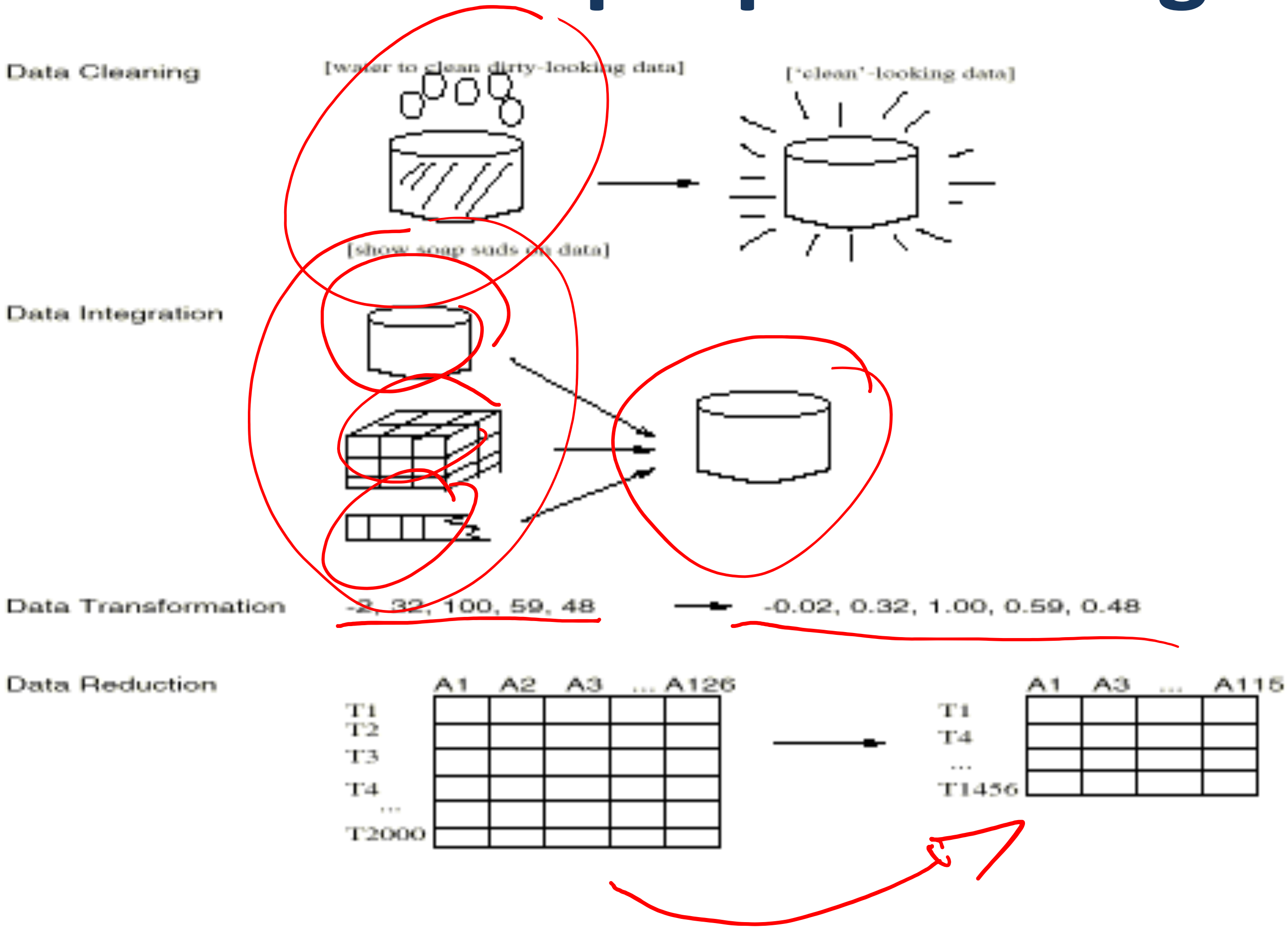
Data Quality and Data Cleaning

Major Tasks in Data Preprocessing

- Data cleaning
  - Fill in missing values, smooth noisy data, identify or remove outliers, and resolve inconsistencies
- Data integration
  - Integration of multiple databases, data cubes, or files
- Data transformation
  - Normalization and aggregation
- Data reduction
  - Obtains reduced representation in volume but produces the same or similar analytical results
- Data discretization
  - Part of data reduction but with particular importance, especially for numerical data

Forms of data preprocessing



THE MEANING OF DATA QUALITY

Meaning of Data Quality (1)

- Generally, you have a problem if the data doesn't mean what you think it does, or should
  - Data not up to spec : garbage in, glitches, etc.
  - You don't understand the spec : complexity, lack of metadata.
- Many sources and manifestations
- Data quality problems are expensive and pervasive
  - Data quality problems cost hundreds of billions each year.
  - Resolving data quality problems is often the biggest effort in a data mining study.

Example

T.Das|97336o8327|24.95|Y|-|0.0|1000
Ted J.|973-360-8779|2000|N|M|NY|1000

- Can we interpret the data?
  - What do the fields mean?
  - What is the key? The measures?
- Data glitches
  - Typos, multiple formats, missing / default values
- Metadata and domain expertise
  - Field three is Revenue. In dollars or cents?
  - Field seven is Usage. Is it *censored*?

Data Glitches

- Systemic changes to data which are external to the recorded process.
  - Changes in data layout / data types
    - Integer becomes string, fields swap positions, etc.
  - Changes in scale / format
    - Dollars vs. euros
  - Temporary reversion to defaults
    - Failure of a processing step
  - Missing and default values
    - Application programs do not handle NULL values well …
  - Gaps in time series
    - Especially when records represent incremental changes.

Conventional Definition of Data Quality

- Accuracy
  - The data was recorded correctly.
- Completeness
  - All relevant data was recorded.
- Uniqueness
  - Entities are recorded once.
- Timeliness
  - The data is kept up to date.
    - Special problems in federated data: time consistency.
- Consistency
  - The data agrees with itself.

Problems …

- Unmeasurable
  - Accuracy and completeness are extremely difficult, perhaps impossible to measure.
- Context independent
  - No accounting for what is important. E.g., if you are computing aggregates, you can tolerate a lot of inaccuracy.
- Incomplete
  - What about interpretability, accessibility, metadata, analysis, etc.
- Vague
  - The conventional definitions provide no guidance towards practical improvements of the data.

Finding a modern definition

- We need a definition of data quality which
  - Reflects the use of the data
  - Leads to improvements in processes
  - Is measurable (we can define metrics)

- First, we need a better understanding of how and where data quality problems occur
  - The data quality continuum

THE DATA QUALITY CONTINUUM

The Data Quality Continuum

Data and information is not static, it flows in a data collection and usage process

- Data gathering
- Data delivery
- Data storage
- Data integration
- Data retrieval
- Data mining/analysis

Data Gathering

- How does the data enter the system?
- Sources of problems:
  - Manual entry
  - No uniform standards for content and formats
  - Parallel data entry (duplicates)
  - Approximations, surrogates – Software/Hardware constraints
  - Measurement errors

Solutions

Potential Solutions:

- Preemptive:
  - Process architecture (build in integrity checks)
  - Process management (reward accurate data entry, data sharing, data stewards)
- Retrospective:
  - Cleaning focus (duplicate removal, merge/purge, name & address matching, field value standardization)
  - Diagnostic focus (automated detection of glitches).

Data Delivery

- Destroying or mutilating information by inappropriate pre-processing
  - Inappropriate aggregation
  - Nulls converted to default values
- Loss of data:
  - Buffer overflows
  - Transmission problems
  - No checks

Solutions

- Build reliable transmission protocols
  - Use a relay server
- Verification
  - Checksums, verification parser
  - Do the uploaded files fit an expected pattern?
- Relationships
  - Are there dependencies between data streams and processing steps
- Interface agreements
  - Data quality commitment from the data stream supplier.

Data Storage

- You get a data set. What do you do with it?
- Problems in physical storage
  - Can be an issue, but terabytes are cheap.
- Problems in logical storage (ER -> relations)
  - Poor metadata.
    - Data feeds are often derived from application programs or legacy data sources. What does it mean?
  - Inappropriate data models.
    - Missing timestamps, incorrect normalization, etc.
  - Ad-hoc modifications.
    - Structure the data to fit the GUI.
  - Hardware / software constraints.
    - Data transmission via Excel spreadsheets, Y2K

Solutions

- Metadata
  - Document and publish data specifications.
- Planning
  - Assume that everything bad will happen.
- Data exploration
  - Use data browsing and data mining tools to examine the data.
    - Does it meet the specifications you assumed?
    - Has something changed?

