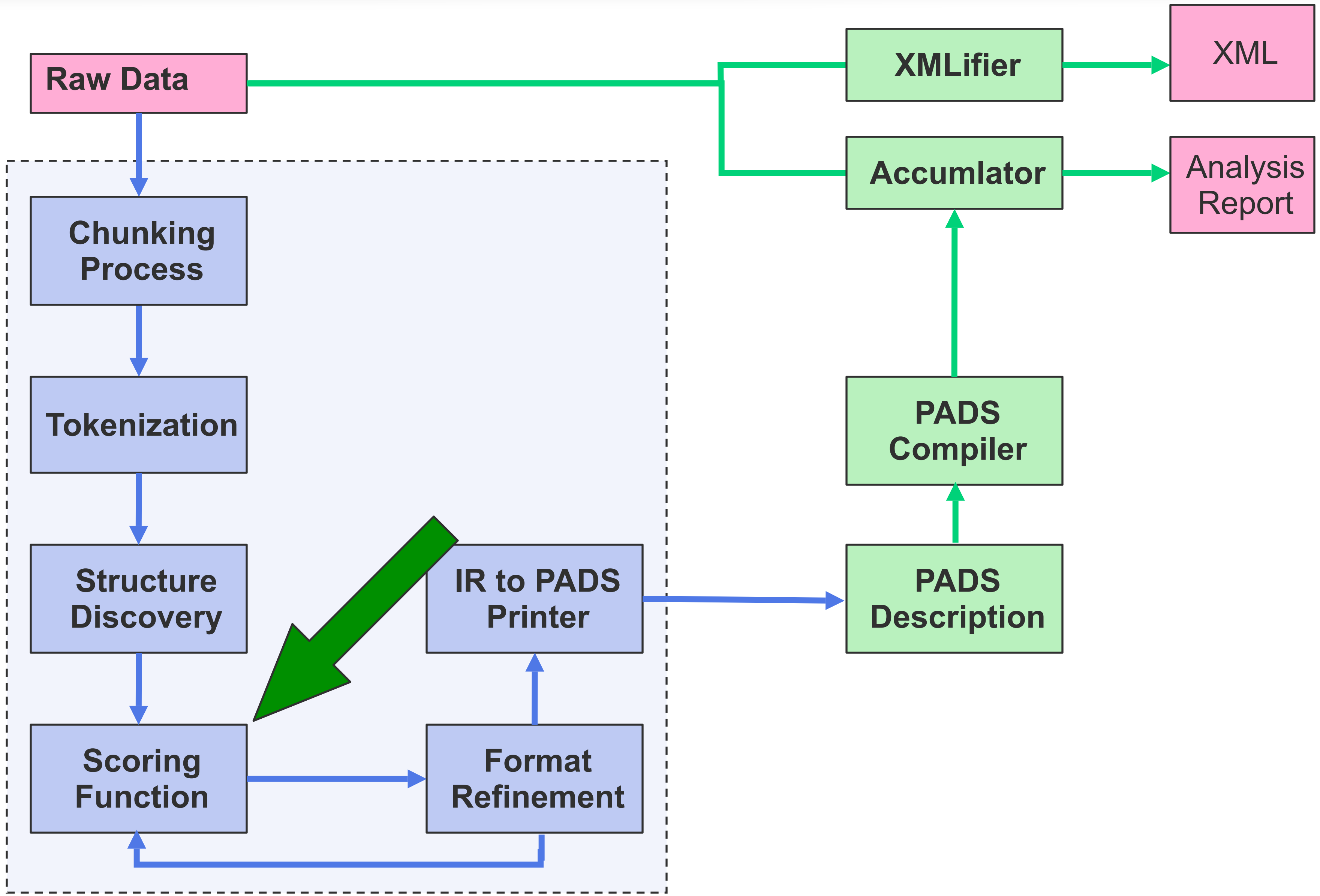


"0, 24"
"foo, beg"
"bar, end"
"0, 56"
"baz, middle"
"0, 12"
"0, 33"
...
structure discovery
struct
"
union
,
union
"
int
alpha
int
alpha
tagging/ table gen
struct
"
union (id1)
,
union (id2)
"
int (id3)
alpha (id4)
int (id5)
alpha (id6)
id1 id2 id3 id4 id5 id6
1 1 0 -- 24 --
2 2 -- foo -- beg
... ...
constraint inference
id3 = 0
id1 = id2
(first union is "int" whenever second union is "int")



