Improving content discovery through combining linked data and data mining techniques

Ross Fenning

May 2, 2016

CONTENTS

CHAPTER

ONE

INTRODUCTION

Media companies produce ever larger numbers of articles, videos, podcasts, games, etc. – commonly collectively known as "content". A successful content-producing website not only has to develop systems to aid producing and publishing that content, but there are also demands to engineer effective mechanisms to aid consumers in finding that content.

Approaches used in industry include providing a text-based search, hierarchical categorisation (and thus navigation thereof) and even more tailored recommended content based on past behaviour or content enjoyed by friends (or sometimes simply other consumers who share your preferences).

1.1 Problems

There are several technical and conceptual problems with building effective content discovery mechanisms, including:

- Large organisations can have content across multiple content management systems, in differing formats and data models. Organisations face a large-scale enterprise integration problem simply trying to gain a holistic view of all their content.
- Many content items are in fairly opaque formats, e.g. video content may be stored as audio-visual binary data with minimal metadata to display on a containing web page. Video content producers may not be motivated to provide data attributes that might ultimately be most useful in determining if a user will enjoy the video.
- Content is being published continuously, which means any search or discovery system needs to keep up with content as it is published and process it into the appropriate data structures. Any machine learning previously performed on the data set may need to be re-run.

1.2 Hypothesis

The following hypotheses are proposed for gaining new insights about an organisation's diverse corpus of content:

- Research and software tools around the concept of *Linked Data* can aid us in rapidly acquiring a broad view (perhaps at the expense of depth) of an organisation's content whilst also providing a platform for simple enrichment of that content's metadata.
- We can establish at least a naïve mapping of an RDF graph representing a content item to an attribute set suitable for data mining. With such a mapping, we can explore applying machine learning – particularly unsupervised learning – across an organisation's whole content corpus.
- Linked Data and Semantic Web *ontologies* and models available can provide data enrichment beyond attributes and keywords explicitly avaiable within content data or metadata.
- We can adapt established machine learning approaches such as clustering for data published continuously in real time.
- Many content-producers currently enrich their web pages with small amounts of semantic metadata to provide better presentation of that content as it is shared on social media. This enables simple collection of a full breadth of content with significantly less effort than direct integration with content management systems.

CHAPTER

TWO

BACKGROUND

This chapter discusses some of the existing research and technologies around machine learning, RDF and combining them. It also covers some of the advantages of using linked data and RDF in an enterprise setting and what tools and approaches are well-defined enough that a corporation could build on top of them rapidly.

Data mining activities such as machine learning rely on structuring data as *feature sets*[2] – a set or vector of properties or attributes that describe a single entity. The process of *feature extraction* generates such feature sets from raw data and is a necessary early phase for many machine learning activities.

The rest of this chapter will show:

1. that extracting feature sets from RDF[1] graphs can be done elegantly and follows naturally from some previous work in this area; and
2. that the RDF graph is a suitable and even desirable data model for content metadata in terms of acquiring, enriching and even transforming that data ahead of feature extraction.

2.1 Data Mining

TODO

2.2 RDF and Feature Extraction

The RDF graph is a powerful model for metadata based on representing knowledge as a set of subject-predicate-object *triples*. The query language, SPARQL, gives us a way to query the RDF graph structure using a declarative pattern and return a set of all variable bindings that satisfy that pattern.

For example, the SPARQL query in Listings 2.1 queries an RDF graph that contains contact information and returns the names and email address of all "Person" entities therein.

[1] http://www.w3.org/TR/PR-rdf-syntax/

Notably, Kiefer, Bernstein and Locher[8] proposed a novel approach called SPARQL-ML – an extension to the SPARQL[18] query language with new keywords to facilitate both generating and applying models. This means that the system capable of parsing and running standard queries must also run machine learning algorithms.

Their work involved developing an extension to the SPARQL query engine for *Apache Jena*[2] that integrates with systems such as *Weka*[3]. A more suitable software application for enterprise use might focus solely on converting RDF graphs into a neutral data structure that can plug into arbitrary data mining algorithms.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?email
WHERE {
  ?person a foaf:Person.
  ?person foaf:name ?name.
  ?person foaf:mbox ?email.
}
```

Listing 2.1 : Example SPARQL query for people's names and email addresses

If we consider an RDF graph, g, to be expressed as a set of triples:

$$(s, p, o) \in g$$

this query could then be expressed as function $f : G \to (S \times S)$ where G is the set of all possible RDF graphs and S is a set of all possible strings. This allows the result of the SPARQL query to be expressed as a set of all SELECT variable bindings that satisfy the WHERE clause:

$$q(g, n, e) = \exists p.(p, type, Person) \in g \ \wedge (p, name, n) \in g \wedge (p, mbox, e) \in g$$

$$g \in G \models f(g) = \{(n, e) \subseteq (S \times S) \mid q(g, n, e)\}$$

This could be generalised to express a given feature set as vector $(a_1, a_2, ..., a_n)$:

$$g \in G \models (a_1, a_2, ..., a_n) \in f(g)$$

and in the case where all $a_k \in f(g)$ are literal (e.g. string or numeric) values, we can thus consider a given SPARQL query to be specific function capable of feature extraction from any RDF graph into sets of categorical or numeric features.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?topic
WHERE {
  ?article rdf:about ?topic .
}
```

Listing 2.2 : SPARQL query to determine what

[2]https://jena.apache.org/
[3]http://www.cs.waikato.ac.nz/ml/weka/

This might allow a query that extracts a country's population, GDP, etc. provide feature extraction for learning patterns in economics, for example. However, this is limited to features derived from single-valued predicates with literal-valued ranges. It is not clear how to formulate a query that expresses whether or not a content item is about a given topic.

In the RDF model, it would be more appropriate to use a query like that in Listings 2.2 where for a given ?*article* identified by URI, we can get a list of URIs identifying concepts which the article mentions. Such a query might be expressed as function $f' : G \to \mathcal{P}(U)$ where U is set of all URIs such that:

$$g \in G \models f'(g, uri) = \{t \mid (uri, about, t) \in g\}$$

An approach of generating attributes for a given resource was proposed by Paulheim and Fürnkranz[12]. They defined specific SPARQL queries and provided case study evidence for the effectiveness of each strategy.

Their work focused on starting with relational-style data (e.g. from a relational database) and using *Linked Open Data* to identify entities within literal values in those relations and generated attributes from SPARQL queries over those entities.

For a large content-producer, there is a more general problem where many content items do not have a relational representation and the content source is a body of text or even a raw HTML page. However, the feature generation from Paulheim and F urnkranz proves to be a promising strategy given we can acquire an RDF graph model for content items in the first place.

2.3 RDF in the enterprise

2.3.1 Data Fragmentation in Organisations

2.3.2 Enterprise Integration

2.3.3 Organisational Difficulties with Enterprise Integration

2.3.4 Linked Enterprise Data

CHAPTER

THREE

SYSTEM DESIGN

In this chapter, a system is inductively derived and concretely design to make use of multiple strategies for:

1. gathering (meta)data about all of an organisations content items;
2. extracting metadata not explicitly modelled in source content management systems;
3. further enriching that metadata with information not explicitly present in the content item itself; and
4. applying machine learning to that content metadata to gain new insights about that content.