Data Integration

- Combine data sets (acquisitions, across departments).
- Common source of problems
  - Heterogenous data : no common key, different field formats
    - **Approximate matching**
  - Different definitions
    - What is a customer: an account, an individual, a family, …
  - Time synchronization
    - Does the data relate to the same time periods? Are the time windows compatible?
  - Legacy data
    - IMS, spreadsheets, ad-hoc structures
  - Sociological factors
    - Reluctance to share – loss of power.

Solutions

- Commercial Tools
  - Significant body of research in data integration
  - Many tools for address matching, schema mapping are available.
- Data browsing and exploration
  - Many hidden problems and meanings : must extract metadata.
  - View before and after results : did the integration go the way you thought?

Data Retrieval

- Exported data sets are often a view of the actual data. Problems occur because:
  - Source data not properly understood.
  - Just plain mistakes.
    - Inner join vs. outer join
    - Understanding NULL values
- Computational constraints
  - E.g., too expensive to give a full history, we'll supply a snapshot.
- Incompatibility

Data Mining and Analysis

- What are you doing with all this data anyway?
- Problems in the analysis.
  - Scale and performance
  - Confidence bounds?
  - Black boxes and dart boards
    - “fire your Statisticians”
  - Attachment to models
  - Insufficient domain expertise
  - Casual empiricism

Solutions

- Data exploration
  - Determine which models and techniques are appropriate, find data bugs, develop domain expertise.
- Continuous analysis
  - Are the results stable? How do they change?
- Accountability
  - Make the analysis part of the feedback loop.

DATA QUALITY

Meaning of Data Quality (1)

There are many types of data, which have different uses and typical quality problems

- Federated data
- High dimensional data
- Descriptive data
- Longitudinal data
- Streaming data
- Web (scraped) data
- Numeric vs. categorical vs. text data

Meaning of Data Quality (2)

- There are many uses of data
  - Operations
  - Aggregate analysis
  - Customer relations …
- Data Interpretation : the data is useless if we don’t know all of the *rules* behind the data.
- Data Suitability: Can you get the answer from the available data
  - Relevant data is missing

Data Quality Constraints

- Many data quality problems can be captured by *static* constraints based on the schema.
  - Nulls not allowed, field domains, foreign key constraints, etc.
- Many others are due to problems in workflow, and can be captured by *dynamic* constraints
  - E.g., orders above $200 are processed by Biller 2
- The constraints follow an 80-20 rule
  - A few constraints capture most cases, thousands of constraints to capture the last few cases.
- Constraints are measurable.

DATA QUALITY METRICS

Data Quality Metrics

- We want a measurable quantity
  - Indicates what is wrong and how to improve
  - Realize that data quality is a messy problem, no set of numbers will be perfect
- Types of metrics
  - Static vs. dynamic constraints
  - Operational vs. diagnostic
- Metrics should be *directionally correct* with an improvement in use of the data.
- A very large number metrics are possible
  - Choose the most important ones.

Examples of Data Quality Metrics

- Conformance to schema
  - Evaluate constraints on a snapshot.
- Conformance to business rules
  - Evaluate constraints on changes in the database.
- Accuracy
  - Perform inventory (expensive), or use proxy (track complaints). Audit samples?
- Accessibility
- Interpretability
- Glitches in analysis
- Successful completion of end-to-end process

TECHNICAL TOOLS

Technical Approaches

- A multi-disciplinary approach is needed to attack data quality problems
  - No one approach solves all problem
- Process management
  - Ensure proper procedures
- Statistics
  - Focus on analysis: find and repair anomalies in data.
- Database
  - Focus on relationships: ensure consistency.
- Metadata / domain expertise
  - What does it mean? Interpretation

Process Management

Business processes which encourage data quality.

- Assign dollars to quality problems
- Standardization of content and formats
- Enter data once, enter it correctly (incentives for sales, customer care)
- Automation
- Assign responsibility: data stewards
- End-to-end data audits and reviews
  - Transitions between organizations.
- Data Monitoring
- Data Publishing
- Feedback loops

Feedback Loops

- Data processing systems are often thought of as open-loop systems.
  - Do your processing then throw the results over the fence.
- Analogy to control systems : *feedback loops*.
  - *Monitor* the system to detect difference between actual and intended
  - *Feedback loop* to correct the behavior of earlier components
  - Of course, data processing systems are much more complicated than linear control systems.

Example

- Sales, provisioning, and billing for telecommunications service
  - Many stages involving handoffs between organizations and databases
  - Simplified picture
- *Transition between organizational boundaries is a common cause of problems*.
- Natural feedback loops
  - Customer complains if the bill is to high
- Missing feedback loops
  - No complaints if we undercharge.

Monitoring

- Use data monitoring to add missing feedback loops.
- Methods:
  – Data tracking / auditing
    - Follow a sample of transactions through the workflow.
    - Build secondary processing system to detect possible problems.
  – Reconciliation of incrementally updated databases with original sources.
  – Mandated consistency with a Database of Record
  – Feedback loop sync-up
  – Data Publishing

Data Publishing

- Make the contents of a database available in a readily accessible and digestible way
  - Web interface (universal client).
  - Data Squashing : Publish aggregates, cubes, samples, parametric representations.
  - Publish the metadata.
- Close feedback loops by getting a lot of people to look at the data.
- Surprisingly difficult sometimes.
  - Organizational boundaries, loss of control interpreted as loss of power, desire to hide problems.

Statistical Approaches

- No explicit data quality methods
  - Traditional statistical data collected from carefully designed experiments, often tied to analysis
  - But, there are methods for finding anomalies and repairing data.
  - Existing methods can be adapted for data quality purposes.
- Four categories can be adapted for data quality
  - Missing, incomplete, ambiguous or damaged data e.g truncated, censored
  - Suspicious or abnormal data e.g. outliers
  - Testing for departure from models
  - Goodness-of-fit

Missing Data

- Missing data - values, attributes, entire records, entire sections
- Missing values and defaults are indistinguishable
- Truncation/censoring - not aware, mechanisms not known
- Problem: Misleading results, bias.

Detecting Missing Data

Overtly missing data

- Match data specifications against data - are all the attributes present?
- Scan individual records - are there gaps?
- Rough checks : number of files, file sizes, number of records, number of duplicates
- Compare estimates (averages, frequencies, medians) with “expected” values and bounds; check at various levels of granularity since aggregates can be misleading.

Missing data detection (cont.)

Hidden damage to data

- Values are truncated or censored - check for spikes and dips in distributions and histograms
- Missing values and defaults are indistinguishable - too many missing values? metadata or domain expertise can help
- Errors of omission e.g. all calls from a particular area are missing - check if data are missing randomly or are localized in some way

Imputing Values to Missing Data

- In federated data, between 30%-70% of the data points will have at least one missing attribute - data wastage if we ignore all records with a missing value
- Remaining data is seriously biased
- Lack of confidence in results
- Understanding pattern of missing data unearths data integrity issues

Missing Value Imputation (1)

- Standalone imputation
  - Mean, median, other point estimates
  - Assume: Distribution of the missing values is the same as the non-missing values.
  - Does not take into account inter-relationships
  - Introduces bias
  - Convenient, easy to implement

Missing Value Imputation (2)

- Better imputation - use attribute relationships
- Assume : all prior attributes are populated
  - That is, *monotonicity* in missing values.

X1	X2	X3	X4	X5
1.0	20	3.5	4	.
1.1	18	4.0	2	.
1.9	22	2.2	.	.
0.9	15	.	.	.