"0, 24"
"foo, beg"
"bar, end"
"0, 56"
"baz, middle"
"0, 12"
"0, 33"
…
structure discovery
struct
"
union
,
union
"
int
alpha
int
alpha
tagging/ table gen
struct
"
union (id1)
,
union (id2)
"
int (id3)
alpha (id4)
int (id5)
alpha (id6)
id1 id2 id3 id4 id5 id6
1 1 0 -- 24 --
2 2 -- foo -- beg
... ...
constraint inference
id3 = 0
id1 = id2
(first union is "int" whenever second union is "int")
rule-based structure rewriting
struct
"
union
"
struct
struct
0
,
int
alpha-string
,
alpha-string
more accurate:
-- first int = 0
-- rules out "int , alpha-string" records

Format inference overview



Scoring

- Goal: A quantitative metric to evaluate the quality of inferred descriptions and drive refinement.
- Challenges:
  - *Underfitting*. Pstring(Peof) describes data, but is too general to be useful.
  - *Overfitting*. Type that exhaustively describes data ('H', 'e', 'r', 'm', 'i', 'o', 'n', 'e',…) is too precise to be useful.
- Sweet spot: Reward compact descriptions that predict the data well.

Minimum Description Length

- Standard metric from machine learning.
- Cost of transmitting the syntax of a description plus the cost of transmitting the data given the description:

```
cost(T,d) =
    complexity(T) + complexity(d|T)
```

- Functions defined inductively over the structure of the type T and data d respectively.
- Normalized MDL gives compression factor.
- Scoring function triggers rewriting rules.

Testing and Evaluation

- Evaluated overall results qualitatively
  - Compared with Excel -- a manual process with limited facilities for representation of hierarchy or variation
  - Compared with hand-written descriptions –- performance variable depending on tokenization choices & complexity
- Evaluated accuracy quantitatively
  - Implemented infrastructure to use generated accumulator programs to determine inferred description error rates
- Evaluated performance quantitatively
  - Tokenization & rough structure inference perform well: less than 1 second on 300K
  - Dependency analysis can take a long time on complex format (but can be cut down easily).

Benchmark Formats

Data source	Chunks	Bytes	Description
1967Transactions.short	999	70929	Transaction records
MER_T01_01.cvs	491	21731	Comma-separated records
Ai.3000	3000	293460	Web server log
Asl.log	1500	279600	Log file of MAC ASL
Boot.log	262	16241	Mac OS boot log
Crashreporter.log	441	50152	Original crashreporter daemon log
Crashreporter.log.mod	441	49255	Modified crashreporter daemon log
Sirius.1000	999	142607	AT&T phone provision data
Ls-l.txt	35	1979	Command ls -l output
Netstat-an	202	14355	Output from netstat -an
Page_log	354	28170	Printer log from CUPS
quarterlypersonalincome	62	10177	Spread sheet
Railroad.txt	67	6218	US Rail road info
Scrollkeeper.log	671	66288	Application log
Windowserver_last.log	680	52394	Log from Mac LoginWindow server
Yum.txt	328	18221	Log from package installer Yum

