Linking data without common identifiers

ISO 15926 and Semantics Conference, Sogndal, 2013-09-06

Lars Marius Garshol, <larsga@bouvet.no>
http://twitter.com/larsga

Entity resolution

Record linkage

Data matching

Identity resolution

Deduplication

Master data management

Merge/purge

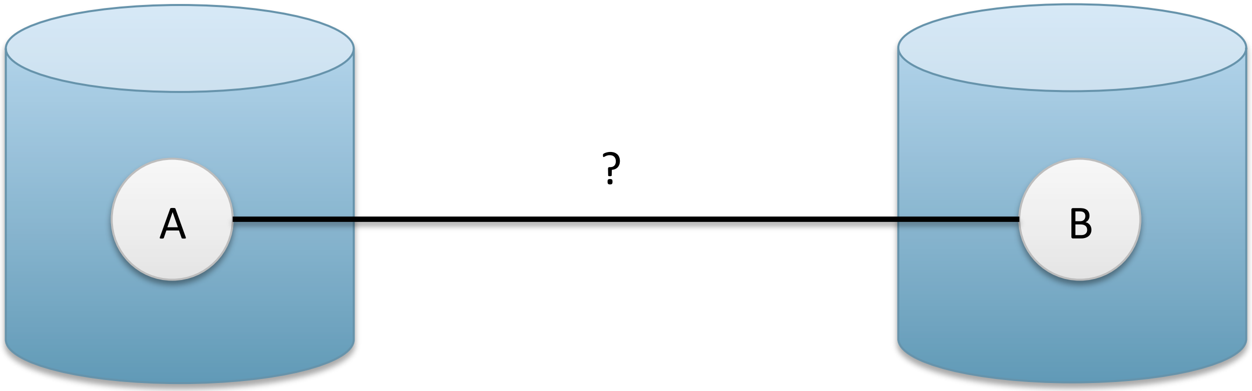
The problem

- How to tell if two different records represent the same real-world entity?

DBPEDIA	
Id	http://dbpedia.org/resource/Samoa
Name	Samoa
Founding date	1962-01-01
Capital	Apia
Currency	Tala
Area	2831
Leader name	Tuilaepa Aiono Sailele Malielegaoi

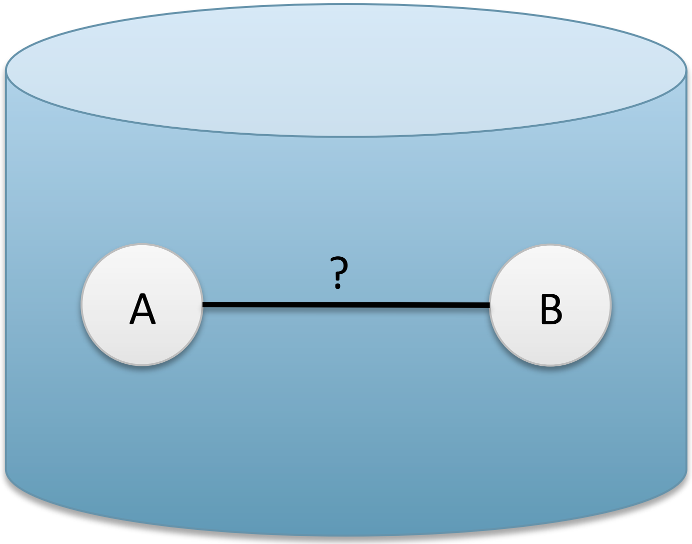
MONDIAL	
Id	17019
Name	Western Samoa
Independence	01 01 1962
Capital	Apia, Samoa
Population	214384
Area	2860
GDP	415