Initially, a business context is described to produce a design for a system that could be a applied within a media or content-producing organisation. This context will guide all design decisions.

3.1 Context

TODO

3.2 Use Cases

3.3 Technical Architecture

The use cases depicted and described in Section 3.2 naturally lead to an application with two distinct interfaces: notification of new content to consider for data mining and retrieval of information about content clusters created to date.

Figure 3.2 shows a high-level view of a "Content Miner" software application that provides both interfaces. A set of "notifier" applications can be created to connect different content production and management systems to the data mining system such that it is notified when new content is created. The notification need only be an *IRI* that uniquely identifies that content item.

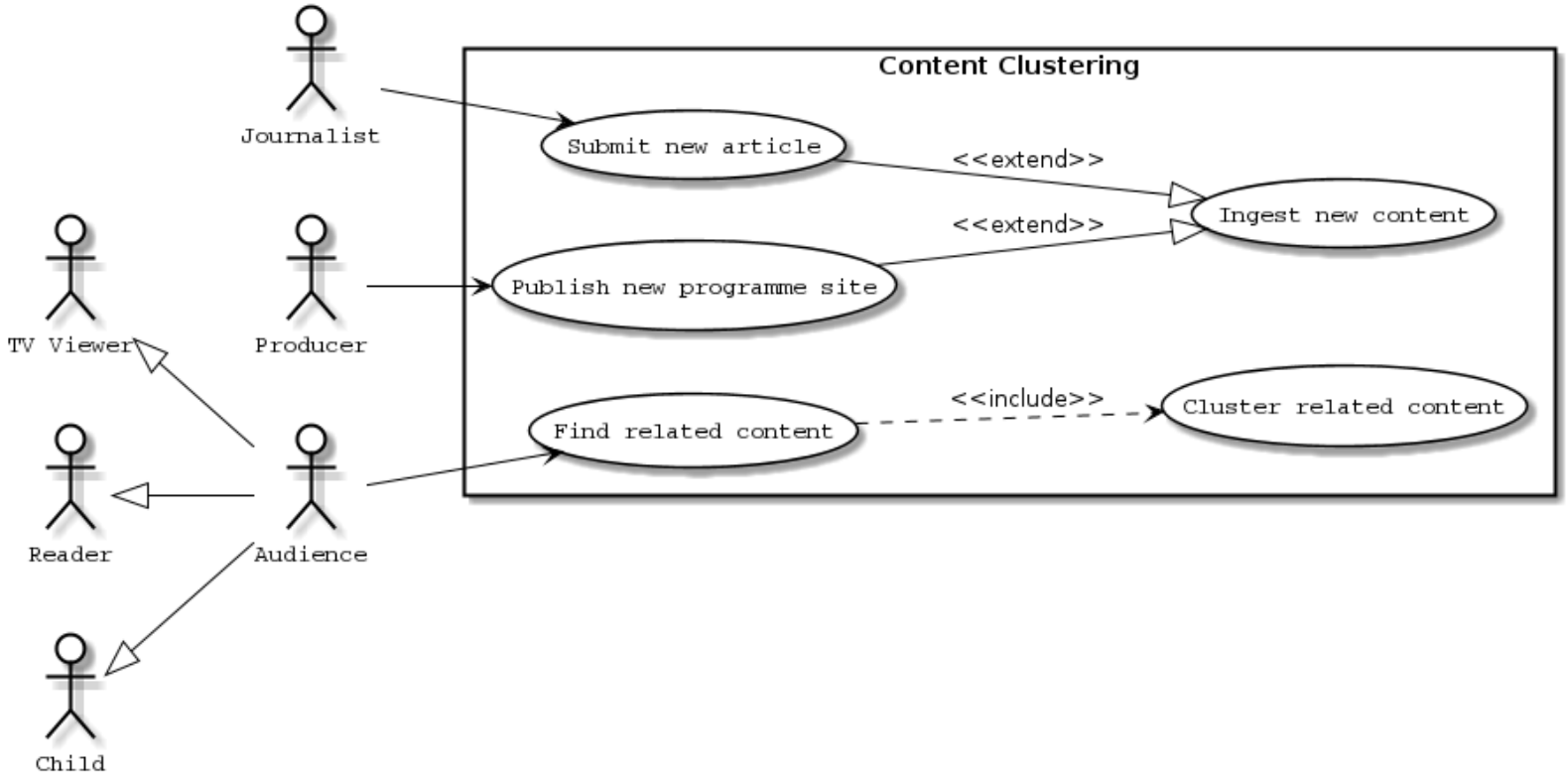

Figure 3.1: Use case diagram for content clustering system

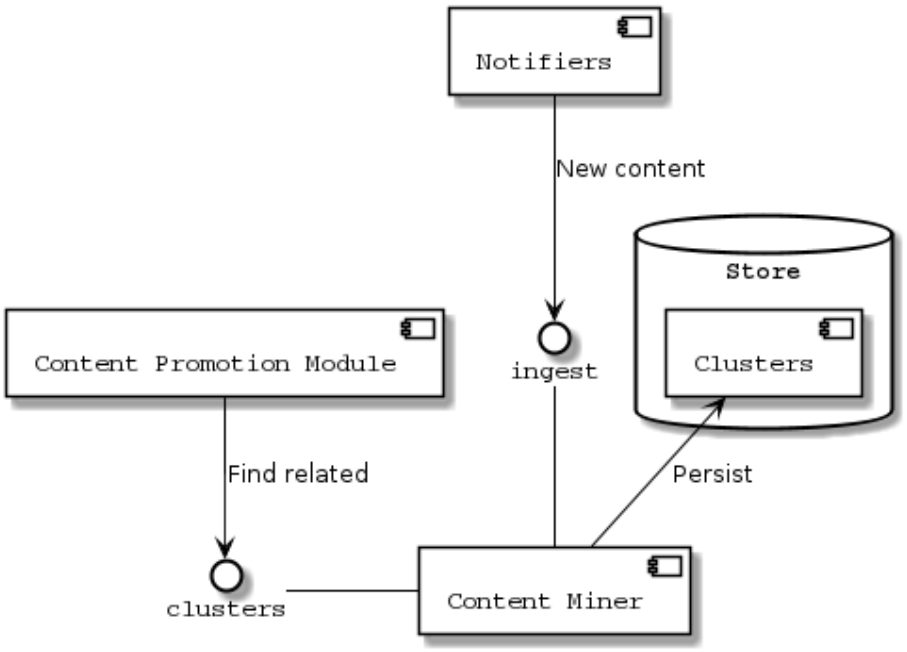

Figure 3.2: High-level component diagram with interfaces for each use case

A *Content Promotion Module* is also depicted as an example application of the use case where website visitors or audience members then use the data mining system to find content related to a page they are currently viewing. The focus of this research is to provide enough visualisation in order to evaluate the quality of content clusters generated, but it is an important design principle to consider use cases at all layers of the system.

3.4 Data Pipeline

A core subsystem in the overall system is a conceptual data pipeline whose input is a URI or IRI identifying a content item published on an organisation's website and the output is feature sets ready for applying machine learning.

