

# Dissertation Draft: Constructing and Populating Ontologies from Text Documents and Database Fields

Kate Byrne

5 Sept 2007

# Outline

## Background

The Problem

My Proposal

Hypotheses to be tested

## Component Tasks

RDBMS and thesaurus conversion

Named Entity Recognition

Relation Extraction

## Query Experiments

Overall evaluation

## Timetable

## Nature of the Data

- RCAHMS corpus: 1546 annotated texts on historical sites
- Entire RCAHMS dataset:
  - 250,000 records of archaeological and architectural sites
  - 1 text document per site
  - average of three archive items per site
- Cross domain testing with similar hybrid datasets (NLS, NMS)

## Sample Document

[[EAST HEADITON]]

[NJ63SW 66 6032 3049]

A sales brochure of [4 October 1932] [[[provides particulars]]] of [Headiton ( East ) Farm] ( lot 33 , p. 26 ) including details of the [farmhouse] and the [steading] . [The house] comprised a kitchen , a milk house and four other rooms . [The steading] comprised a [barn] with loft and a [water wheel] , two [stables] for a total of eight horses , a [byre] for twenty - four cattle and ten calves , a [cart shed] , a [pigsty] , a man ' s room , a [machine shed] and a [fowl house] . In [1932] [the farm] was let to Mr [Peter Ledingham] ; at that time [it] extended to about 86 acres ( 34.8 ha ) and [its] rent was ? 81 per annum. [[Information]] from [RCAHMS] ( [JRS] ) , [28 May 1998] . [NMRS] , MS / 992/3 .

# Corpus Annotation

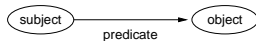
- Annotation designed to be generic to cultural heritage:
  - Who? What? Where? When?
  - isA, seeAlso, sameAs, partOf
- Overall approach is generic across all hybrid datasets
  - annotation is the only domain-specific component

# Data Access Problems

1. Complex database structure
2. Specialist terminology
3. Limited access to text content

# The Proposal

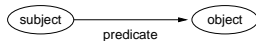
- Transform hybrid data into directed graph: *datagraph*



- Components:
  - structured database fields
  - text documents
  - domain thesauri
- Datagraph characteristics:
  - graph of binary relations, expressed as RDF triples
  - nodes are NEs, thesaurus terms, database field contents
  - edges are predicates, database field names
  - not necessarily consistent across entire extent
- *(Quite a number of dead ends, not covered today)*

# The Proposal

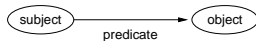
- Transform hybrid data into directed graph: *datagraph*



- Components:
  - structured database fields
  - text documents
  - domain thesauri
- Datagraph characteristics:
  - graph of binary relations, expressed as RDF triples
  - nodes are NEs, thesaurus terms, database field contents
  - edges are predicates, database field names
  - not necessarily consistent across entire extent
- *(Quite a number of dead ends, not covered today)*

# The Proposal

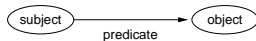
- Transform hybrid data into directed graph: *datagraph*



- Components:
  - structured database fields
  - text documents
  - domain thesauri
- Datagraph characteristics:
  - graph of binary relations, expressed as RDF triples
  - nodes are NEs, thesaurus terms, database field contents
  - edges are predicates, database field names
  - not necessarily consistent across entire extent
- *(Quite a number of dead ends, not covered today)*

# The Proposal

- Transform hybrid data into directed graph: *datagraph*



- Components:
  - structured database fields
  - text documents
  - domain thesauri
- Datagraph characteristics:
  - graph of binary relations, expressed as RDF triples
  - nodes are NEs, thesaurus terms, database field contents
  - edges are predicates, database field names
  - not necessarily consistent across entire extent
- (*Quite a number of dead ends, not covered today*)

# The Hypotheses

1. Text can be adequately realised as graph of binary relations
2. Solves the three problems (structure, terminology, text)
3. Extra gains:
  - a) use graph locality to deal with inconsistency
  - b) summarise intermediate results using graph characteristics
  - c) discovery of latent relationships

# Evaluation Plan

- Evaluate significant tasks (NER, RE) using standard metrics
- Overall evaluation by query experiments on RDF graph