Linking across datasets



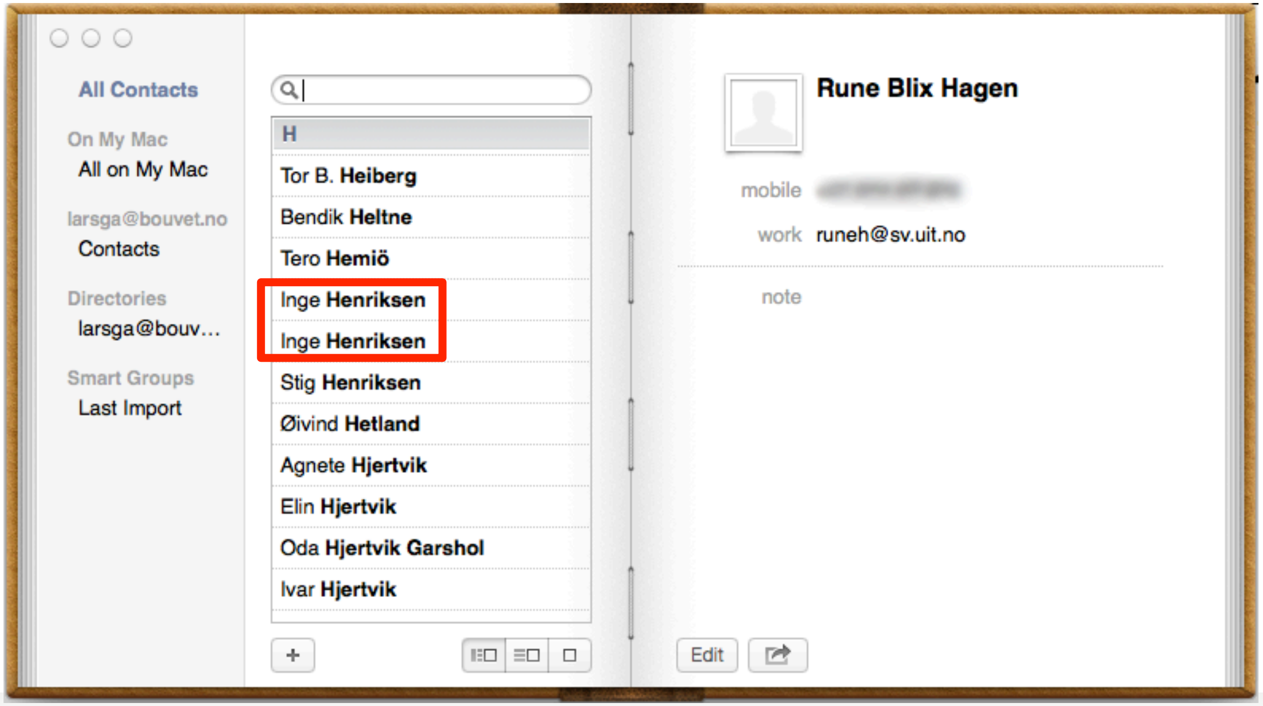
- Independent applications, for example
  - – even when unique identifiers exist, they tend to be unreliable, or missing

Within datasets



- Duplicate records in the same application
  - in different tables (poor modelling)
  - in the same table (poor UI, poor logic, sloppy entry)

Any non-trivial database contains duplicates!



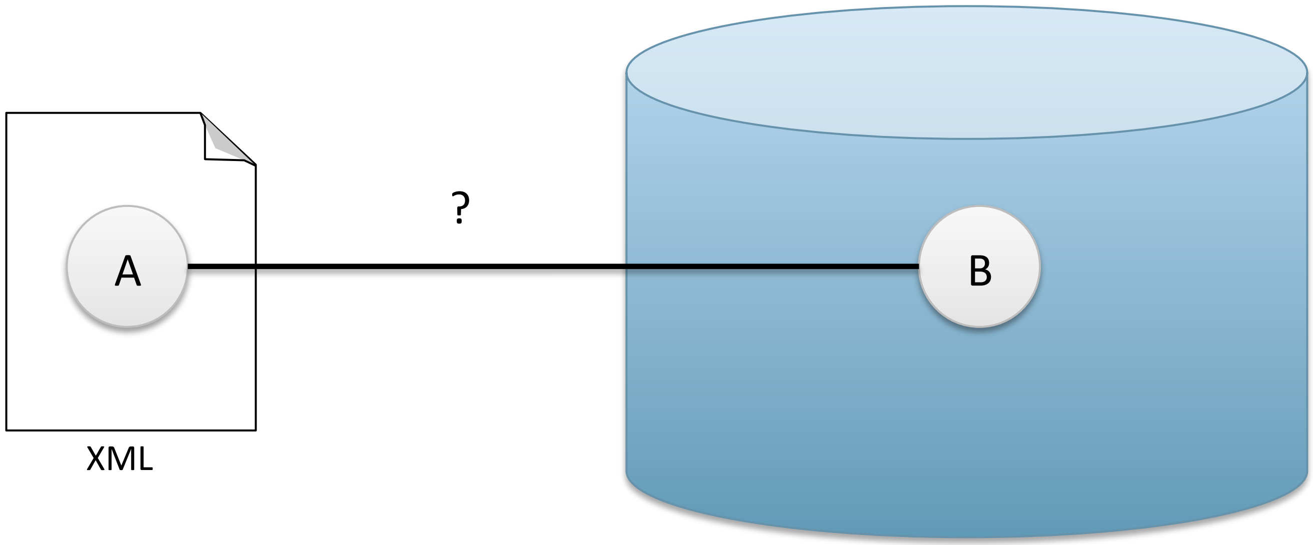
Tor B. **Heiberg**
Bendik **Heltne**
Tero **Hemiö**
Inge **Henriksen**
Inge **Henriksen**
Stig **Henriksen**
Øivind **Hetland**
Agnete **Hjertvik**
Elin **Hjertvik**
Oda **Hjertvik Garshol**
Ivar **Hjertvik**

Rune Blix Hagen

mobile [redacted]
work runeh@sv.uit.no

note

In data interchange



- Receiving data from third parties
  - do we have this record already?