In this section, a theoretical pipeline is inductively defined in steps such that an application of this pipeline would choose to implement some subset of all potential pipeline stages as appropriate for the relevant problem domain.

In Chapter 4, aa system is engineered that implements as many of these pipeline stages as possible such that a running instance of the application can configure which components to use and which not to use. Then in Chapter 6, an evalution of the system is given while it is running each component in isolation to demonstrate which of the theoretically-defined processes in this chapter appears to be most effective in generating feature sets specifically for clustering web content.

3.4.1 Definitions

This system requires some initial definition of some data structures in use:

IRI
: The input to the system is a character string conformant to the IRI syntax defined in RFC 3987[1]. This allows more generality offered by URIs[2] but is trivially made compatible with systems that use URIs through the conversion algorithm defined in section 3.2 of RFC 3987. Note that the public URL by which the public can read or otherwise consume the content is a valid identifier, but we are not restricted to that.

Feature Set
: The final output of this data pipeline is a data structure analogous to a relation or tuple per IRI fed into the system. Every IRI should have a literal value against all possible columns or fields. For binary fields, (e.g. the presence of absence of a concept tag), a more pragmatic structure might be a list of tags positively associated with the IRI rather than explicitly assigning $false$ to all tags to which the content item does not pertain. This is analogous to a spare matrix when dealing with a large number of dimensions.

Named RDF Graph
: The structure used throughout most of the data pipeline is that of an RDF graph. This is used for all the benefits outlined in Section 2.3 such as ease of transformation and combining of data sets. Named graphs are used such that all data acquired are keyed back to the IRI of the content item being processed. This also allows all graphs to be combined in a *triplestore* if needed to allow SPARQL queries across the combined data for all content items. This can be modelled as a data structure in many programming languages, but where a serialisation is used (e.g. examples shown here or to send the data between components), the JSON-LD[19] syntax will be used.

3.4.2 Identity Graph

```
{
  "@id": "http://example.com/entity/1",
  "@graph": []
}
```

Listing 3.1 : Identity graph for a content item in JSON-LD syntax

With the knowledge only of a content item's IRI, we are arguably only able to produce an empty named RDF graph. Such a graph for an example IRI `http://example.com/entity/` is illustrated in JSON-LD syntax in Listings 3.1.

The most naïve feature set we can generate from such an RDF graph is clearly a singleton relation `("http://example.com/entity/")` where a single IRI field has the value `"http://example.com/entity/"`. It is also clear that a set of one-dimension feature vectors with unique values in each is not suitable for

[1] http://tools.ietf.org/html/rfc3987
[2] http://tools.ietf.org/html/rfc3986

any form of machine learning activity. This does, however, illustrate a baseline for a working software application that is – at least in the syntactic sense – transforming IRI inputs to feature sets outputs. Such a *null* feature generator is depicted in Figure 3.3.

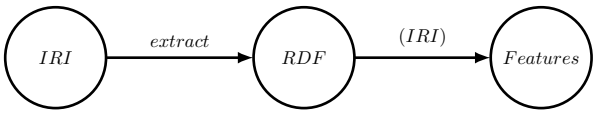

Figure 3.3: Null feature generator

Note that Figure 3.3 shows all three data structures involved despite having no functional use. We can also see top-level definitions of the process where we first *extract* semantic information in RDF from a content item indentified by IRI and then *generate* features therefrom. More useful models can now be inductively defined by adding atomic subcomponents that may each add value to the overall transformation.

There are three clear axes along which we can improve this pipeline: *extract* more RDF data knowing only an item's IRI, expand or *enrich* an existing RDF graph and then improve how we *generate* features for data mining. In the first instance, we can consider the former and add a single pipeline stage for expanding the RDF graph.

3.4.3 RDF Extraction

Tim Berners-Lee outlined four rules[1] for Linked Data, rule number three of which states "When someone looks up a URI, provide useful information, using the standards". If we assume that many pages have embedded some semantic web or RDF data, then a simple extraction strategy would be to deference the content item's IRI via an HTTP GET and pass the content to a parser capable of extracting RDFa, microformats, etc.

Many tools such as the RDFLib[3] provide functionality for taking a URL and returning an RDF graph of all data found when fetching the resource it represents, so this is arguably an ideal first choice in attempting to learn something about a content item from its IRI.

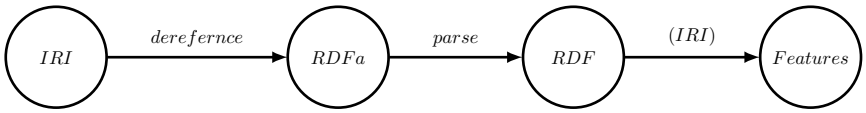

Figure 3.4: Semantic web data extraction

Figure 3.4 depicts the pipeline with a simple dereference step added. Note that the feature set generated is still the singleton relation with only the IRI value. Thus the next step should be to add a step that improves the feature set generation.

[3]https://github.com/RDFLib/rdflib

3.4.4 Feature Set Generation

Paulheim and Fürnkranz[12] described a number of SPARQL queries for generating feature sets from RDF data, which could inspire a simple query such as that shown in Listings 3.2. This query generates a boolean `true` value for any properties that match and implies `false` for those that do not.

```
SELECT ?p ?v
WHERE { ?iri ?p ?v . }
```

Listing 3.2 : Generates field `content_?p_?v` with value `true`

As also noted by Paulheim and Fürnkranz, this overlooks the *open world assumption*[16]. However, application of clustering algorithms on binary data can employ asymmetric distance metrics such as Jaccard similarity coeffcient[20], which notably avoids deriving similarity from negative values. That is, two content items lacking a particular property will contribute no information about their (dis)similarity. Thus we safely avoid inadvertently grouping together one item that genuinely lacks the property with another that indeed has the property, but we lack the positive assertion thereof in the data extracted.

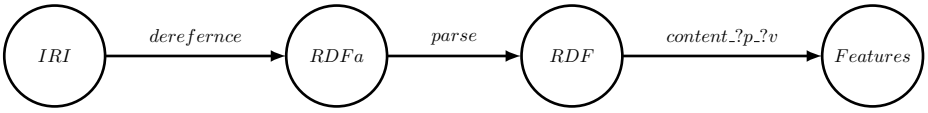

Figure 3.5: Semantic web content extraction with basic SPARQL feature generation

The basic pipeline in Figure 3.4 can thus be augmented with this basic feature extraction to produce the pipeline depicated in Figure 3.5.

3.4.5 RDF Enrichment

The third and final direction in which this data pipeline can be improved is in terms of data enrichment. A simple strategy here is to repeat the deferencing used in Section 3.4.3, but for each IRI found as the object of a triple in which the initial IRI is the subject. Formally:

$$g \in G \models \exists p, o.(IRI, p, o) \in g \rightarrow g' = deref(o)$$

That RDF graphs can be modelled as mathematical sets as in Section 2.2 means we can express a graph enriched this way as a *union* of the initial graph with each graph returned from all dereferencing:

$$g \in G \models g' = g \cup \bigcup \{deref(o) \mid (IRI, p, o) \in g\}$$

Figure **??** shows the data pipeline with this additional enrichment stage. Note that now we have potential for some stages being executed in parallel.