## Questions Not Being Covered

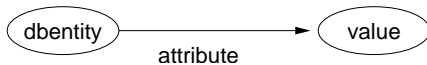
- Will it scale?
- How easy is data management?
- Are there other benefits?
  - NLG: producing simpler text; using other languages
  - schema flexibility
  - interface design possibilities
  - dataset linking
- Will it catch on?
- *Is the game worth the candle?*

## Multiple Component Tasks

- RDBMS to RDF conversion
- Conversion of thesauri
- Text to RDF conversion:
  - Step 1. NER – identify and classify node terms
  - Step 2. RE – relations are between NEs
- Designed set of predicates; not learnt or generated

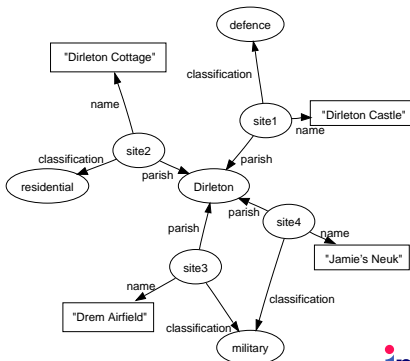
# RDBMS to RDF

- Basic procedure straightforward:



SITE

siteNo	name	parish	classification
1	Dirleton Castle	Dirleton	defence
2	Dirleton Cottage	Dirleton	residential
3	Drem Airfield	Dirleton	military
4	Jamie's Neuk	Dirleton	military



## Variations on the Basic Procedure

- Use “entity” nodes including primary key instead of b-nodes
- Ignore empty fields
- Replace concatenated keys with single-column surrogates
- Manual schema design - group similar attributes together
- “Pre-join” tables to eliminate redundant nodes
- Theoretical maximum: 235 million triples (rows x cols)
- Actual: 15.4 million (without schema)

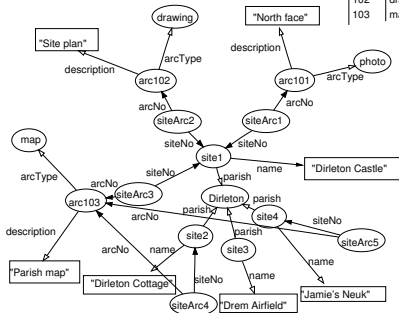
## Variations on the Basic Procedure

- Use “entity” nodes including primary key instead of b-nodes
- Ignore empty fields
- Replace concatenated keys with single-column surrogates
- Manual schema design - group similar attributes together
- “Pre-join” tables to eliminate redundant nodes
- Theoretical maximum: **235 million** triples (rows x cols)
- Actual: **15.4 million** (without schema)

# Dealing with Relational Joins

SITE			SITE-ARC		
siteNo	name	parish	siteArcNo	siteNo	arcNo
1	Dirleton Castle	Dirleton	1	1	101
2	Dirleton Cottage	Dirleton	2	1	102
3	Drem Airfield	Dirleton	3	1	103
4	Jamie's Neuk	Dirleton	4	2	103
			5	4	103

ARCHIVE		
arcNo	arcType	description
101	photo	North face
102	drawing	Site plan
103	map	Parish map



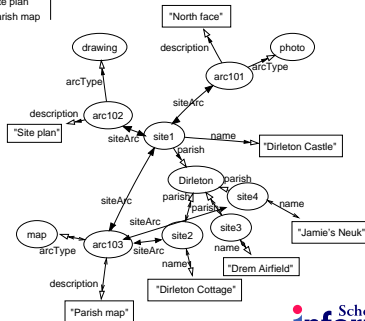
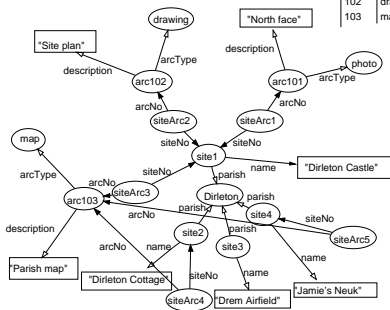
# Dealing with Relational Joins

SITE		
siteNo	name	parish
1	Dirleton Castle	Dirleton
2	Dirleton Cottage	Dirleton
3	Drem Airfield	Dirleton
4	Jamie's Neuk	Dirleton