A difficult problem

- It requires *$O(n^2)$* comparisons for *n* records
  - a million comparisons for 1000 records, 100 million for 10,000, ...
- Exact string comparison is not enough
  - must handle misspellings, name variants, etc
- Interpreting the data can be difficult even for a human being
  - is the address different because there are two different people, or because the person moved?
- ...

Record linkage

- Statisticians very often must connect data sets from different sources
- They call it “record linkage”
  - term coined in 1946[1)]
  - mathematical foundations laid in 1959[2)]
  - formalized in 1969 as “Fellegi-Sunter” model[3)]
- A whole subject area has been developed with well-known techniques, methods, and tools
  - these can of course be applied outside of statistics

1) http://ajph.aphapublications.org/cgi/reprint/36/12/1412
2) http://www.sciencemag.org/content/130/3381/954.citation
3) http://www.jstor.org/pss/2286061

The basic idea

Field	Record 1	Record 2	Probability
Name	acme inc	acme inc	0.9
Assoc no	177477707		0.5
Zip code	9161	9161	0.6
Country	norway	norway	0.51
Address 1	mb 113	mailbox 113	0.49
Address 2			0.5
			0.931

Standard algorithm

- n^2 comparisons for n records is unacceptable
  - must reduce number of direct comparisons
- Solution
  - produce a key from field values,
  - sort records by the key,
  - for each record, compare with n nearest neighbours
  - sometimes several different keys are produced, to increase chances of finding matches
- Downsides
  - requires coming up with a key
  - records may match even if the keys do not

String comparison

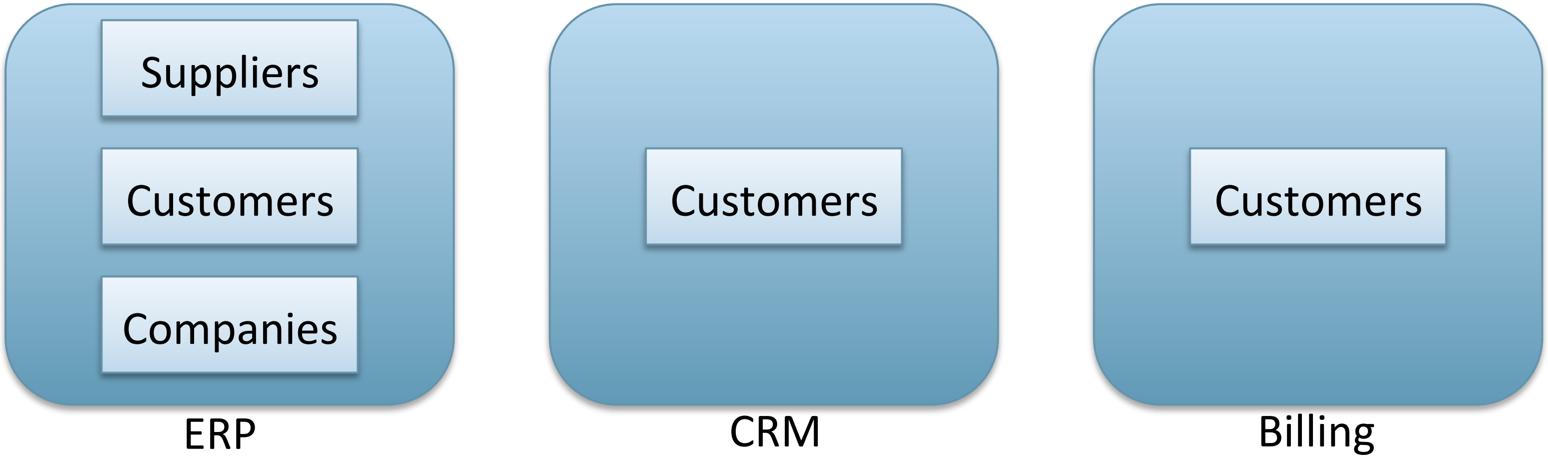
- Measure “distance” between strings
  - must handle spelling errors and name variants
  - must estimate probability that values belong to same entity despite differences
- Examples
  - John Smith ≈ Jonh Smith
  - J. Random Hacker ≈ James Random Hacker
- Many well-known algorithms have been developed
  - no one best choice, unfortunately
  - some are eager to merge, others less so
- Many are computationally expensive
  - $O(n^2)$ where n is length of string is common
  - this gets very costly when number of record pairs to compare is already high