So far in this section, we have inductively built up a data pipeline from a "null" base working with only identity graphs to a simple pipeline capable of *extracting* an RDF graph, *enriching* it and then *generating* features from it.

What needs to be proven through experimentation is *which* of these provides the information required for effective data mining. As part of this experimentation, we can now look at further techniques and approaches to try.

3.4.6 Improving Extraction

In order to gain a larger set of RDF data at the start of the pipeline, we can derive further ways to get information about a content item given only its IRI as input.

Rizzo and Troncy[15] defined a framework called NERD capable of combining multiple entity extraction systems to provide a unified way of identifying – and disambiguating – named entities within a given body of text. With such a system, we can create a second, parallel RDF extraction strategy that creates a graph of triples in the form:

$$(IRI, rdf{:}about, Entity)$$

where *Entity* is an IRI representing a concept or entity believed to be found in the content's textual content. A data pipeline complementing the RDFa-based extraction is depicted in Figure 3.7. Note the ability to apply a simple set union to the result of each extraction as with the enrichment.

Another strategy can be to infer a relationship between two content items where one contents an HTML link to another. It is not always possible to derive precise semantics of such a link (unless the publisher has kindly provided a `rel` attribute), but a weak relationship such as:

$$(IRI_1, ex : related, IRI_2)$$

might prove – through experiementation – to be useful enough for data mining insights.

The fourth and final extraction strategy explored is acquiring metadata from bespoke Content Management Systems and other internal APIs. This is generally the only option used in enterprises settings, as discussed in Section 2.3. This is likely to be the richest source of information where an enterprise has typically preferred bespoke integrations against non-hypermedia interfaces, so experimentation should help quantity or qualify the value such a direct integration adds (perhaps to consider it in combination with the cost of bepsoke, repeated integration projects).

The assertion explored here is that such bespoke integrations can *complement* cheaper work such as extracting RDFa with pre-built tools (and thus be developed one-by-one after the initial release of an application such as this data pipeline). Note that repeated custom integration projects means that each data source requires a different application be developed (as opposed to the reuse of a single RDFa parser or HTML link scraper). This also means we are not necessarily comparing like-for-like if we introduce only one at a time. It also makes it difficult to evaluate data mining of a diverse content corpus if an integration against an API provides additional metadata for only, say, 10% of that corpus.

These challenges aside, it is clear that bespoke integrations have a clear place in this data pipeline being applied in a real enterprise setting. Now that we have a complete set of theoretical stages for *extraction*, the remaining improves lie now in the *enrichment* and *feature generation* stages.

3.4.7 Improving Enrichment

In addition to enriching through dereferencing linked entities, it is proposed to explore the following options:

- inferring relationships to hypernyms as defined by Wordnet[11];
- inferring facts based using rules derirved from expert domain knowledge;
- using RDFS and OWL to generate new triples with well-established Ontology rules.

In the first approach, we can consider a relationship rule such as:

$$(IRI, ex : related, ex : Dog)$$

and *infer* the fact:

$$(IRI, ex : related, ex : Animal)$$

and produce an enriched graph containing all additional facts inferred in this way.

When dealing with proper nouns and named entities, inferring facts based on domain knowledge may be more appropriate. For instance, the rule in n3 syntax:

```
{
  ?article ex:takesPlaceIn ?city .
  ?city a ex:City .
  ?city ex:capitalOf ?country .
} -> { ?iri ex:takesPlaceIn ?country }
```

might be useful ot help cluster articles that take place in the same country – even if the countries are not always explicitly mentioned therein. Such an inference requires knowledge about cities and countries to write and domain experts for different types of content might be able to offer more nuanced rules.

An example for BBC content might be to infer that all articles written under the *Newsround* brand is suitable for children or that programmes that have broadcast times during the day are also suitable for children.

The third and final proposed improvement makes use of standard tools to find *closures* using, e.g. RDFS, ontology rules. With this approach, we can infer that entities that have a given type or class also have their superclasses and supertypes. This gives us similar inference to hypernyms, but with knowledge present in well-established ontologies.

An obvious example might where two content items have been identified as related to the same concept – so they would be candidates for clustering together – but when RDF data are extracted, it is found that two different URIs have been used for each:

$$(IRI_1, <http://dbpedia.org/property/related>, ex1 : entity)(IRI_2, <http://dbpedia.org/property/related>,$$

In the RDF graph for IRI_2, say, we might find the source had provided an `owl:sameAs` assertion such as:

$$(ex2 : anotherEntity, owl : sameAs, ex1 : entity)$$

This is possible in the case where the second item's data source uses its own set of identifiers for entities, but has chosen to provide equivalences to a more standard set of identifiers (e.g. DBpedia). With this information, our data pipeline can infer:

$$(IRI_1, <http://dbpedia.org/property/related>, ex1 : entity)(IRI_2, <http://dbpedia.org/property/related>,$$

and the feature generation stage might provide the common features `dbprop_related_ex1_entity=true` for both content items.

3.4.8 Improving Feature Generation

The feature generation outlined so far relies solely on boolean values indicating whether or not a given content item is related by some property to some entity. This recreates the concept of *tagging* where a given object is either associated or not associated with a series of *tags*.

One of the advantages of the RDF graph model is that we are not constrained necessarily to properties and attributes directly applicable to the entity. We could imagine adding a level of indirection to the query in Listings **??** to create the query in Listings **??**.

```
SELECT ?p1 ?p2 ?v
WHERE {
  ?iri ?p1 ?o .
  ?o ?p2 ?v .
}
```

Listing 3.3 : Generates field `content_?p1_?p2_?v` with value `true`

With the this query, features of the form `content_?p1_?p2_?v` can be generated. An example of this might be where a television programme content item has information about the actors that appeared therein, e.g. `ex:hasActor`, and furthermore we have information about those actors such as where they were born, e.g. `ex:bornIn`. With the path created by following both of these predicates, it is possible to create features for a television programme such as `content_ex:hasActor_ex:bornIn_Edinburgh` and we can potentially find similarity between programmes where the actors were born in the same city.

Perhaps a third step in the predicate path followed can give us even more useful features. The example above could be expanded to `content_ex:hasActor_ex:bornIn_ex:cityIn_Scotlan` to allow the more general ability to cluster programmes with Scottish actors, for instance.

Appropriate experimentation should show whether more value is gained by adding these additional levels of indirection.

3.4.9 Maximal Data Pipeline

In this section, a data pipeline was inductively built up from a base, identify pipeline with suggestions for potential improvements in different stages. An

application of all the ideas discussed so far might look like that depicated in Figure 3.8.

The goal of this work is to evaluate how the quality of unsupervised clustering changes with respect to enabling different combinations of the maximal data pipeline. In the next section, some of the technical architecture of this pipeline is outlined and the next chapter will cover how the overall system is then implemented.