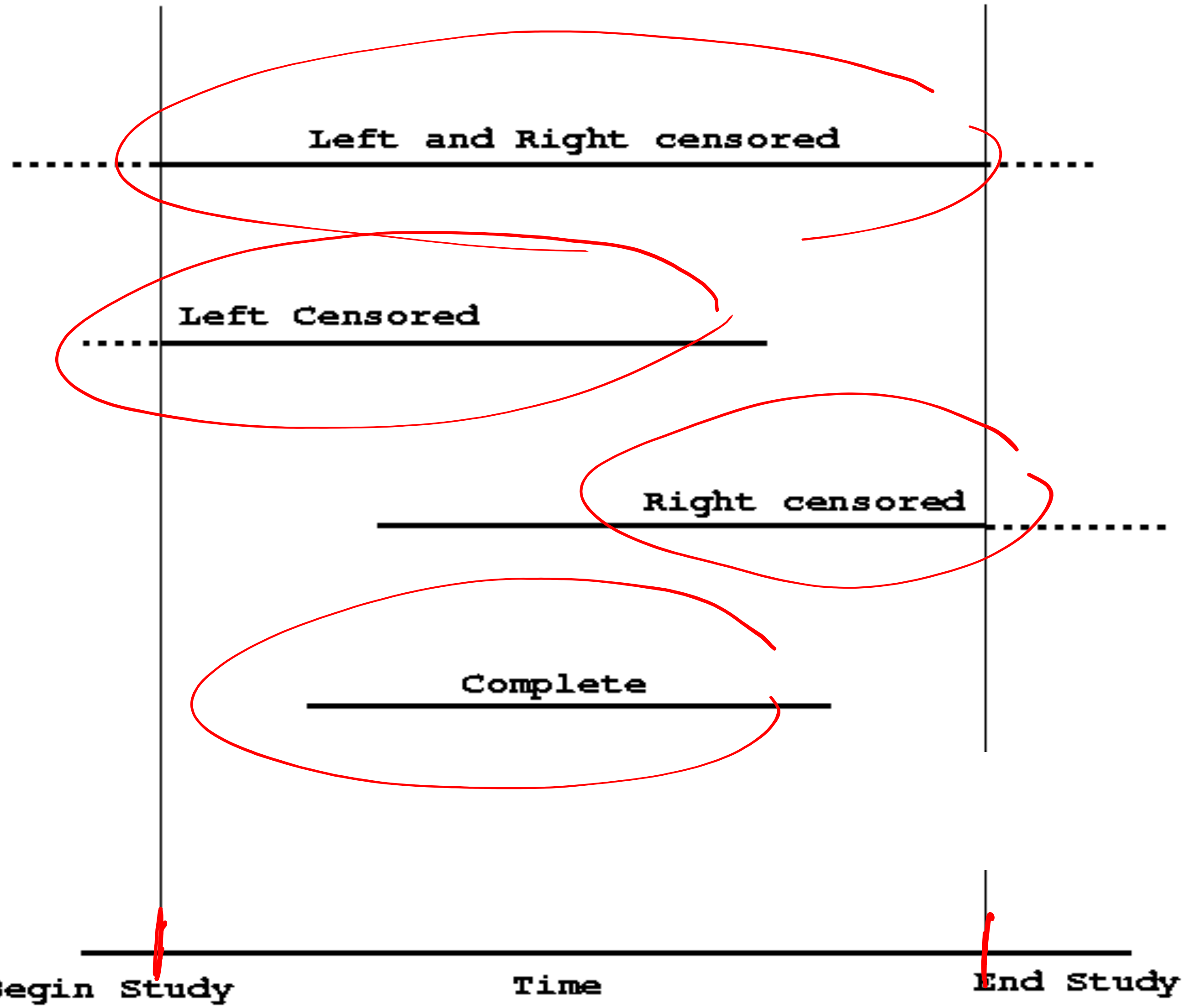
- Two techniques
  - Regression (parametric),
  - Propensity score (nonparametric)

Censoring and Truncation

- Well studied in Biostatistics, relevant to time dependent data e.g. duration
- *Censored* - Measurement is bounded but not precise e.g., Call duration > 20 are recorded as 20
- *Truncated* - Data point dropped if it exceeds or falls below a certain bound e.g. customers with less than 2 minutes of calling

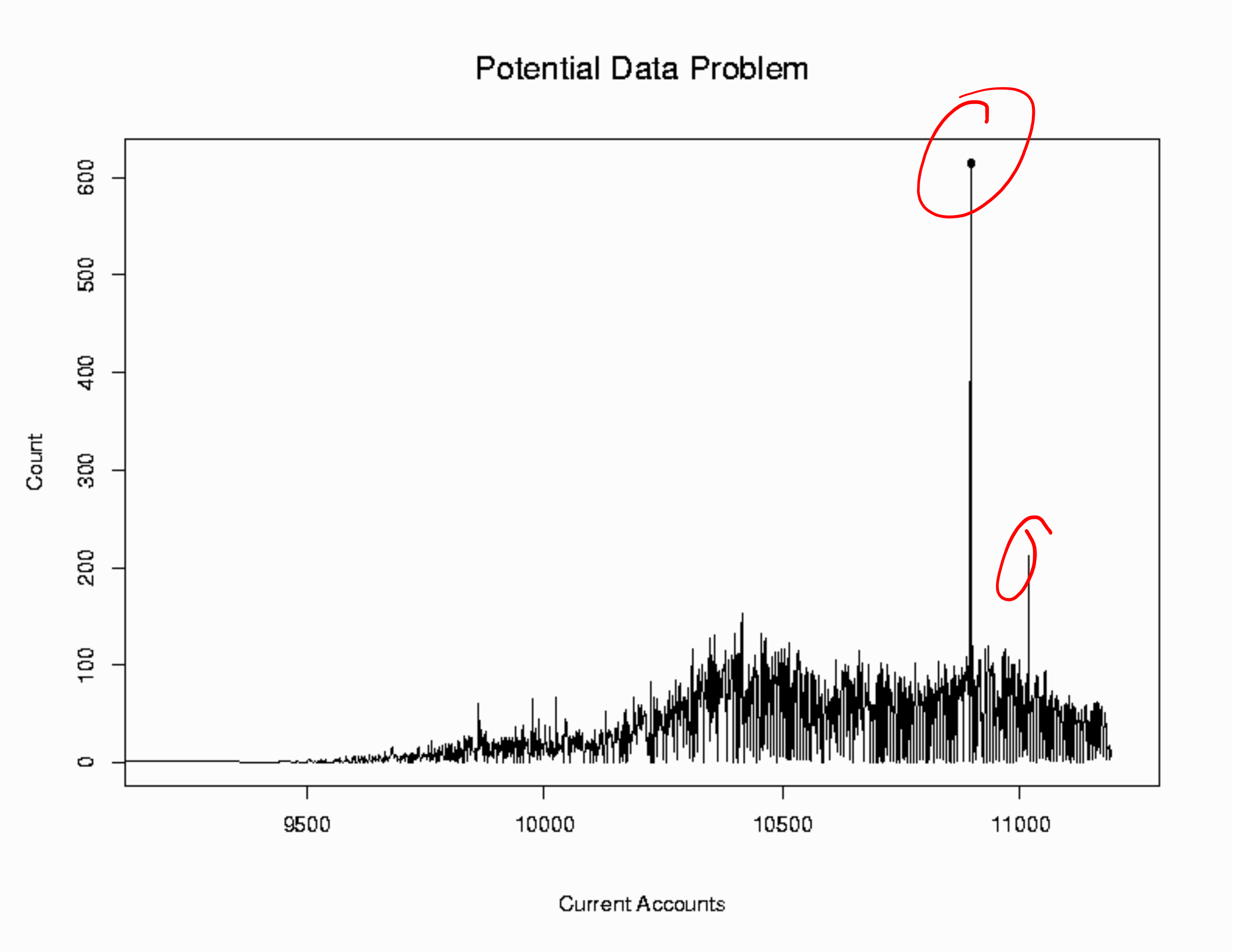
  per month

Censored Time Intervals



Censoring/Truncation (cont.)

- If censoring/truncation mechanism not known, analysis can be inaccurate and biased.
- But if you know the mechanism, you can mitigate the bias from the analysis.
- Metadata should record the existence as well as the nature of censoring/truncation


Potential Data Problem
Count
0
100
200
300
400
500
600
9500
10000
10500
11000
Current Accounts

Suspicious Data

- Consider the data points

  3, 4, 7, 4, 8, 3, 9, 5, 7, 6, 92

- "92" is suspicious - an *outlier*
- Outliers are potentially legitimate
- Often, they are data or model glitches
- Or, they could be a data miner's dream, e.g., highly profitable customers

Outliers

- Outlier – "departure from the expected"
- Types of outliers – defining "expected"
- Many approaches
  - Error bounds, tolerance limits – control charts
  - Model based – regression depth, analysis of residuals
  - Geometric
  - Distributional
  - Time Series outliers

Model Fitting and Outliers

- Models summarize general trends in data
  - more complex than simple aggregates
  - e.g. linear regression, logistic regression focus on attribute relationships
- Data points that do not conform to well fitting models are *potential outliers*
- Goodness of fit tests (DQ for analysis/mining)
  - check suitableness of model to data
  - verify validity of assumptions
  - data rich enough to answer analysis/business question?

Set Comparison and Outlier Detection

- “Model” consists of partition based summaries
- Perform nonparametric statistical tests for a rapid section-wise comparison of two or more massive data sets
- If there exists a baseline “good’’ data set, this technique can detect potentially corrupt sections in the test data set

Time Series Outliers

- Data is a time series of measurements of a large collection of entities (e.g. customer usage).
- Vector of measurements define a trajectory for an entity.
- A trajectory can be glitched, and it can make radical but valid changes.
- Approach: develop models based on entity's past behavior (*within*) and all entity behavior (*relative*).
- Find potential glitches:
  - Common glitch trajectories
  - Deviations from within and relative behavior.