Duke

DUplicate **K**ill**E**r

Context

- Doing a project for Hafslund where we integrate data from many sources
- Entities are duplicated both inside systems and across systems



Requirements

- Must be flexible and configurable
  - no way to know in advance exactly what data we will need to deduplicate
- Must scale to large data sets
  - CRM alone has 1.4 million customer records
  - that's 2 trillion comparisons with naïve approach
- Must have an API
  - project uses SDshare protocol everywhere
  - must therefore be able to build connectors
- Must be able to work incrementally
  - process data as it changes and update conclusions on the fly

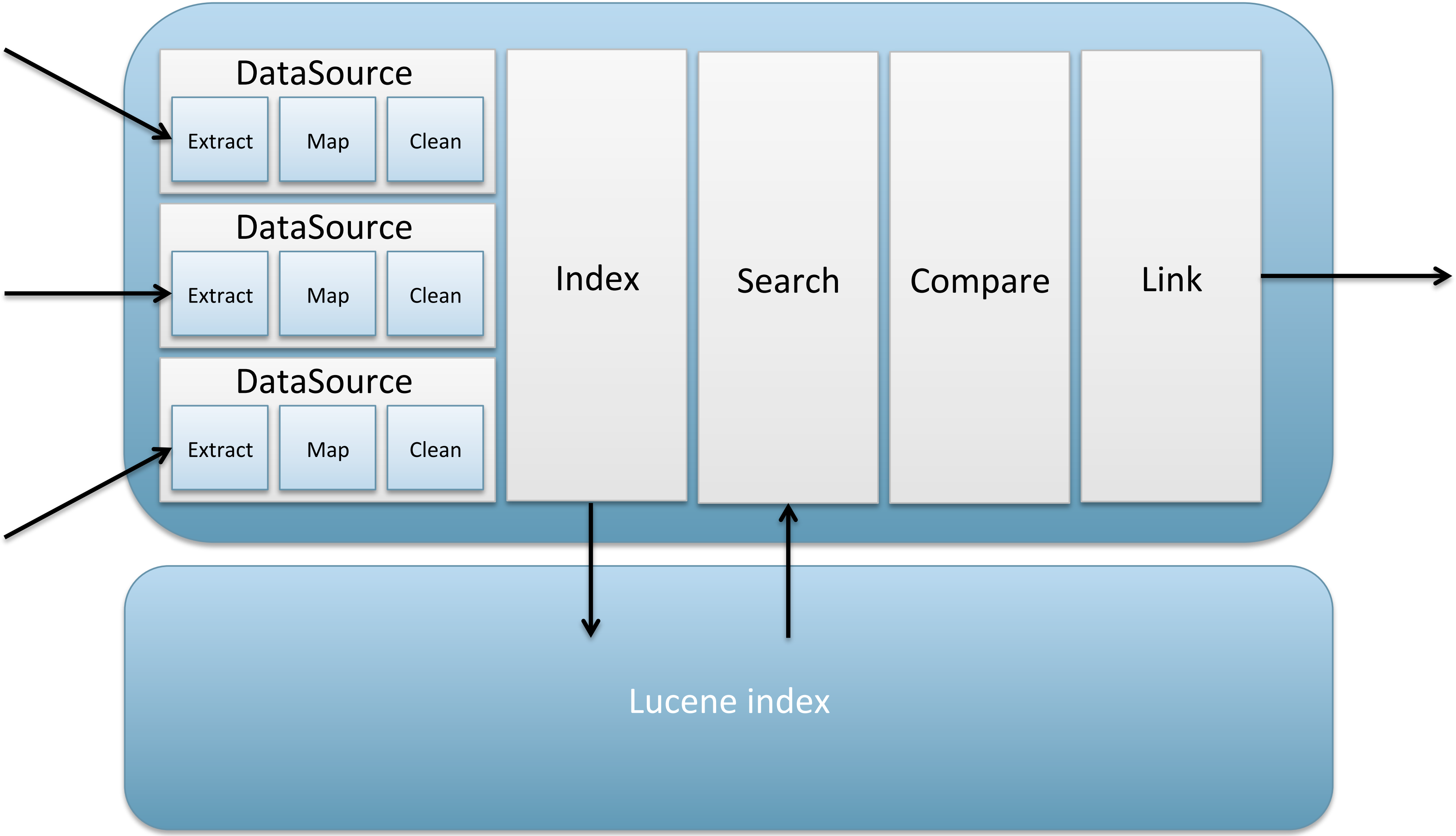
Existing tools

- Commercial tools
  - big, sophisticated, and expensive
  - seem to follow the same principles
  - presumably also effective
- Open source tools
  - generally made by and for researchers
  - nice user interfaces and rich configurability
  - advanced maths
  - architecture often not as flexible as it could be