3.5 Pipeline Architecture

TODO

3.6 Architecting for Experimentation

TODO

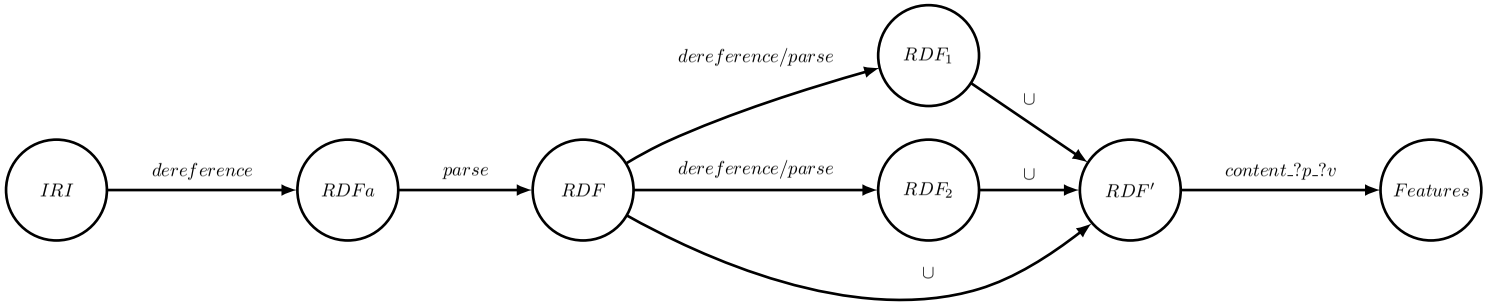

Figure 3.6: Semantic web content miner with addtional dereferencing of linked entities

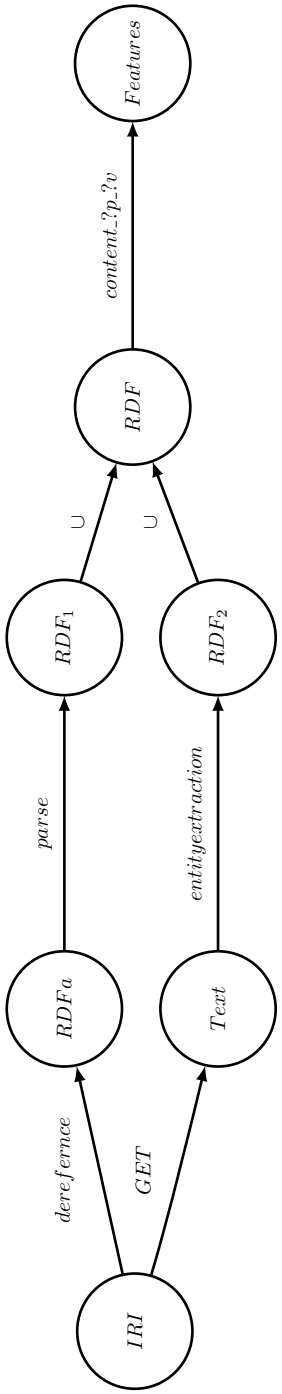

Figure 3.7: Named entity extraction in addition to semantic web extraction

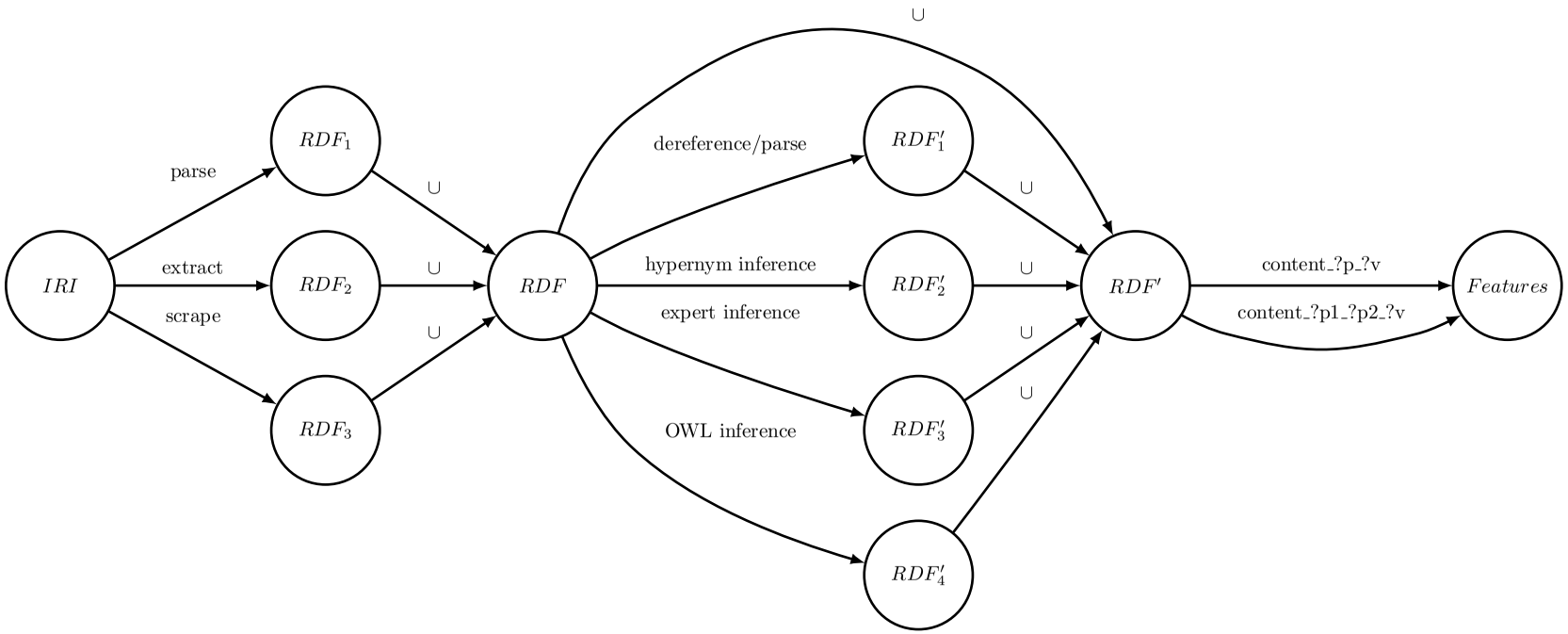

Figure 3.8: Maximal Data Pipeline

CHAPTER

FOUR

IMPLEMENTATION

In this chapter, some of the details of the system implementation are described. Initially, we discuss some of the implementation strategy employed due the experimental nature of the system and the desire to run multiple, competing approaches side-by-side in an efficient way.

The remaining sections walk through the implementation of each part of the data pipeline from upstream to feature sets and the chapter concludes with implementation of the clustering and generation of clusters for evaluation.

4.1 Software Architecture

All functionality was implemented as part of a single Python module named *distillery* that is run via the Unix command line, with different subcommands for each feature. Each command was designed around the Unix Philosophy[14] of doing one job per command and that they were composable via Unix pipes. This allows pipelines to be composed:

```
distillery extract <iris.txt | distillery enrich | distillery generate
```

that resemble the entire theoretical data pipeline described in section 3.4 without the need for middleware such as message queues. More typical enterprise architectures around message oriented middleware or event-driven architecture might need to be employed to scale this system to millions of documents, but the simple approach above is sufficient for tens of thousands of documents.