Database Tools

- Most DBMS's provide many data consistency tools
  - Transactions
  - Data types
  - Domains (restricted set of field values)
  - Constraints
    - Column Constraints
      - Not Null, Unique, Restriction of values
    - Table constraints
      - Primary and foreign key constraints
  - Powerful query language
  - Triggers
  - Timestamps, temporal DBMS

Why is every DB dirty?

- Consistency constraints are often not used
  - Cost of enforcing the constraint
    - E.g., foreign key constraints, triggers.
  - Loss of flexibility
  - Constraints not understood
    - E.g., large, complex databases with rapidly changing requirements
  - DBA does not know / does not care.
- Garbage in
  - Merged, federated, web-scraped databases
- Undetectable problems
  - Incorrect values, missing data
- Metadata not maintained
- Database is too complex to understand

Too complex to understand …

- Unintended consequences
  - Best example: cascading deletes to enforce participation constraints
    - Consider salesforce table and sales table. Participation constraint of salesforce in sales. Then you fire a salesman …
- Real life is complicated. Hard to anticipate special situations
  - Textbook example of functional dependencies: zip code determines state. Except for a few zip codes in sparsely populated regions that straddle states.

Tools

- Extraction, Transformation, Loading
- Approximate joins
- Duplicate finding
- Database exploration

Data Loading

- Extraction, Transformation, Loading (ETL)
- The data might be derived from a questionable source.
  - Federated database, Merged databases
  - Text files, log records
  - Web scraping
- The source database might admit a limited set of queries
- The data might need restructuring
  - Field value transformation
  - Transform tables (e.g. denormalize, pivot, fold)

Extract, Transform, Load

- Provides tools to
  - Access data (DB drivers, web page fetch, parse tools)
  - Validate data (ensure constraints)
  - Transform data (e.g. addresses, phone numbers)
  - Load data
- Design automation
  - Schema mapping
  - Queries to data sets with limited query interfaces (web queries)

Web Scraping

- Lots of data in the web, but its mixed up with a lot of junk.
- Problems:
  - Limited query interfaces
    - Fill in forms
  - “Free text” fields
    - E.g. addresses
  - Inconsistent output
    - i.e., html tags which mark interesting fields might be different on different pages.
  - Rapid change without notice.

Tools

- Automated generation of web scrapers
  - Excel will load html tables
- Automatic translation of queries
  - Given a description of allowable queries on a particular source
- Monitor results to detect quality deterioration
- Extraction of data from free-form text
  - E.g. addresses, names, phone numbers
  - Auto-detect field domain

Approximate Matching

- Relate tuples whose fields are “close”
  - Approximate string matching
    - Generally, based on edit distance.
    - Fast SQL expression using a *q-gram* index
  - Approximate tree matching
    - For XML
    - Much more expensive than string matching
    - Recent research in fast approximations
  - Feature vector matching
    - Similarity search
    - Many techniques discussed in the data mining literature.
  - Ad-hoc matching
    - Look for a clever trick.

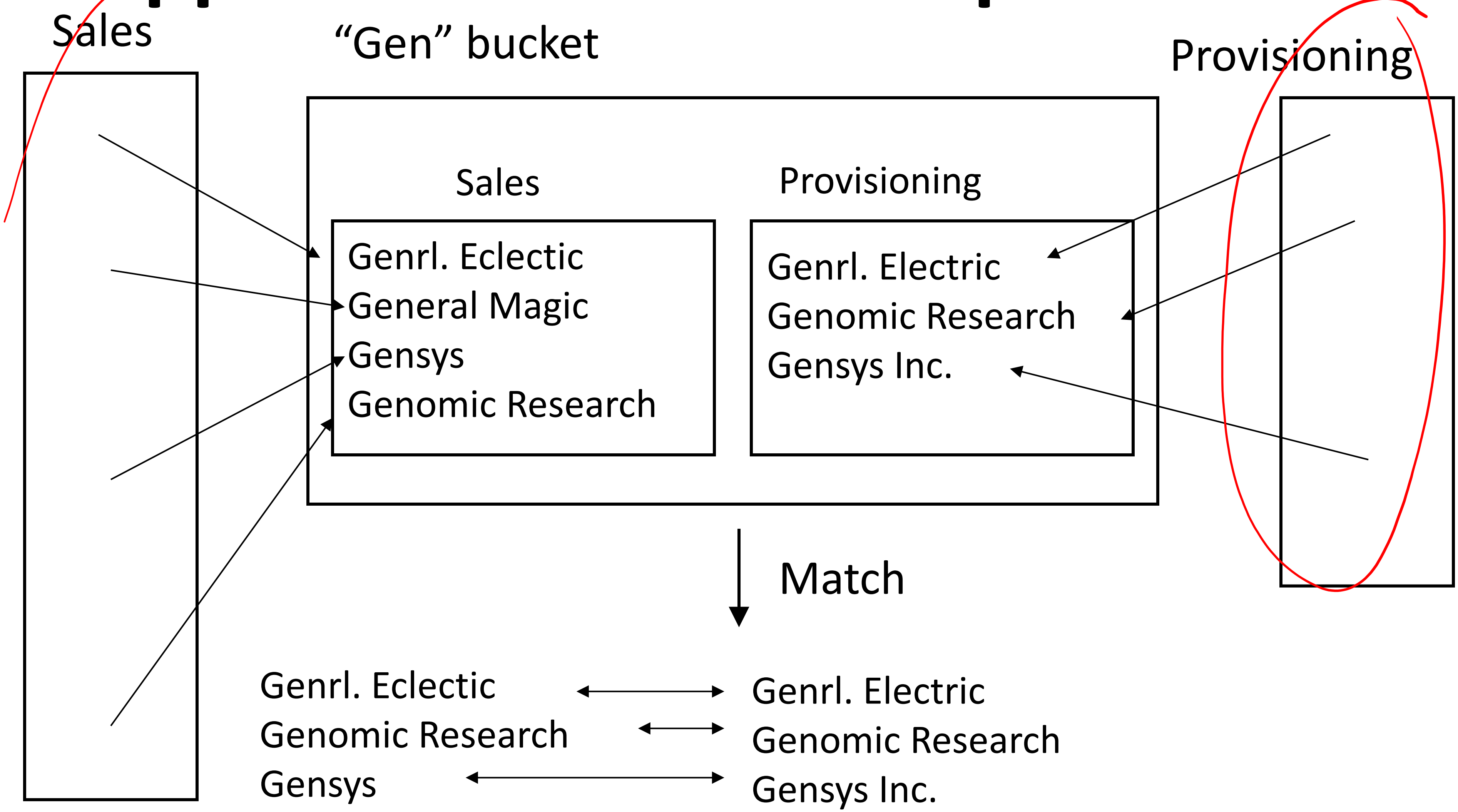
Approximate Joins and Duplicate Elimination

- Perform joins based on incomplete or corrupted information.
  - Approximate join : between two different tables
  - Duplicate elimination : within the same table
- More general than approximate matching.
  - **Semantics**: Need to use special transforms and scoring functions.
  - **Correlating information**: verification from other sources, e.g. usage correlates with billing.
  - **Missing data**: Need to use several orthogonal search and scoring criteria.
- But approximate matching is a valuable tool …

Algorithm

- Partition data set
  - By hash on computed key
  - By sort order on computed key
  - By similarity search / approximate match on computed key
- Perform scoring within the partition
  - Hash: all pairs
  - Sort order, similarity search: target record to retrieved records
- Record pairs with high scores are matches
- Use multiple computed keys / hash functions
- Duplicate elimination : duplicate records form an equivalence class.

Approximate Join Example



Database Exploration

- Tools for finding problems in a database
  - Opposite of ETL
  - Similar to data quality mining
- Simple queries are effective:

```
Select Field, count(*) as Cnt
from Table
Group by Field
Order by Cnt Desc
```

  - Hidden NULL values at the head of the list, typos at the end of the list
- Just look at a sample of the data in the table.

Database Profiling

- Systematically collect summaries of the data in the database
  - Number of rows in each table
  - Number of unique, null values of each field
  - Skewness of distribution of field values
  - Data type, length of the field
    - Use free-text field extraction to guess field types (address, name, zip code, etc.)
  - Functional dependencies, keys
  - Join paths
- Does the database contain what you think it contains?
  - Usually not.