Duke

- Java deduplication engine
  - released as open source
  - http://code.google.com/p/duke/
- Does not use key approach
  - instead indexes data with Lucene
  - does Lucene searches to find potential matches
- Still a work in progress, but
  - high performance,
  - several data sources and comparators,
  - being used for real in real projects,
  - flexible architecture

How it works


DataSource
Extract
Map
Clean
DataSource
Extract
Map
Clean
DataSource
Extract
Map
Clean
Index
Search
Compare
Link
Lucene index

Cleaning of data

Côte D’Ivoire → cote d’ivoire

Main St 113 → 113 main street

Dec 25 1973 → 1973-12-25

- Used to normalize data from sources
  - necessary to get rid of differences that don’t matter
- Makes comparing much more effective
  - pluggable in Duke
  - utilities and components already provided

Components

Data sources

- CSV
- JDBC
- Sparql
- NTriples
- <plug in your own>

Backends

- Lucene
- Naïve in-memory
- Own search engine

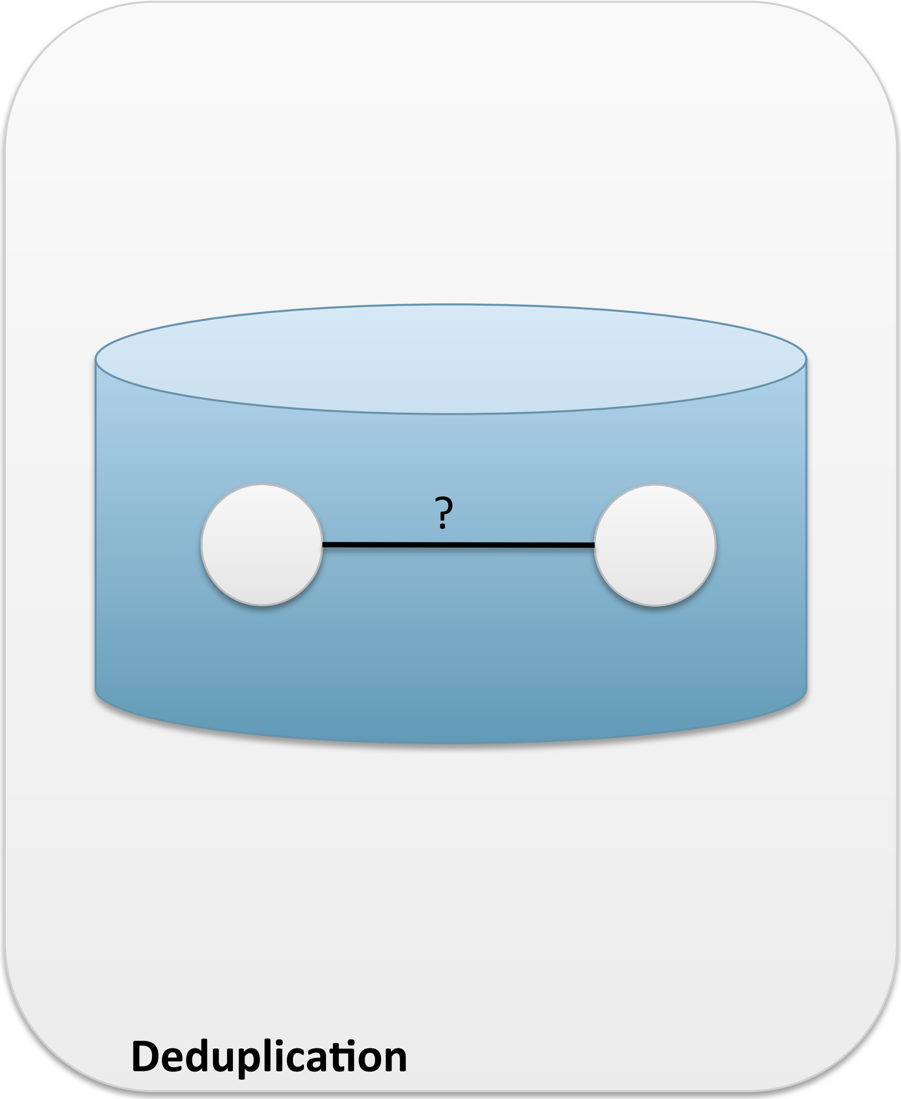
Comparators

- ExactComparator
- NumericComparator
- SoundexComparator
- Levenshtein
- Weighted Levenshtein
- JaroWinkler
- Dice coefficient
- Geoposition
- QGramComparator
- <plug in your own>