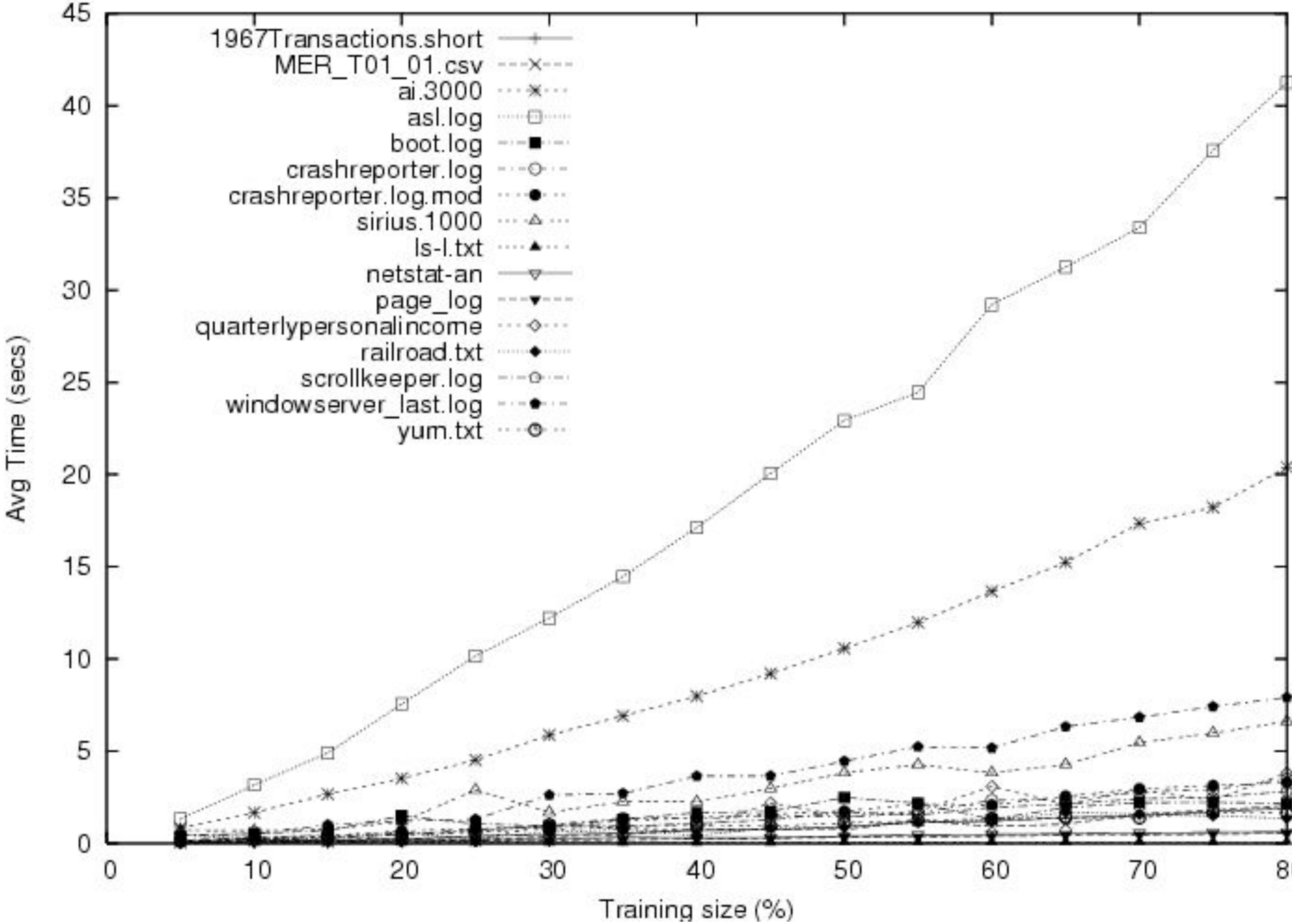
Execution Times

Data source	SD (s)	Ref (s)	Tot (s)	HW (h)
1967Transactions.short	0.20	2.32	2.56	4.0
MER_T01_01.cvs	0.11	2.82	2.92	0.5
Ai.3000	1.97	26.35	28.64	1.0
Asl.log	2.90	52.07	55.26	1.0
Boot.log	0.11	2.40	2.53	1.0
Crashreporter.log	0.12	3.58	3.73	2.0
Crashreporter.log.mod	0.15	3.83	4.00	2.0
Sirius.1000	2.24	5.69	8.00	1.5
Ls-l.txt	0.01	0.10	0.11	1.0
Netstat-an	0.07	0.74	0.82	1.0
Page_log	0.08	0.55	0.65	0.5
quarterlypersonalincome	0.07	5.11	5.18	48
Railroad.txt	0.06	2.69	2.76	2.0
Scrollkeeper.log	0.13	3.24	3.40	1.0
Windowserver_last.log	0.37	9.65	10.07	1.5
Yum.txt	0.11	1.91	2.03	5.0