Even this simple approach would scale very well with suitably low network latency. The largest bottleneck observed was the necessity to do at least one HTTP request per item ingested and then enrich via an HTTP request per object of a triple where the IRI is a subject. For some content items, this could easily be over 100 HTTP requests at which point parallel CPU cores or asynchronous I/O programming does not overcome the demands on network performance.

Throughput is a particular challenge when data processing systems are first launched. The day-to-day volume of item creations and updates may be low enough for even a low-scale application, but the performance demands increase significantly when we wish to bootstrap a backlog of all content items historically

published – of which could easily be tens of millions for organisations such as the BBC.

For this, there is an appeal to designing such a system on a cloud computing platform such that it can be scaled up for the initial import of existing data and then back down for ongoing updates when that import completes. Such design is out of scope for this paper, but some discussion of limitations of the existing system in section **??** highlights where this system may be architected differently to be more suitable for such a cloud-based deployment.

The higher-level compositional architecture of the individual data processing stages can, of course, be altered independently of the design of those stages. The following sections will describe in greater detail those composable modules.

4.2 Avoiding Redundancy Across Multiple Experiments

The design of the data pipeline in chapter 3 would suggest an implementation with each stage implemented as a function converting a graph into a new, richer graph.

That is, whilst the different strategies illustrated in the maximal data pipline in Figure 3.8 can be run in parallel, the extraction stage itself simply converts the identity graph described in section 3.4.2 to a single graph (after having taken the union of all the strategies).

In the experiment that is the subject of this paper, the intent was to compare each of the extraction strategies individually, but also all possible (union) combinations thereof. This presents at least two potential implementation strategies:

1. Build the whole data pipeline and machine learning system such that it is configurable which strategies are invoked and run an instance per configurable combination; or

2. implement the software to produce all possible outcomes along the pipeline.

The former approach is stronger if the aim is to build the system closer to how it would be implemented in an enterprise: only one set of clusters is needed and even a production, enterprise system would be configurable if maintainers wished to enable or disable features as desired.

The latter strategy is arguably better for experimentation as every IRI fed into the source end of the pipeline is processed at the same time for every extraction and enrichment approach. This gives better assurance that the same input is used to evaluate each competing system.

Other operational reasons include being able to run the system on a single machine if necessary and that there is no repeated work: extraction by one means can be used in isolation, but also feed into a union with another technique without having to recalculate that graph. That is, if we have graphs A and B generated once each, then we can output A, B and $A \cup B$ for comparison. A parallel copy of each system would be calculating A and B twice each.

The limitation here is a production-ready implementation of the system would have be built again from the ground-up, perhaps reusing functions and library code from the experimental version, as the application itself will be

structured around this idea of multiple outputs for each input. This is likely to be the case for any experimental system and is arguably in line with Brooks' classic assertion to "Plan to Throw One Away"[3].

```
def all_extracted_graphs(iri):

    extraction_strategies = [
        dereference, extract_entities, find_links
    ]

    # This creates all 3 RDF Graphs only once
    graphs = [s(iri) for s in extraction_strategies]

    # Empty set removed from powerset
    for idx, gs in enumerate(powerset(graphs) - {{},}):
        # union defined elsewhere: list(graph) -> graph
        graph = reduce(union, gs)
        # Loop counter idx tells us from which technique
        # combination each graph comes
        yield idx, graph
```

Listing 4.1 : Python function that generates all possible RDF Graphs

An illustrative Python function is shown in Listings 4.1 where each of the three extraction techniques is invoked only once and a powerset function is used to generate all possible unions of those graphs and thus yield all possible usages or non-usages of each strategy to generate RDF graphs. The details of the three extraction functions are covered in section **??**.

4.3 Obtaining IRIs

It is a substantial undertaking to design, build and test a production-quality enterprise system that processing all content produced by a media organisation. An experimental system needs to focus on quick results and therefore is optimised to work only over a small sample of that content.

As introduced in section **??**, all that content is also likely to be distributed across multiple databases, content management systems and other stores. Whilst it is hypothesised in this research that semantics and linked data provide a good way to extract data about the total breadth of all content without any bespoke integration against internal systems, there still remains the enterprise integration problem of *discovery* of those content items.

Essentially, the data pipeline outlined in section 3.4 notably requires that known IRIs of content items are fed into it, but makes no statement about how those IRIs are acquired. This is a deliberate design decision to decouple the concepts of discovery from this extraction/enrichment workflow. Thus item may easily be re-ingested several times if it is updated (or simply periodically) and a modular approach like this defends against upstream systems changing and having to rebuild the triggers that notify when content is created or updated.

What follows it that the enterprise integration task has been reduced (we do not need to write bespoke code to extract basic data about every content item)

but not wholly eliminated (some custom adaptors need to be created to notify the pipeline of creation or update events in each respective data store).

However creation of any such notification adaptors is out of scope for this research altogether, so a heuristic approach was employed where ten different, themed content aggregation pages on the BBC website[1] were polled for any new content promoted by editorial teams. This provided a way to find new items shortly after they are published and also ensured the data sample ultimately used focused on content that was deemed noteworthy or interesting in the last few months.

This is not an effective method for obtaining a large number of items quickly as it is entirely dependent on the rate these aggregate pages are updated. Running slowly over a long period yielded approximately ten thousand individual IRIs with little effort nor need for large amounts of integration work.

4.4 Extracting RDF Graphs

Listings 4.1 hinted at three individual functions capable of obtaining an RDF graph for a given IRI, all based on the extraction strategies outlined in section 3.4.3.

4.4.1 Dereferencing

Listings 4.2 shows how trivial a dereference function can be if we use the comprehensive RDFLib library for Python. The library can handle content type negotiation[6] to ensure it can happily dereference and parse any response that contains semantics, particularly RDFa (including embedded Turtle) and microdata within an HTML page.

```
from rdflib import Graph

def dereference(iri):
    g = Graph(identifier=iri)
    g.parse(iri)
    return g
```

Listing 4.2 : Python function that generates all possible RDF Graphs

4.4.2 Entity Extraction

DBPedia Spotlight[4] was chosen to perform the entity extraction. A hosted version is available as a free service and it has been shown to be effective in many cases. A true comparative study of the merits of alternative entity extractors is out of scope for this paper.

```
import requests
from rdflib import Graph, URIRef, Namespace
```

[1]Including, among others, the BBC Homepage (http://www.bbc.co.uk), BBC Arts (http://www.bbc.co.uk/arts) and BBC Science (http://www.bbc.co.uk/science)

```
FOAF = Namespace('http://xmlns.com/foaf/0.1/')

def extract_entities(iri):
    g = Graph(identifier=iri)

    text = requests.get(iri).text

    spotlight_response = requests.post(
        'http://spotlight.sztaki.hu:2222/rest/annotate',
        data={
          'text': text, 'confidence': 0.8, 'support': 20
        },
        headers={'Accept': 'application/json'},
        timeout=600)

    if spotlight_response.ok:
        r_json = spotlight_response.json()
        if 'Resources' in r_json:
            annotations = {
                resource['@URI']
                for resource in r_json['Resources']
                }
            if annotations:
                for entity in annotations:
                    g.add((
                        URIRef(iri),
                        FOAF.topic,
                        URIRef(entity)))
    return g
```

Listing 4.3 : Python function that uses DBPedia Spotlight to extract entities from web pages

Listings 4.3 shows a basic implementation of a function that fetches a page and then feeds the content into a request to DBPedia Spotlight. All entities thus found are then fed in as objects to a series of `foaf:topic` triples.