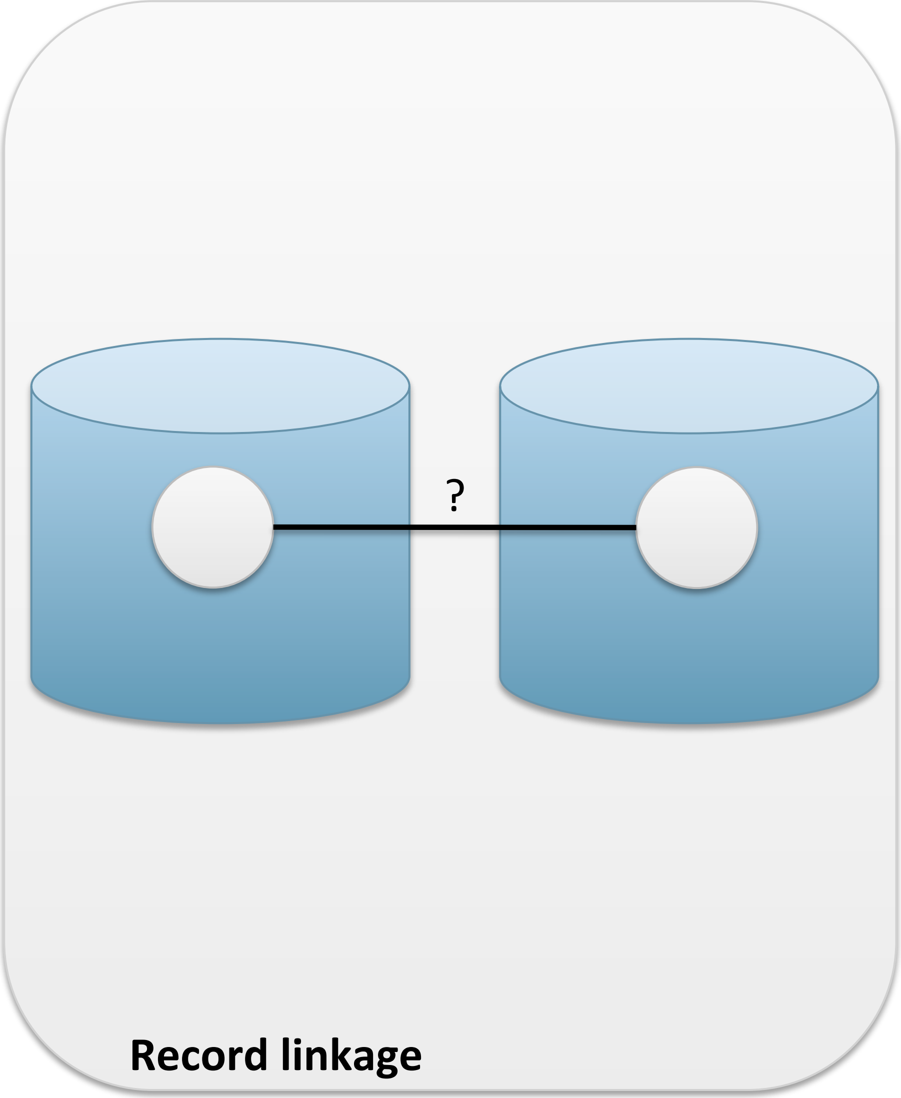
Other features

- Fairly complete command-line tool
  - with debugging support
- Very flexible API for embedding
  - everything is pluggable, from comparators to data sources to backends
- Incremental processing
- Multi-threaded
- APIs for working with collected matches

Two modes



Deduplication



Record linkage

Probabilities weigh all the evidence

Field	Record 1	Record 2	Probability	Accum.
Name	acme inc	acme inc	0.9	0.9
Assoc no	177477707		0.5	0.9
Zip code	9161	9161	0.6	0.931
Country	norway	norway	0.51	0.934
Address 1	mb 113	mailbox 113	0.3	0.857
Address 2			0.5	**0.857**

Probabilities combined with Bayes Theorem

Probabilities reduced if values don't match exactly

Linking Mondial and DBpedia

A real-world example

http://code.google.com/p/duke/wiki/LinkingCountries

Finding properties to match

- Need properties providing identity evidence
- Matching on the properties in bold below
- Extracted data to CSV for ease of use

DBPEDIA	
Id	http://dbpedia.org/resource/Samoa
Name	**Samoa**
Founding date	1962-01-01
Capital	**Apia**
Currency	Tala
Area	**2831**
Leader name	Tuilaepa Aiono Sailele Malielegaoi