SITE-ARC		
siteArcNo	siteNo	arcNo
1	1	101
2	1	102
3	1	103
4	2	103
5	4	103

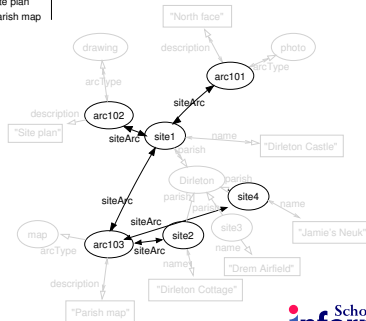
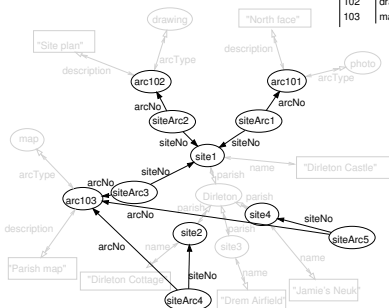
ARCHIVE		
arcNo	arcType	description
101	photo	North face
102	drawing	Site plan
103	map	Parish map



# Dealing with Relational Joins

SITE			SITE-ARC		
siteNo	name	parish	siteArcNo	siteNo	arcNo
1	Dirleton Castle	Dirleton	1	1	101
2	Dirleton Cottage	Dirleton	2	1	102
3	Drem Airfield	Dirleton	3	1	103
4	Jamie's Neuk	Dirleton	4	2	103
			5	4	103

ARCHIVE		
arcNo	arcType	description
101	photo	North face
102	drawing	Site plan
103	map	Parish map

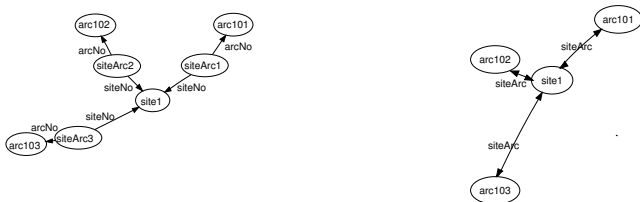


# Dealing with Relational Joins

SITE			SITE-ARC		
siteNo	name	parish	siteArcNo	siteNo	arcNo
1	Dirleton Castle	Dirleton	1	1	101
2	Dirleton Cottage	Dirleton	2	1	102
3	Drem Airfield	Dirleton	3	1	103
4	Jamie's Neuk	Dirleton	4	2	103
			5	4	103

ARCHIVE		
arcNo	arcType	description
101	photo	North face
102	drawing	Site plan
103	map	Parish map



## Mapping to URIs

- In RDF everything is either a resource or a literal
- Resources are identified by URIs (which may be URLs)
  - e.g. “Kate Byrne” might be “<http://homepages.inf.ed.ac.uk/s0233752/>”
- When to use which?
  - “#2 is 6' long, photographed in b&w”, “NT27SE”, “Edinburgh”*
- Use blank nodes? (how do we correlate *#value* nodes?)
- Metadata in URI: entire db provenance? (Berners-Lee, 2006)
- Co-reference issues (whose “Edinburgh” do we use?)
- Tremendous data bloat
- We have to decode the URI to get the actual value we want

## Mapping to URIs

- In RDF everything is either a resource or a literal
- Resources are identified by URIs (which may be URLs)
  - e.g. “Kate Byrne” might be “<http://homepages.inf.ed.ac.uk/s0233752/>”
- When to use which?
  - “#2 is 6' long, photographed in b&w”, “NT27SE”, “Edinburgh”
  - Use blank nodes? (how do we correlate *#value* nodes?)
  - Metadata in URI: entire db provenance? (Berners-Lee, 2006)
  - Co-reference issues (whose “Edinburgh” do we use?)
  - Tremendous data bloat
  - We have to decode the URI to get the actual value we want

## Mapping to URIs

- In RDF everything is either a resource or a literal
- Resources are identified by URIs (which may be URLs)  
e.g. “Kate Byrne” might be “<http://homepages.inf.ed.ac.uk/s0233752/>”
- When to use which?  
*“#2 is 6’ long, photographed in b&w”, “NT27SE”, “Edinburgh”*
- Use blank nodes? (how do we correlate *#value* nodes?)
- Metadata in URI: entire db provenance? (Berners-Lee, 2006)
- Co-reference issues (whose “Edinburgh” do we use?)
- Tremendous data bloat
- We have to decode the URI to get the actual value we want