There is much that can be tweaked in this implementation and the final version used in the experiment relied on a ad hoc whitelist of XPATH queries known to narrow down the content to the main article content on BBC pages. This is because the implementation shown in listings 4.3 does nothing to prevent entities being extracted from page chrome such as navigation links and other cross-site concerns.

In a given organisation, such an approach might well be sufficient to extract main article content from a controlled set of pages. In a production system, it may be more appropriate to consider readability systems such as that from Arc90.

4.4.3 Hyperlink Relationships

In listings 4.3, we see a simple Python function that uses the Beautiful Soup library[2] to search across all links in a document and infer a "related" property on the assumption that pages like to other pages that are in some way related.

```
import requests
from rdflib import Graph, URIRef, Namespace

DBPROP = Namespace('http://dbpedia.org/property/')

def find_links(iri):
    g = Graph(identifier=iri)

    text = requests.get(iri).text
    doc = BeautifulSoup(text)

    # For all <a> tags that have an href attribute...
    for a_tag in doc.find_all('a', attrs={
        'href': re.compile('.+')
    }):
        # ... handle href values being relative links ...
        other_iri = urljoin(iri, a_tag.attrs['href']).strip()

        # ... infer triple if IRI is to another BBC page
        if bbc_url.match(url):
            g.add((
                URIRef(iri),
                URIRef(DBPROP.related),
                URIRef(other_iri),
            ))

    return g
```

Listing 4.4 : Python function generates triples based on links to other pages

This falls foul of similar issues to entity extraction as described in section 4.4.2 in that site-wide navigation links and other supporting page elements may link to very general pages (e.g. a persistent link to "home" on every page).

With appropriate feature selection, this may have no impact, but for this experiment a simple heuristic was employed to narrow down to the "interesting" part of BBC pages.

Another feature in listings 4.4 is a restriction to consider only links to other BBC pages. There may be a positive or negative effect in addtionally noting external pages linked to from content pages, but that is left to future research to consider.

[2]https://www.crummy.com/software/BeautifulSoup/

4.5 Enriching and not Enriching

The combinatorial consequence of the usages and non-usages of the three extraction techniques described so far is that the experimental system developed is already comparing seven (2^n less the case where no extraction is employed).

In order to maintain focus in the experiment, only one of the enrichment techniques described in chapter **??** was implemented and its usage and non-usage merely doubled the number of systems for comparison to fourteen.

This proved sufficient to give indicative results as to whether there is value added through any enrichment, but future research could certainly compare the merits of different enrichment approaches.

```
import rfc3987

def get_objects(graph):
    return {str(row.o) for row in graph.query('''
            SELECT DISTINCT ?o
            WHERE {
                ?iri ?p ?o .
            }
            ''', initBindings={'iri': graph.identifier})}

def enrich(graph):
    new_graph = copy(graph)

    for o in get_objects(graph):
        if rfc3987.match(o, rule='IRI'):
            new_graph.parse(iri2uri(o))

    return new_graph
```

Listing 4.5 : Python function that enriches a graph via deferencing objects

Listings 4.5 illustrates a simple function to deference all objects of triples where the current IRI is a subject. Note the convention to set the graph's own identifier to that of the content item of interest.

Using SPARQL in this way opens up possibilities to perform multiple queries to choose IRIs to dereference. For example, we may wish to include semantics for objects reachable by following two predicates on the RDF graph and consider much more indirect data. Evaluating the utility of this is left for future work.

4.6 Feature Set Generation

The feature generation strategy employed was dervied from a distilled portion of the approach introduced by Paulheim and Fürnkranz[12].

Listings 4.6 shows an illustrative, recursive function that "walks" the RDF graph starting with the content item's IRI as the initial subject.

```
from rdflib Literal, URIRef

def generate(g, subject=None, depth=3, features=None, prefix=''):
```

```
    if depth <= 0:
        return
    if not features:
        features = {}
    if not subject:
        subject = g.identifier
    for p, o in g.predicate_objects(subject=subject):
        if type(o) == URIRef:
            new_prefix = prefix + clean(p) + '_'
            features[new_prefix + clean(o)] = True
            generate(g, depth=(depth - 1), features=features, prefix=new_prefix
        elif type(o) == Literal:
            features[clean(p)] = str(o)

    return features
```

Listing 4.6 : Python function that generates feature sets from RDF graphs

The `clean` function simply provides some cosmetic cleaning up of predicate IRIs to prefer CURIEs as they are more compact. A typical example feature might then be `foaf:topic_dbpedia:Albert_Einstein: true` where the leaf object is an entity or `ogp:title: "Gandhi: Reckless teenager to father of India"` if the leaf object is a literal.

4.7 Feature Selection

A simple selection heuristic was employed, again inspired by the work of Paulheim and Fürnkranz[12], to go some way to reduce potential noise in the subsequent machine learning phase.

In this experiment, all features that have the same value or entirely unique values were removed and values that occurred twice or more for a given feature had to make up over 5% of possible values for that feature. That is, if a feature had over 95% of its values being unique or all the same, then it was rejected.

4.8 Clustering Implementation

Hierarchical, agglomerative clustering was implemented by implementation of a `cluster` function that was merge the two "nearest" clusters on each invocation. This assumes that all content items not yet in a cluster are implicitly in a singleton cluster and the ability to merge one at a time allowed for control of when to cease merging.

The closeness of two potential, candidate clusters was calculated via a max linkage on the basis that it ensures that any two content items within a cluster to be considered related. This avoids the so-called *chaining phenomenum* where single linkage may create clusters where two unrelated items are members because they are each related to an item of chain of related items therebetween.[5]

The distance between content items was defined by the Jaccard distance due to its suitability for binary and categorical data.

4.9 Results Generation Strategy

With all of the components described so far in place, an experiment was run processing nearly 10,000 content items from the BBC website with each going into one of fourteen combination of extraction and enrichment approaches.

For each approach, hiearchical clustering was applied repeatedly until the next merge would produce a cluster with cohesion below 0.5. This is reasonably arbitrary as each technique will have different concepts of cohesion, but it provided a strong basis on which to evaluate each approach on whether its cohesion meaningfully predicts whether a human would agree that cluster contains related items.

CHAPTER

FIVE

RESULTS AND ANALYSIS

In chapter 3, a data pipeline is derived that – at the logical maximum (see figure 3.8) could generate feature sets using any of 112 combinations of extraction and enrichment techniques. This is based on the product of using or not using any of three described extraction approaches (less the case where no extraction is performed) and enabled or disabling any of four distinct enrichment approaches (retaining the case where no enrichment is performed).

More practically, in chapter 4, the implementation of a real system for evaluating a subset of only fourteen combinations of approaches is described. Therefore there are fourteen resultant sets of clusters of the same BBC web content produced for comparative analysis.