MONDIAL	
Id	17019
Name	**Western Samoa**
Independence	01 01 1962
Capital	**Apia, Samoa**
Population	214384
Area	**2860**
GDP	415

Configuration – data sources

```
<group>
  <csv>
    <param name="input-file" value="dbpedia.csv"/>
    <param name="header-line" value="false"/>

    <column name="1" property="ID"/>
    <column name="2"
        cleaner="no.priv...CountryNameCleaner"
        property="NAME"/>
    <column name="3"
        property="AREA"/>
    <column name="4"
        cleaner="no.priv...CapitalCleaner"
        property="CAPITAL"/>
  </csv>
 </group>
```

```
<group>
  <csv>
    <param name="input-file" value="mondial.csv"/>

    <column name="id" property="ID"/>
    <column name="country"
        cleaner="no.priv...examples.CountryNameCleaner"
        property="NAME"/>
    <column name="capital"
        cleaner="no.priv...LowerCaseNormalizeCleaner"
        property="CAPITAL"/>
    <column name="area"
        property="AREA"/>
  </csv>
 </group>
```

Using groups tells Duke that we are linking across two data sets, not deduplicating by comparing all records against all others

Configuration – matching

```
<schema>
  <threshold>0.65</threshold>

  <property type="id">
    <name>ID</name>
  </property>
  <property>
    <name>NAME</name>
    <comparator>no.priv.garshol.duke.Levenshtein</comparator>
    <low>0.3</low>
    <high>0.88</high>
  </property>
  <property>
    <name>AREA</name>
    <comparator>AreaComparator</comparator>
    <low>0.2</low>
    <high>0.6</high>
  </property>
  <property>
    <name>CAPITAL</name>
    <comparator>no.priv.garshol.duke.Levenshtein</comparator>
    <low>0.4</low>
    <high>0.88</high>
  </property>
</schema>
```

Duke analyzes this setup and decides only NAME and CAPITAL need to be searched on in Lucene.

```
<object class="no.priv.garshol.duke.NumericComparator"
        name="AreaComparator">
  <param name="min-ratio" value="0.7"/>
 </object>
```

Result

- Correct links found: 206 / 217 (94.9%)
- Wrong links found: 0 / 12 (0.0%)
- Unknown links found: 0
- Percent of links correct 100.0%, wrong 0.0%, unknown 0.0%
- Records with no link: 25
- Precision 100.0%, recall 94.9%, f-number 0.974

Examples

Field	DBpedia	Mondial
Name	albania	albania
Area	28748	28750
Capital	tirana	tirane
Probability	**0.980**	

Field	DBpedia	Mondial
Name	kazakhstan	kazakstan
Area	2724900	2717300
Capital	astana	almaty
Probability	**0.838**	

Field	DBpedia	Mondial
Name	côte d'ivoire	cote divoire
Area	322460	322460
Capital	yamoussoukro	yamoussoukro
Probability	**0.975**	

Field	DBpedia	Mondial
Name	grande comore	comoros
Area	1148	2170
Capital	moroni	moroní
Probability	**0.440**	

Field	DBpedia	Mondial
Name	samoa	western samoa
Area	2831	2860
Capital	apia	apia
Probability	**0.824**	

Field	DBpedia	Mondial
Name	serbia	serbia and mont
Area	102350	88361
Capital	sarajevo	sarajevo
Probability	**0.440**	

Western Samoa or American Samoa?

Field	DBpedia	Mondial	Probability
Name	samoa	western samoa	0.3
Area	2831	2860	0.6
Capital	apia	apia	0.88
Probability			**0.824**

Field	DBpedia	Mondial	Probability
Name	samoa	american samoa	0.3
Area	2831	199	0.4
Capital	apia	pago pago	0.4
Probability			**0.067**