SD: structure discovery
Ref: refinement
Tot: total

HW: hand-written

Training Time



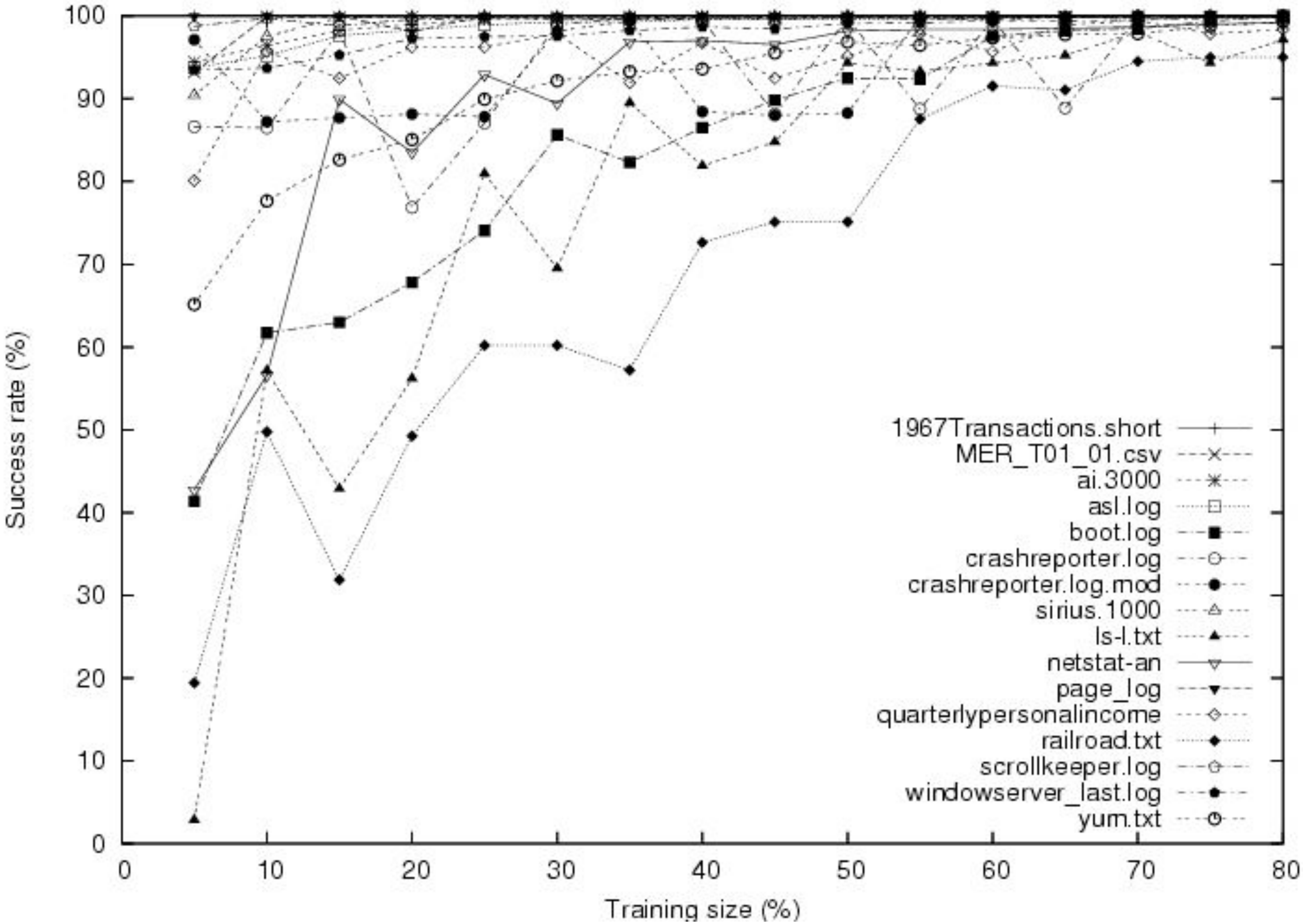
Normalized MDL Scores

Data source	SD	Ref	HW
1967Transactions.short	0.295	0.218	0.268
MER_T01_01.cvs	0.648	0.112	0.138
Ai.3000	0.503	0.332	0.338
Asl.log	0.630	0.267	0.361
Boot.log	0.620	0.481	0.703
Crashreporter.log	0.607	0.328	0.348
Crashreporter.log.mod	0.612	0.329	0.347
Sirius.1000	0.602	0.470	0.438
Ls-l.txt	0.559	0.333	0.401
Netstat-an	0.413	0.394	0.319
Page_log	0.540	0.107	0.353
quarterlypersonalincome	0.544	0.367	0.354
Railroad.txt	0.715	0.506	0.522
Scrollkeeper.log	0.625	0.354	0.352
Windowserver_last.log	0.618	0.241	0.267
Yum.txt	0.827	0.305	0.474

SD: structure discovery
Ref: refinement

HW: hand-written

Training Accuracy



Type Complexity and Min. Training Size

Data source	Norm. Ty Complexity	90%	95%
Sirius.1000	0.0001	5	10
1967Transaction.short	0.0003	5	5
Ai.3000	0.0004	5	10
Asl.log	0.0012	5	10
Scrollkeeper.log	0.0020	5	5
Page_log	0.0032	5	5
MER_T01_01.csv	0.0037	5	5
Crashreporter.log	0.0052	10	15
Crashreporter.log.mod	0.0053	5	15
Windowserver_last.log	0.0084	5	15
Netstat-an	0.0118	25	35
Yum.txt	0.0124	30	45
quarterlypersonalincome	0.0170	10	10
Boot.log	0.0213	45	60
Ls-l.txt	0.0461	50	65
Railroad.txt	0.0485	60	75