# Schema Design

- Metadata in schema relations, rather than in URI string
- Simplify datagraph by moving complexity to schema
- In practice this means:
  - use `rdf:type`, `rdfs:domain`, `rdfs:range`
  - group RDBMS attributes to small set of predicates:  
`loc`, `ind`, `id`, `sitename`, `date`, `sitetype`, `agent`, `desc`
  - up to four sub-properties for some of these

# Named Entity Recognition

- NE classes:
  - `org, persname, role, sitetype, artefact, sitename, place, address, period, date, event`
- event subclasses:
  - `survey, excavation, find, visit, description, creation, alteration`
- Using supervised ML tagging, with CandC NE classifier
- NE subclasses not dealt with so far

# Named Entity Recognition

- NE classes:  
org, persname, role, sitetype, artefact, sitename, place,  
address, period, date, event
- event subclasses:  
survey, excavation, find, visit, description, creation,  
alteration
- Using supervised ML tagging, with CandC NE classifier
- NE subclasses not dealt with so far

# Named Entity Recognition

- NE classes:  
org, persname, role, sitetype, artefact, sitename, place,  
address, period, date, event
- event subclasses:  
survey, excavation, find, visit, description, creation,  
alteration
- Using supervised ML tagging, with CandC NE classifier
- NE subclasses not dealt with so far

## Nested Entities

- RCAHMS corpus: 10% of NEs have others nested within them
- Up to three levels of nesting in corpus, e.g.  
[[[Edinburgh]<sup>PLACE</sup> University]<sup>ORG</sup> Library]<sup>ORG</sup>
- If nested NEs not found, relations involving them are lost
- Problem for standard classifier: one label per token
- Possible approaches:
  - multilabel tagging (McDonald et al, 2005)
  - joined label tagging (Alex et al, 2007)
  - my approach: multi-word tokenisation
- Compared against single token tagging on outermost NEs

## Nested Entities

- RCAHMS corpus: 10% of NEs have others nested within them
- Up to three levels of nesting in corpus, e.g.  
 [[[Edinburgh]<sup>PLACE</sup> University]<sup>ORG</sup> Library]<sup>ORG</sup>
- If nested NEs not found, relations involving them are lost
- Problem for standard classifier: one label per token
- Possible approaches:
  - multilabel tagging (McDonald et al, 2005)
  - joined label tagging (Alex et al, 2007)
  - my approach: multi-word tokenisation
- Compared against single token tagging on outermost NEs

## Multi-word Tokenisation

when O

when\_Edinburgh O

when\_Edinburgh\_University O

Edinburgh PLACE

Edinburgh\_University ORG

Edinburgh\_University\_Library ORG

University O

University\_Library O

University\_Library\_was O

Library O

Library\_was O

Library\_was\_built O

## NER Results

Using CandC	P %	R %	F %	Correct NEs
Best single-token run	76.98	75.18	76.07	18,379
Multi-token, unweighted	87.70	66.79	75.83	18,322
Multi-token, smoothed	78.43	75.91	77.15	20,825

- Precision improved at expense of recall – good!
- Performance comparable to method that only finds single level
- Smoothed model outputs 13% extra NEs
- Main drawback: much slower

## Results Using Untuned Classifier

Using ZLMaxent	P %	R %	F %
single-word tokens	41.06	48.56	44.49
multi-word tokens	78.59	46.90	58.75

- Simple experiment using classifier not optimised for NER
- Performance significantly higher with multi-tokens
- Bias towards precision confirmed
- Very fast

## Relation Extraction - Work in Progress

- Binary relations:

`isA, sameAs, seeAlso, partOf, hasLocation, hasPeriod`

- `eventRel` has higher arity:

`eventRel(eventType, eventPatient, eventDate, eventAgent, eventRole, eventResult)`

- will be converted into binary relations

‘‘A sales brochure of 4 October 1932 provides particulars of Headiton ( East ) Farm ( lot 33 , p. 26 )’’

- Using ZLMaxent for tagging (may use svmlight also)
- Detect relations between pairs of NEs - across document
- Model each predicate separately