An example of failure

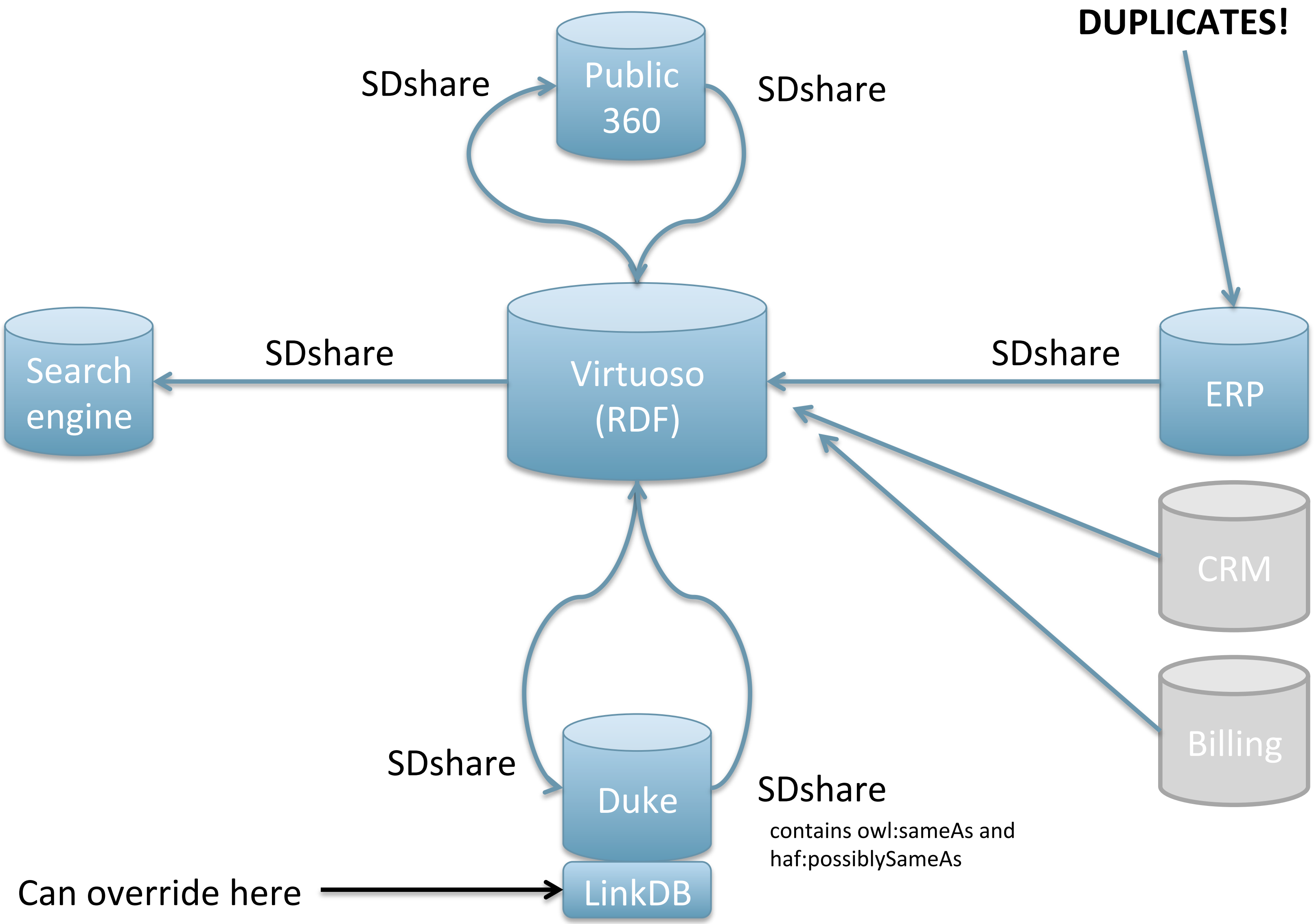
Field	DBpedia	Mondial
Name	kazakhstan	kazakstan
Area	2724900	2717300
Capital	astana	almaty
Probability	**0.838**	

- Duke doesn’t find this match
  - no tokens matching exactly
  - Lucene search finds nothing
- Detailed comparison gives correct result
  - the problem is Lucene search
- Lucene does have Levenshtein search, but
  - in Lucene 3.x it was very slow
  - therefore not enabled yet
  - thinking of adding option to enable where needed
  - Lucene 4.x has fixed the performance problem

Usage at Hafslund

Duke in real life

The big picture



contains owl:sameAs and haf:possiblySameAs

Experiences so far

- Incremental processing works fine
  - links added and retracted as data changes
- Performance not an issue at all
  - despite there being ~1.4 million records
  - requires a little bit of tuning, however
- Matching works well, but not perfect
  - data are very noisy and messy
  - biggest issue is clusters caused by generic values
  - also, matching is *hard*

Other known users

- Yoxel Portal
  - commercial cloud CRM offering
- Cityhotels.com
  - online hotel booking portal
- Easyrec
  - online recommendation engine
- More
  - there are more, but they haven't wanted to be explicitly identified

Recent work on Duke

- Version 1.1 is coming
  - adds genetic algorithm to help users create configurations
  - uses active learning so people don’t need test data
  - also adds performance profiling to make it easier to debug performance issues
- Much, much more work in the pipeline
  - tools for working with found matches
  - experiments on how to cluster matching pairs into equivalence classes
  - try different backends

Comments/questions?

Slides will appear on http://slideshare.net/larsga