Finding Keys and Functional Dependencies

- **Key**: set of fields whose value is unique in every row
- **Functional Dependency:** A set of fields which determine the value of another field
  - E.g., ZipCode determines the value of State
    - But not really …
- Problems: keys not identified, uniqueness not enforced, hidden keys and functional dependencies.
- Key finding is expensive: O(f^k) `Count Distinct` queries to find all keys of up to k fields.
- Fortunately, we can prune a lot of this search space if we search only for *minimal* keys and FDs
- **Approximate keys** : almost but not quite unique.
- **Approximate FD** : similar idea

Finding Join Paths

- How do I correlate this information?
- In large databases, hundreds of tables, thousands of fields.
- Field names are *very* unreliable.
  - Natural join does not exist outside the laboratory.
- Use data types and field characterization to narrow the search space.

Domain Expertise

Data quality gurus: “We found these peculiar records in your database after running sophisticated algorithms!”

Domain Experts: “Oh, those apples - we put them in the same baskets as oranges because there are too few apples to bother. Not a big deal. We knew that already.”

Why Domain Expertise?

- Domain expertise is important for understanding the data, the problem and interpreting the results
  - "The counter resets to 0 if the number of calls exceeds N".
  - "The missing values are represented by 0, but the default billed amount is 0 too."
- Insufficient domain expertise is a primary cause of poor data quality– data are unusable
- Domain expertise should be documented as metadata

Metadata

- Usually in people's heads – seldom documented
- Fragmented across organizations
  - Often experts don't agree. Force consensus.
- Lost during personnel and project transitions
- If undocumented, deteriorates and becomes fuzzy over time

Metadata

- Data about the data
- Data types, domains, and constraints help, but are often not enough
- Interpretation of values
  - Scale, units of measurement, meaning of labels
- Interpretation of tables
  - Frequency of refresh, associations, view definitions
- Most work done for scientific databases
  - Metadata can include programs for interpreting the data set.

XML

- Tree structured
  - Multiple field values, complex structure, etc.
- "Self-describing" : schema is part of the record
  - Field attributes
- DTD : minimal schema in an XML record.

<tutorial>
<title> Data Quality and Data Cleaning: An Overview <\title>
<Conference area="database"> SIGMOD <\Conference>
<author> T. Dasu
<bio> Statistician <\bio> <\author>
<author> T. Johnson
<institution> AT&T Labs <\institution> <\author>
<\tutorial>

Lineage Tracking

- Record the processing used to create data
  - Coarse grained: record processing of a table
  - Fine grained: record processing of a record
- Record graph of data transformation steps.
- Used for analysis, debugging, feedback loops

EXAMPLE

Motivation: Data Cleaning

Find movies starring Tom Hanks

Star	Title	Year	Genre
Keanu Reeves	The Matrix	1999	Sci-Fi
Tom Hanks	Toy Story 3	2010	Animation
Schwarzenegger	The Terminator	1984	Sci-Fi
Samuel Jackson	The man	2006	Crime

Movies starring S..warz…ne…ger?

Star	Title	Year	Genre
Keanu Reeves	The Matrix	1999	Sci-Fi
Tom Hanks	Toy Story 3	2010	Animation
Schwarzenegger	The Terminator	1984	Sci-Fi
Samuel Jackson	The man	2006	Crime

Similarity Search

Find movies with a star “similar to” Schwarrzenger.

Star	Title	Year	Genre
Keanu Reeves	The Matrix	1999	Sci-Fi
Samuel Jackson	Iron man	2008	Sci-Fi
Schwarzenegger	The Terminator	1984	Sci-Fi
Samuel Jackson	The man	2006	Crime

Record Linkage

Table R

Star
Keanu Reeves
Samuel Jackson
Schwarzenegger
…

Table S

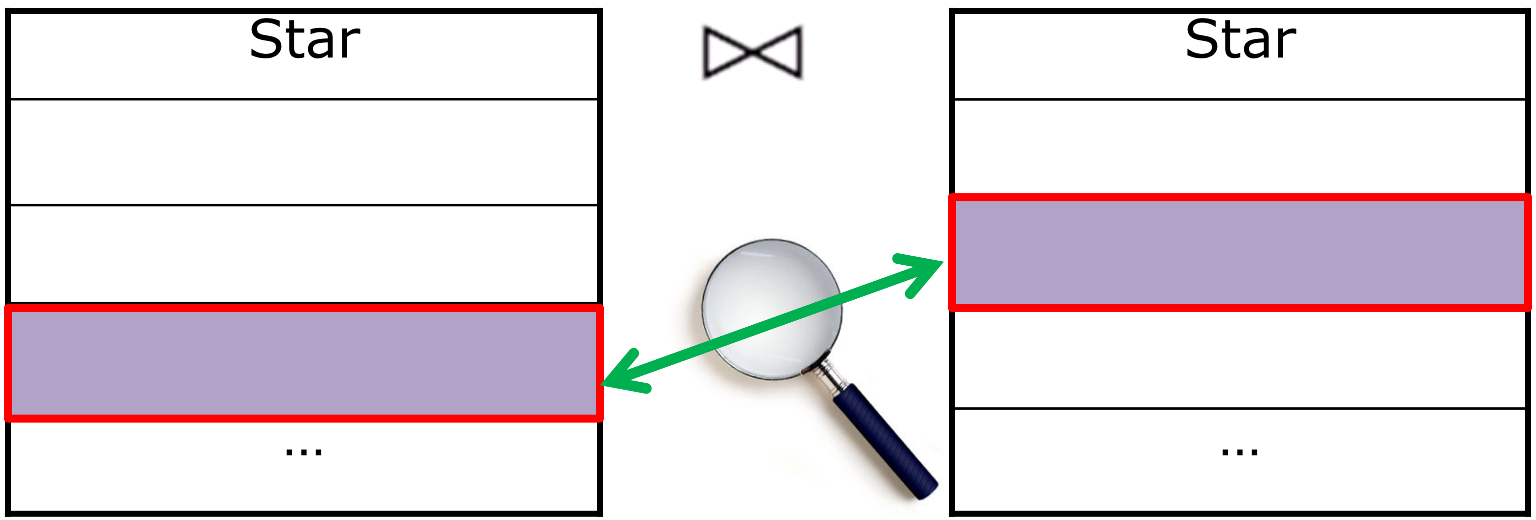
Star
Keanu Reeves
Samuel L. Jackson
Schwarzenegger
…