Related Work

- Grammar Induction
  - Generally impossible with only positive examples (Gold, 1967).
  - Focus has been on theoretical problems and natural language.
- Information Extraction
  - User labels training data, trained system then extracts similar fragments (eg, rentals in Craig's list).
  - Soderland's Whisk system (1999), Kushmerick (1997)
- XML Inference
  - Leverages known tokenization for XML and tree-structure of XML data to infer DTDs and XSchema
  - Garofalakis et al, 2000; Fernau 2001; Arasu and Garcia-Molena 2003, Bex et al 2006,2007;

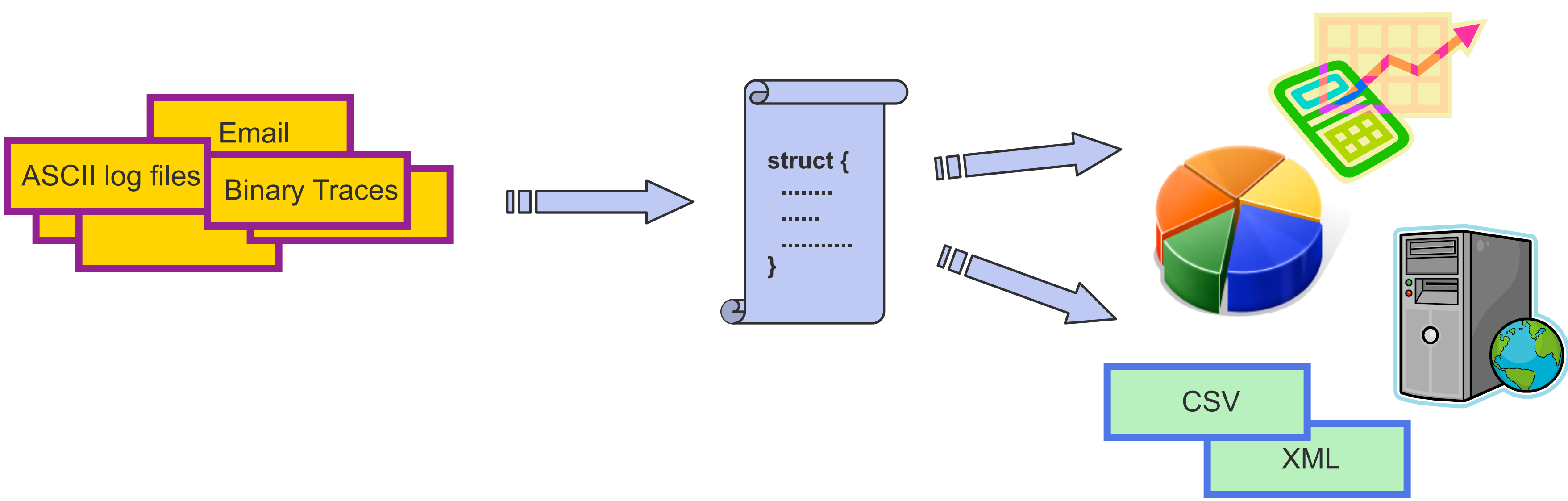
And much more; see paper for more details.

How do language ideas help?

- PADS declarative data description language serves as expressive target for inference system.
  - Specify only what data format is, not how it should be parsed or what data structures to build while parsing.
- Rewriting rules allow us to improve description in semantic-preserving way.
- Core type theoretic semantics allows us to generate PADS/ML or PADS/C specifications.
- Type-directed programming techniques will enable generic tool construction.

Technical Summary

- Format inference and automatic tool generation is feasible for many ASCII data formats.
- Current work: learning tokenizations.
- More information:
  - PADS web site: www.padsproj.org
  - POPL 2008 Paper: "From Dirt to Shovels"



Thanks & Acknowledgements

- Collaborators
  - David Walker (Princeton)
  - Kenny Zhu (Princeton)
  - Peter White (Galois)
- Other contributors
  - Alex Aiken (Stanford)
  - David Blei (Princeton)
  - David Burke (Galois)
  - Vikas Kedia (Stanford)
  - John Launchbury (Galois)
  - Rob Shapire (Princeton)
  - Qian Xi (Princeton)