## Relation Extraction - Work in Progress

- Binary relations:

isA, sameAs, seeAlso, partOf, hasLocation, hasPeriod

- eventRel has higher arity:

eventRel(eventType, eventPatient, eventDate, eventAgent, eventRole, eventResult)

- will be converted into binary relations

‘‘A sales brochure of 4 October 1932 provides particulars of Headiton ( East ) Farm ( lot 33 , p. 26 )’’

- Using ZLMaxent for tagging (may use svmlight also)
- Detect relations between pairs of NEs - across document
- Model each predicate separately

## Relation Extraction - Work in Progress

- Binary relations:

isA, sameAs, seeAlso, partOf, hasLocation, hasPeriod

- eventRel has higher arity:

eventRel(eventType, eventPatient, eventDate, eventAgent, eventRole, eventResult)

- will be converted into binary relations

‘‘A sales brochure of 4 October 1932 provides particulars of Headiton ( East ) Farm ( lot 33 , p. 26 )’’

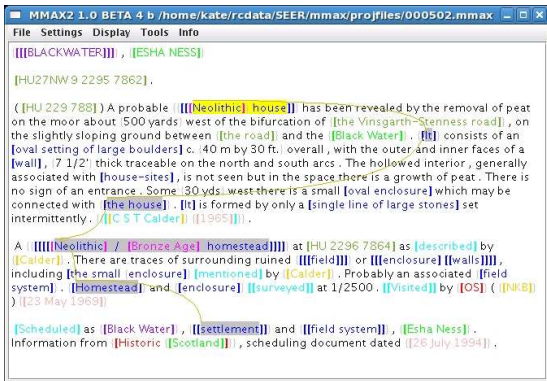
- Using ZLMaxent for tagging (may use svmlight also)
- Detect relations between pairs of NEs - across document
- Model each predicate separately

## Relation Extraction - Work in Progress

- Binary relations:
    - isA, sameAs, seeAlso, partOf, hasLocation, hasPeriod
  - eventRel has higher arity:
    - eventRel(eventType, eventPatient, eventDate, eventAgent, eventRole, eventResult)
      - will be converted into binary relations
- ‘A sales brochure of 4 October 1932 provides particulars of Headiton ( East ) Farm ( lot 33 , p. 26 )’
- Using ZLMaxent for tagging (may use svmlight also)
  - Detect relations between pairs of NEs - across document
  - Model each predicate separately

## Annotation Issues

- IAA figures not yet done – will be low?
- Could “fix” annotation scheme (hard) but must reflect reality
- Co-reference/pronoun anaphora not yet done



## Evaluation Plan

- NER performance evaluated using standard P-R-F measures
- Will do same for RE
- Cross domain NER, RE performance if possible (NLS, NMS)
- Overall evaluation of hypotheses by query experiments:
  - a) compare graph querying with standard RDBMS queries
  - b) compare queries with and without relations derived from text
  - c) demonstrate that intermediate results summaries are possible
- Formal metrics difficult to define
- Probably insufficient time for cross domain query evaluation

## Evaluation Plan

- NER performance evaluated using standard P-R-F measures
- Will do same for RE
- Cross domain NER, RE performance if possible (NLS, NMS)
- Overall evaluation of hypotheses by query experiments:
  - a) compare graph querying with standard RDBMS queries
  - b) compare queries with and without relations derived from text
  - c) demonstrate that intermediate results summaries are possible
- Formal metrics difficult to define
- Probably insufficient time for cross domain query evaluation

## Evaluation Plan

- NER performance evaluated using standard P-R-F measures
- Will do same for RE
- Cross domain NER, RE performance if possible (NLS, NMS)
- Overall evaluation of hypotheses by query experiments:
  - a) compare graph querying with standard RDBMS queries
  - b) compare queries with and without relations derived from text
  - c) demonstrate that intermediate results summaries are possible
- Formal metrics difficult to define
- Probably insufficient time for cross domain query evaluation