Two-step solution

Table R

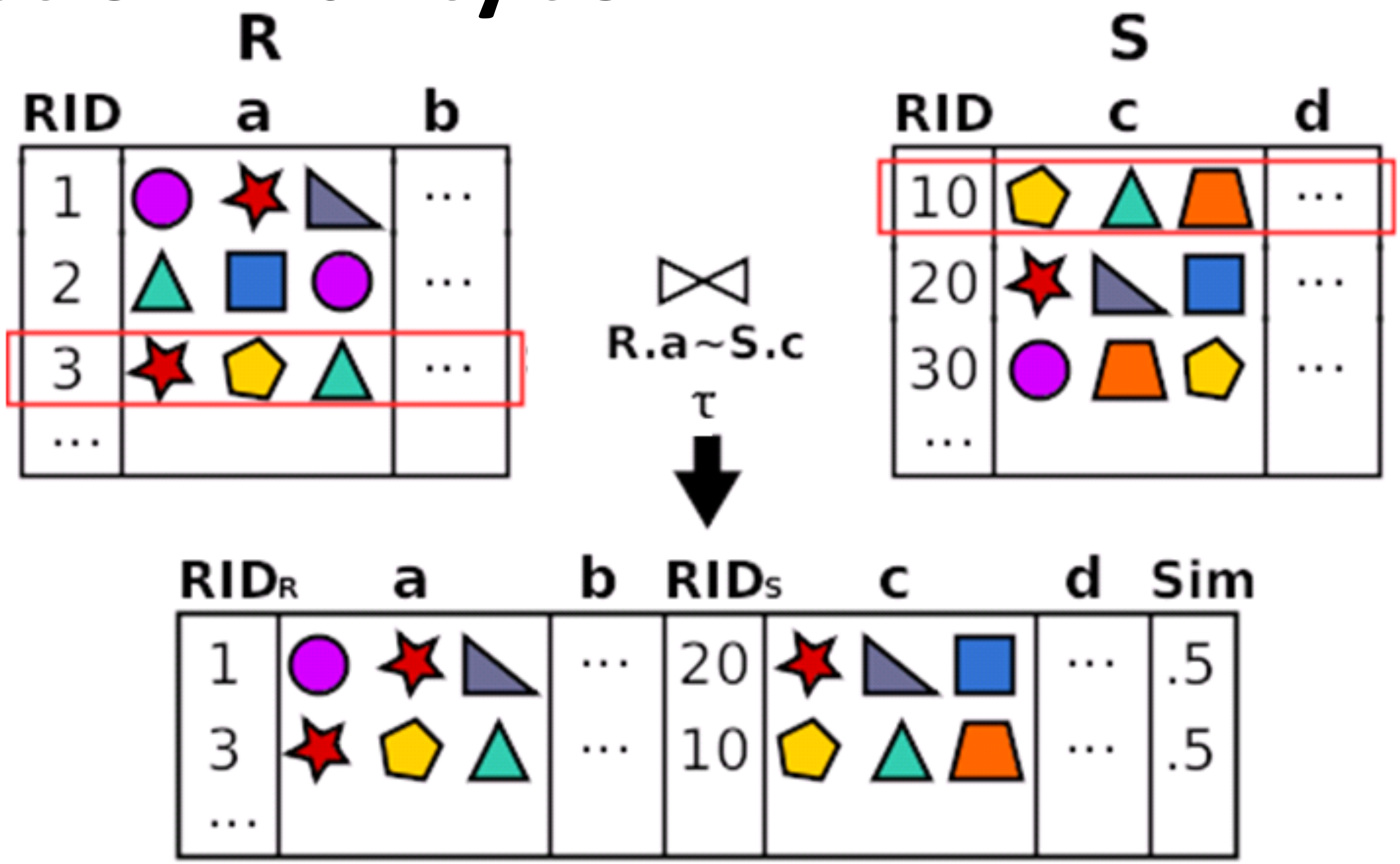
Step 1: Similarity Join

Table S



Step 2: Verification

Set-Similarity Join



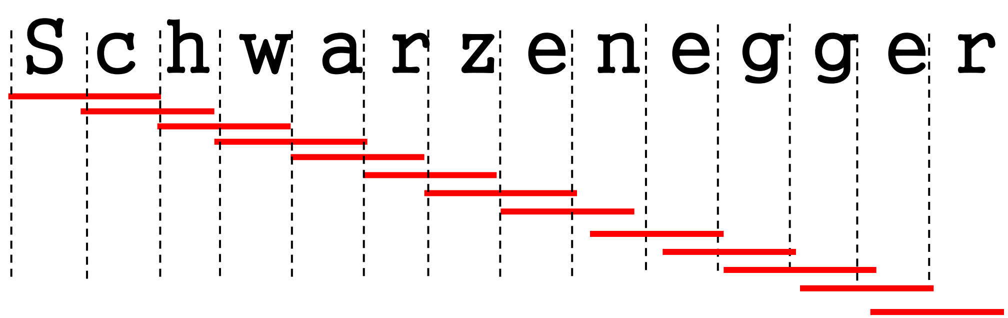
Finding pairs of records with a similarity on their join attributes > t

Why this formulation?

- Word tokens:

  "Samuel L. Jackson" → {Samuel, L., Jackson}
  "Samuel Jackson" → {Samuel, Jackson}

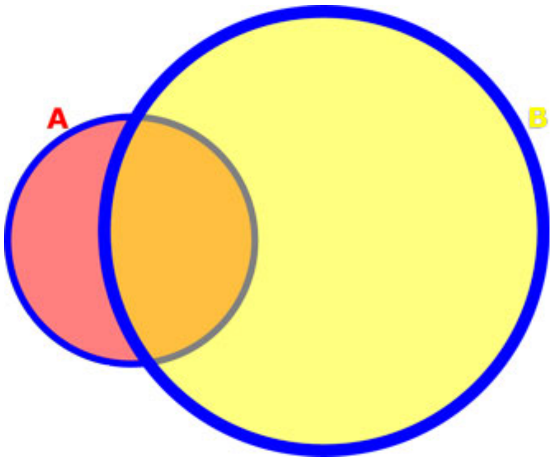
- Gram tokens:



Set-similarity functions

- Jaccard
- Dice
- Cosine
- Hamming
- …

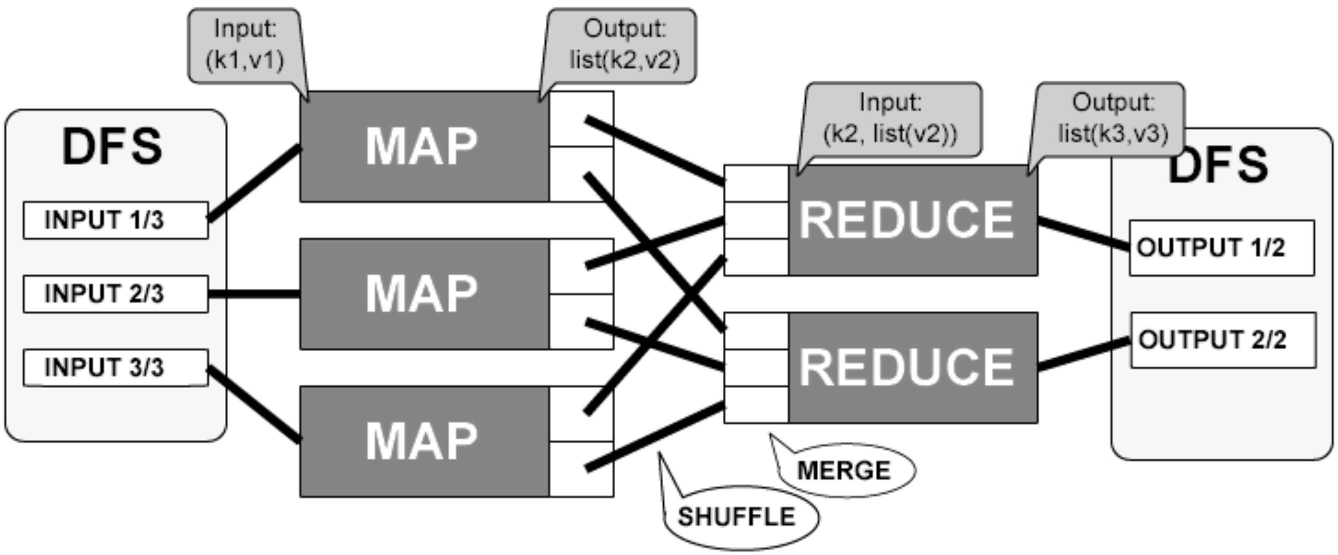
$$Jaccard(A,B) = \frac{|A \cap B|}{|A \cup B|} \geq t$$



All solvable in this framework

A naïve solution

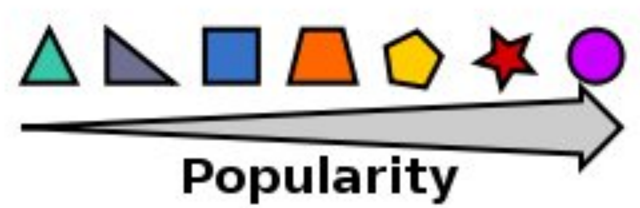
- Map: <23, (a,b,c)> → (a, 23), (b, 23), (c, 23)
- Reduce:(a,23),(a,29),(a,50), …→ Verify each pair



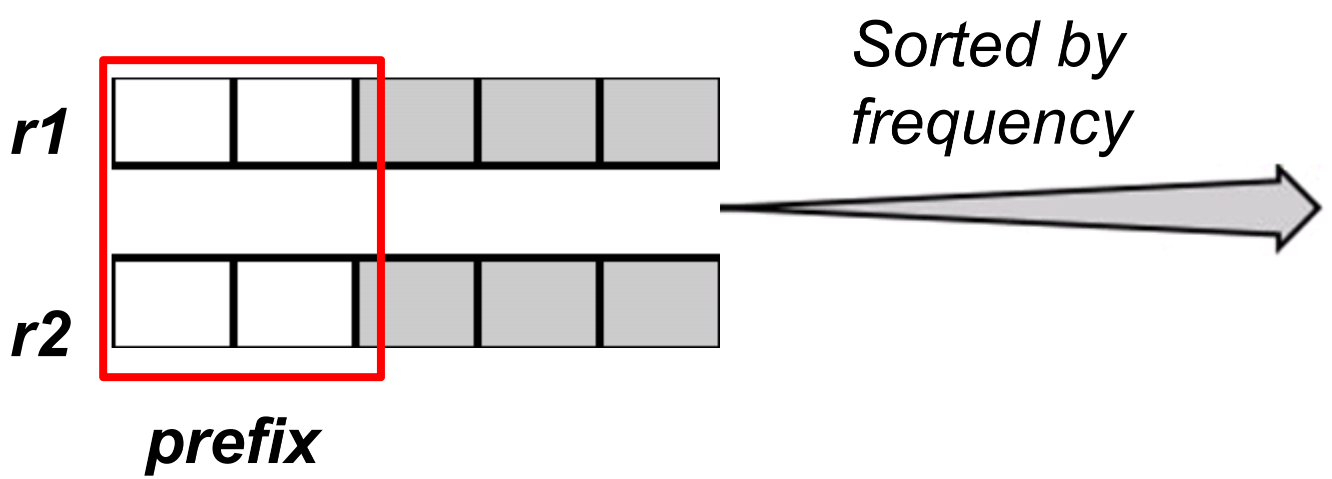
- Too much data to transfer
- Too many pairs to verify

Solving frequency skew: prefix filtering

- Sort tokens by frequency (ascending)

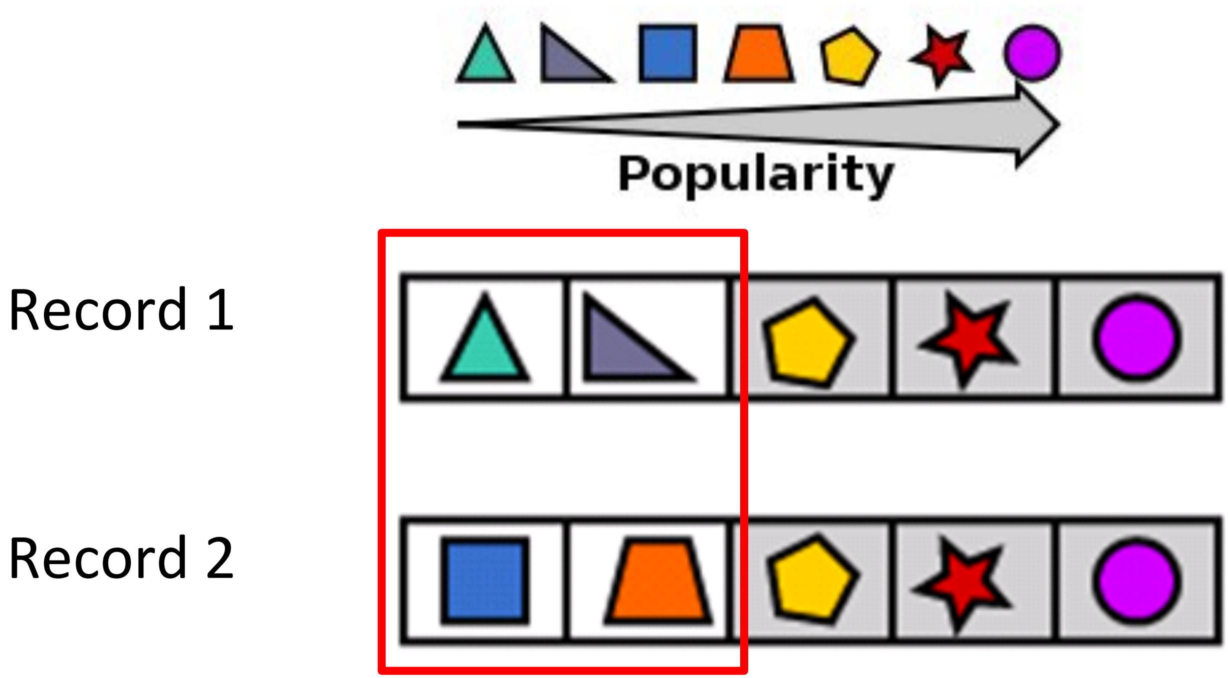


- Prefix of a set: least frequent tokens



- Prefixes of similar sets should share tokens

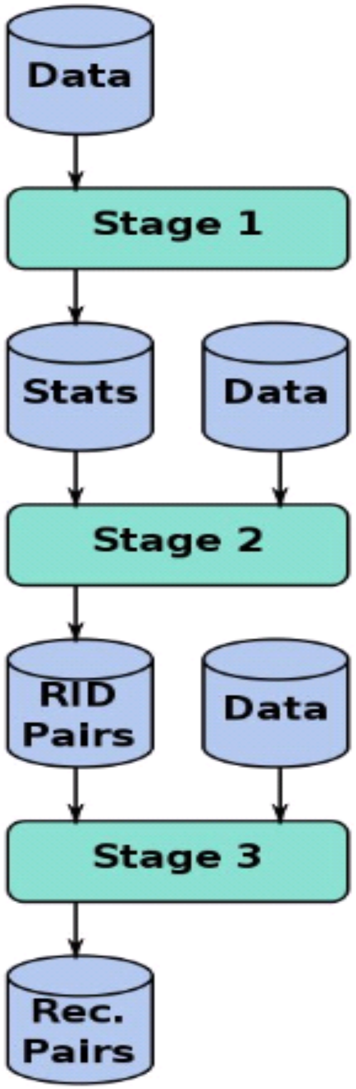
Prefix filtering: example



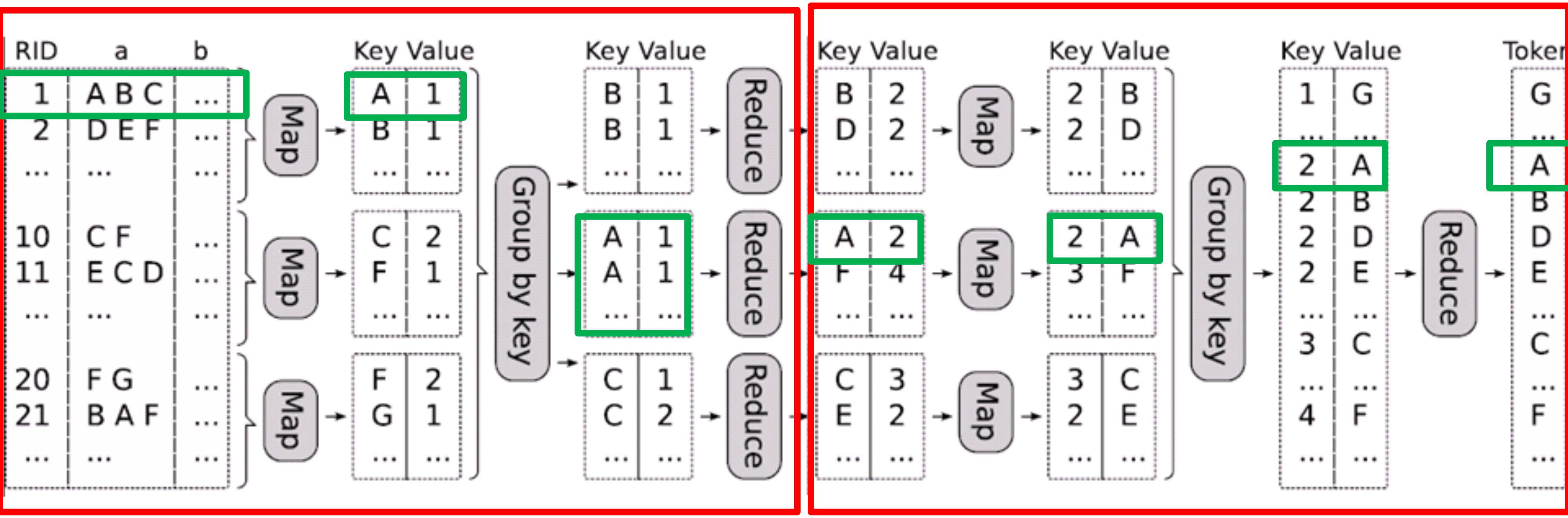
- Each set has 5 tokens
- “Similar”: they share at least 4 tokens
- Prefix length: 2

Hadoop Solution: Overview

- Stage 1: Order tokens by frequency
- Stage 2: Finding "similar" id pairs
- Stage 3: id pairs → record pairs