## Datagraph v RDBMS

- Use SPARQL over RDF datagraph; SQL over RDBMS
- Compare queries considered routine in RDBMS
- Attempt to demonstrate finding of latent relationships:
  - RDBMS only able to return known attributes
  - can datagraph produce result without knowing predicate?
  - uses locality – prefer shorter path lengths
- If time: compare SQL over RDF with SPARQL. But...
  - SPARQL is being improved all the time
  - summarisation may *require* SQL; but not a formal evaluation

## Datagraph v RDBMS

- Use SPARQL over RDF datagraph; SQL over RDBMS
- Compare queries considered routine in RDBMS
- Attempt to demonstrate finding of latent relationships:
  - RDBMS only able to return known attributes
  - can datagraph produce result without knowing predicate?
  - uses locality – prefer shorter path lengths
- If time: compare SQL over RDF with SPARQL. But...
  - SPARQL is being improved all the time
  - summarisation may *require* SQL; but not a formal evaluation

## Basic v Text-enhanced Datagraph

- Attempt to show improved results from extended datagraph:
  - queries that fail or produce partial results in RDBMS...
  - ...but give valid info from datagraph, using text relations
- Demonstrate capacity for summarisation over datagraph
- Demonstrate possibility of “guided queries”
- Cross domain queries (NLS, NMS) if time

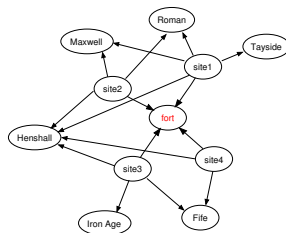
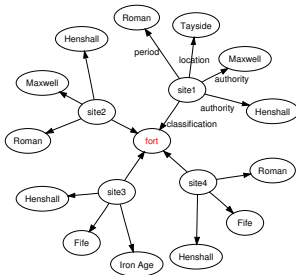
## Basic v Text-enhanced Datagraph

- Attempt to show improved results from extended datagraph:
  - queries that fail or produce partial results in RDBMS...
  - ...but give valid info from datagraph, using text relations
- Demonstrate capacity for summarisation over datagraph
- Demonstrate possibility of “guided queries”
- Cross domain queries (NLS, NMS) if time

## Basic v Text-enhanced Datagraph

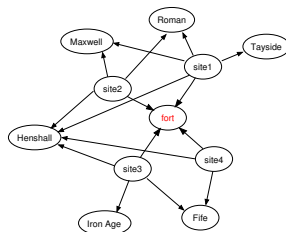
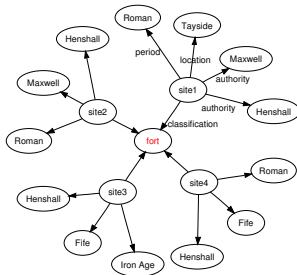
- Attempt to show improved results from extended datagraph:
  - queries that fail or produce partial results in RDBMS...
  - ...but give valid info from datagraph, using text relations
- Demonstrate capacity for summarisation over datagraph
- Demonstrate possibility of “guided queries”
- Cross domain queries (NLS, NMS) if time

# Graph Summarisation



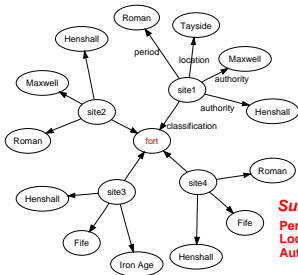
1. identify sites with required classification (“fort”)
2. summarise over *other* attributes of those sites
3. let user pick from summary to refine query: guided query

# Graph Summarisation



1. identify sites with required classification (“fort”)
2. summarise over *other* attributes of those sites
3. let user pick from summary to refine query: guided query

# Graph Summarisation

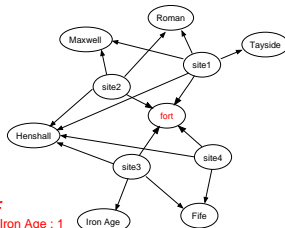


### Summary for "fort":

**Period** – Roman : 3, Iron Age : 1

**Location** – File : 2, Tayside : 1

**Authority** – Henshall : 4, Maxwell : 2



1. identify sites with required classification ("fort")
2. summarise over *other* attributes of those sites
3. let user pick from summary to refine query: guided query

## Work Still To Do

Relation Extraction work	by end Oct 2007
Finish database conversion, integrate data	by end Nov 2007
Query experiments	by end Jan 2008
Finish thesis	by end Mar 2008