Stage 1: Sort tokens by frequency



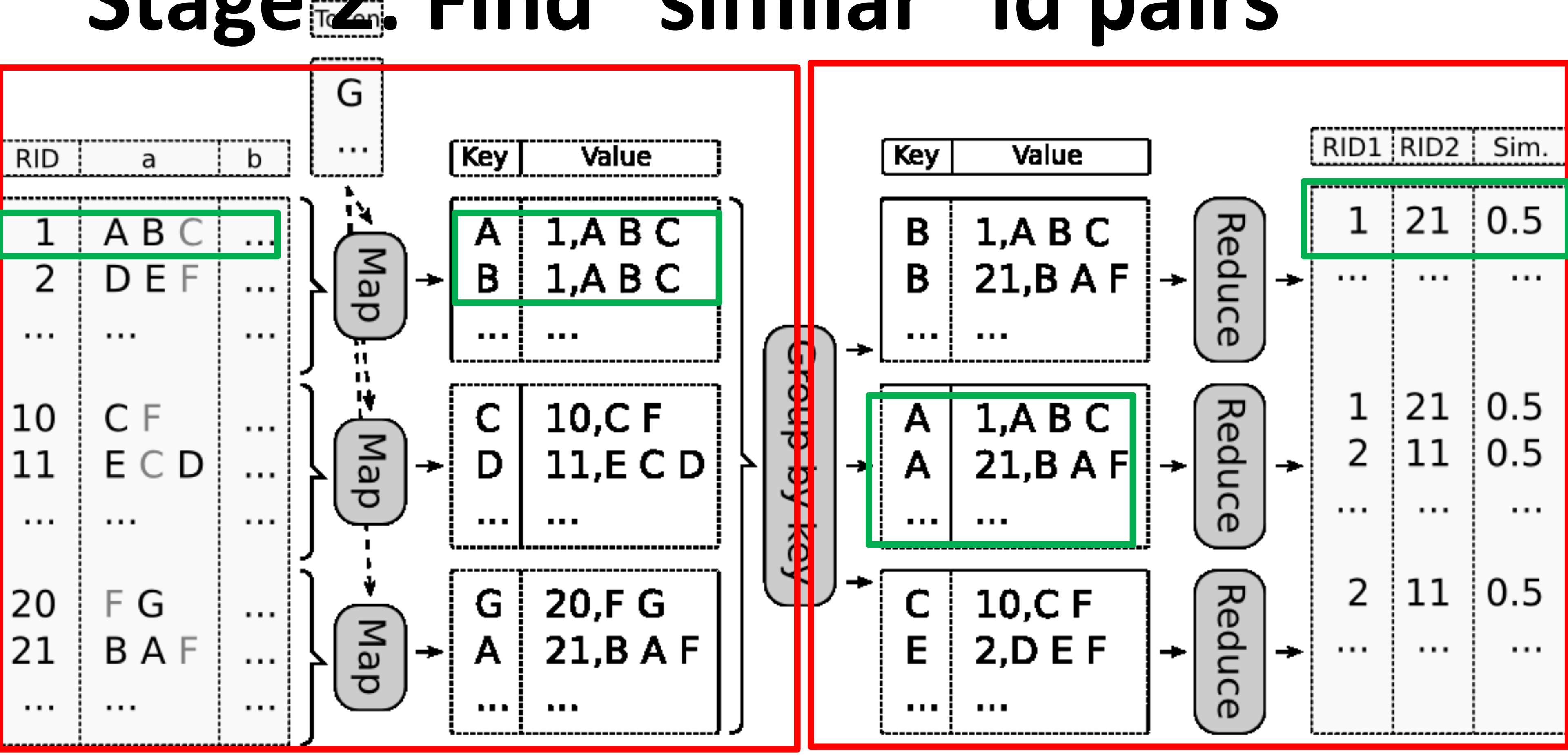
Compute token frequencies

Sort them

MapReduce phase 1

MapReduce phase 2

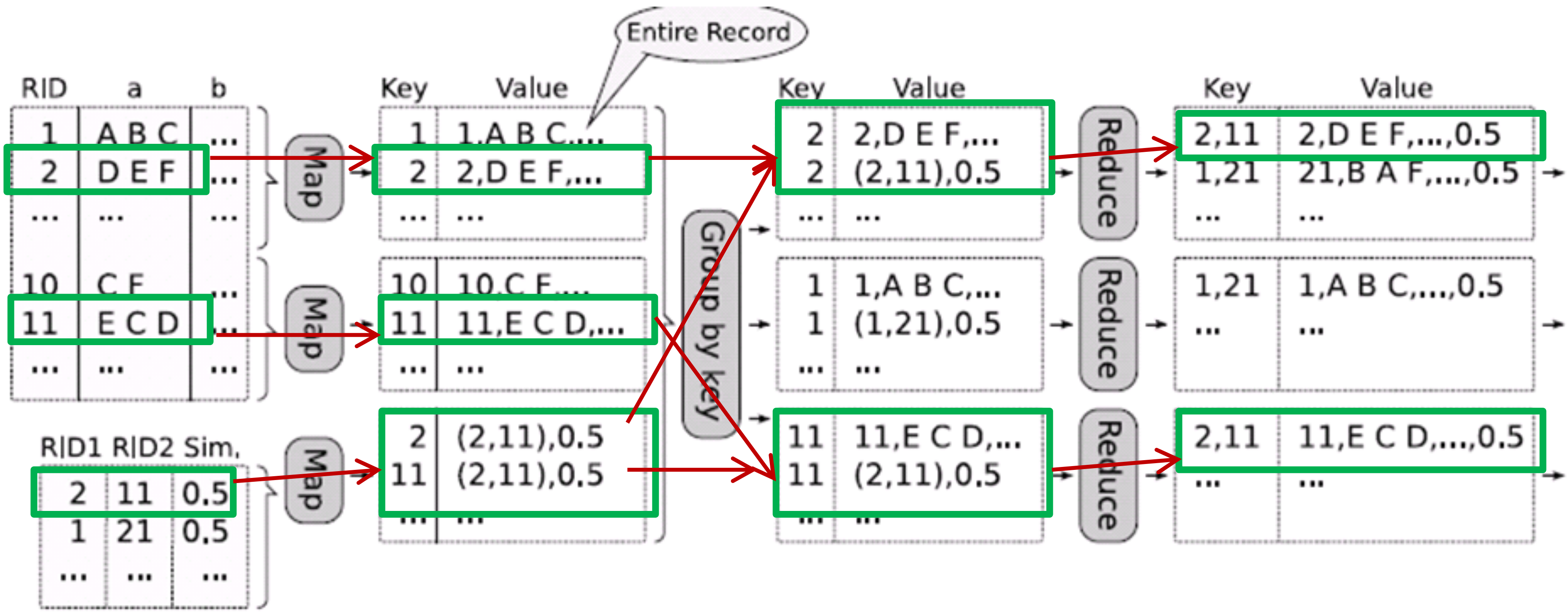
Stage 2: Find "similar" id pairs



Partition using prefixes

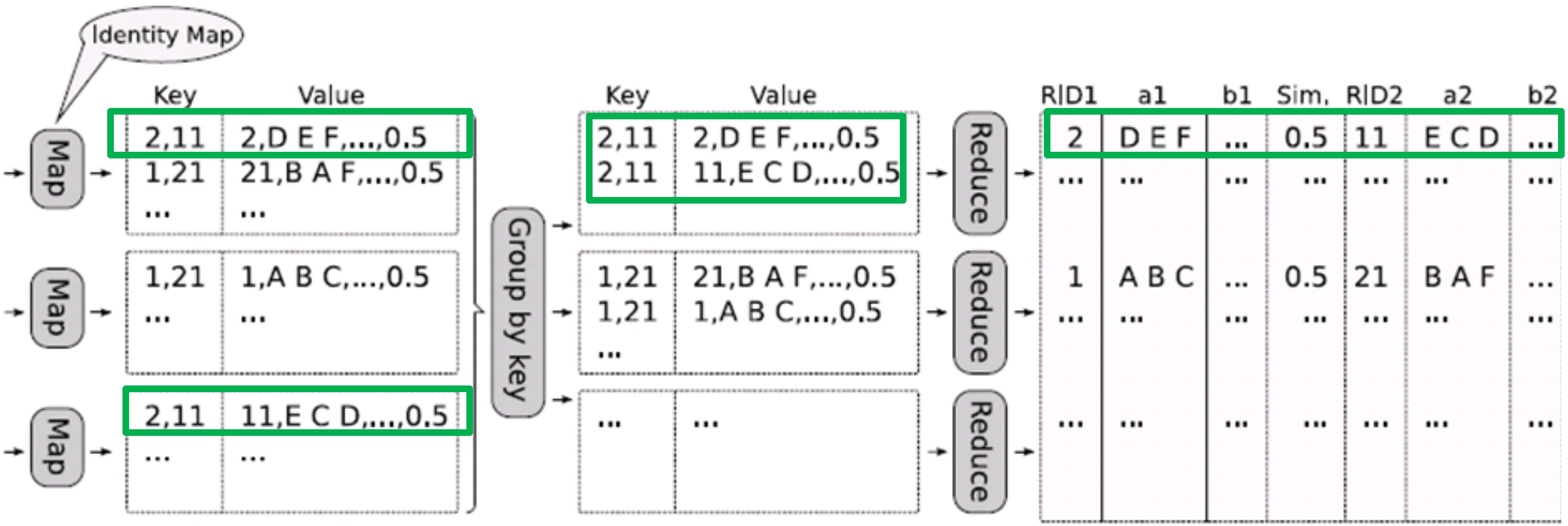
Verify similarity

Stage 3: id pairs → record pairs (phase 1)



Bring records for each id in each pair

Stage 3: id pairs → record pairs (phase 2)



Join two half filled records