In the following sections of this chapter, an objective and technical analysis of the clusters is performed, leading to a critical review of each individual extraction and enrichment approach, based on how each respective technique appears to contribute to the results.

In chapter 6, there is an evaluation of the overall system based on qualitative data gathered by a survey of real human users. That evaluation is more focused on the suitability of the holistic use of semantics and machine learning in the real world application of suggesting related media content to web users.

Conversely, in this chapter, we take a much narrower focus on the shape, size and quality of clusters themselves.

5.1 Comparison of Clusters Produced

In this section, we take a walkthrough of some of the clusters produced based on what seems interesting. Even reviewing a handful of the results shows some insights into how the different semantic extraction approaches have behaved and we can see some clear evidence where the process is generally working and where it is making mistakes.

It is not feasible to analysis every result here in this way, but chapter 6 describes how every approach was evaluating against every other approach The sections following this one in this chapter try to give a summary of advantages and disadvantages we can see in each technique ahead of the qualitative evaluation.

The most stark difference between the results produced is the large range in terms of number of clusters produced for each approach respectively.

Table 5.1: Number of clusters produced by each combination of approaches

	Not enriched	Enriched
Deference	5	249
Entity Extraction	35	32
Hyperlink Relationships	2	56
Dereference and Entity Extraction	329	340
Entity Extraction and Hyperlink Relationships	199	233
Dereference and Hyperlink Relationships	53	117
All three extractions	184	223

Table 5.1 shows the number of clusters produced by each combination of techniques. The range here is explained by some variants of the system produced few, very large clusters and some a large number of smaller clusters.

5.1.1 Few, Large Clusters

Which of the two extreme outcomes is more suitable is arguably a matter for real evaluation, but it is perhaps reasonable to claim that clusters which are too large are less useful for choosing related content items to suggest to users. In the case of the unenriched graphs obtained via hyperlink relationships only, one of the two clusters produced contains 2,323 items – a number that risks "related content" suggestions being too erratic and random.

A deeper inspection of that largest cluster reveals every item was deemed to be related to an FAQ page on how to share BBC pages on social media websites – a page that appears to be linked to from every BBC programme information page or iPlayer catch-up video page. This is clearly not a good indication of the content of the page and raises two immediate issues:

1. the system would benefit from further heuristics that avoid inferring too many relationships from ancillary pages such as FAQs that are unrelated to the content itself; and

2. the nature of agglomerative, hierarchical clustering means that this system variant spent a large amount of time grouping all pages that link to this FAQ page at the expense of other, potentially useful clusters. This is a consequence of hierarchical clustering being a greedy algorithm.

Further discussion of the merits and problems with hyperlink relationships is covered in section 5.2.3.

5.1.2 More, Small Clusters

If we look at the other end of the spectrum, the system variant that produced the most clusters was the enriched union of dereference and entity extraction. With so many clusters, the distribution of sizes is better shown by the histogram in figure 5.1. Here we can see the results are dominated by clusters with 25 or fewer items.

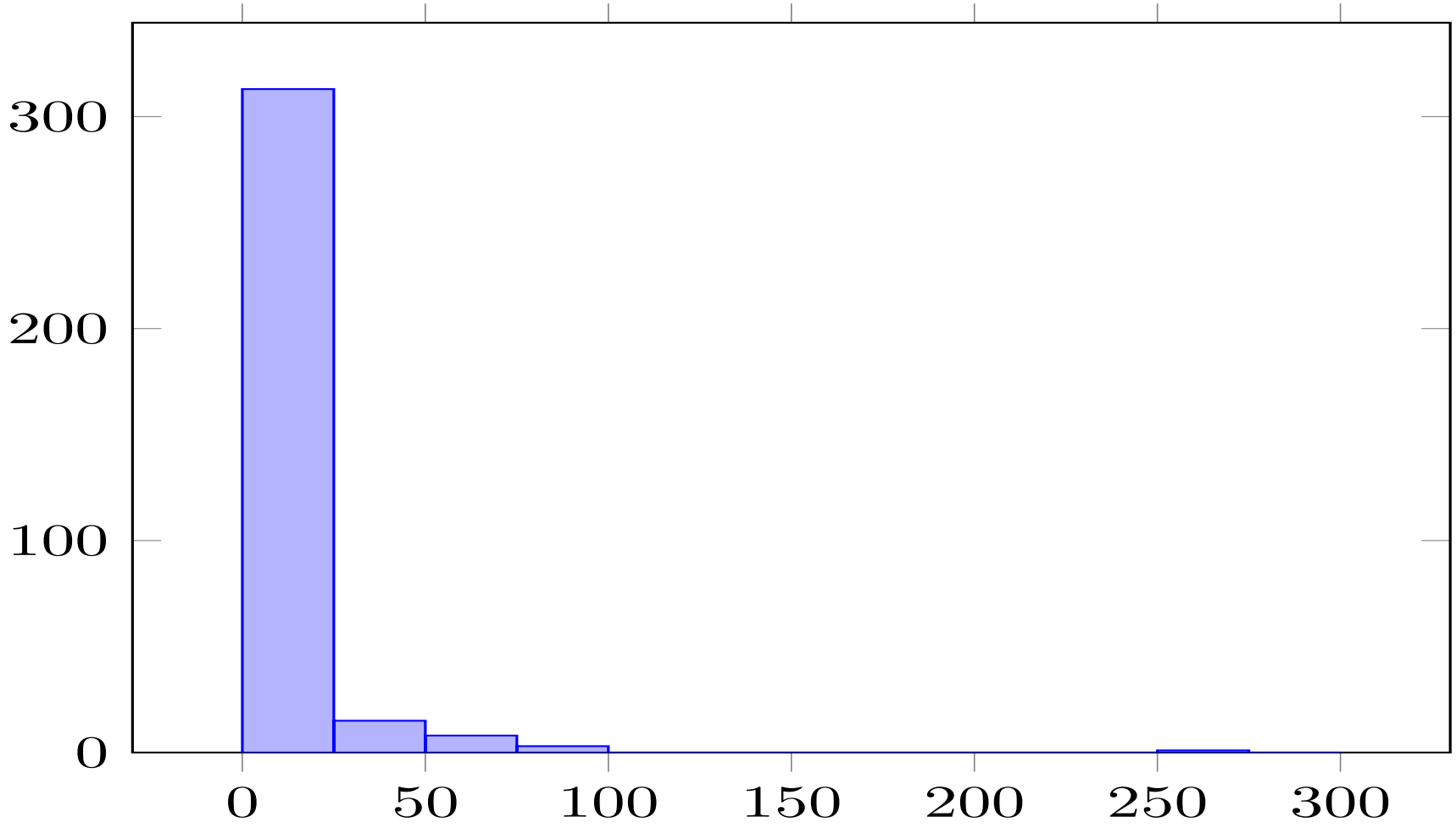

Figure 5.1: Distribution of cluster sizes produced by the enriched union of dereference and entity extraction

An exploratory insight into the makeup of these clusters is shown in table 5.2 in which the ten most common features are shown for the one of the smallest, one of the biggest and one of the mean-sized clusters.

Smallest Cluster

The smallest cluster with 2 items in table 5.2 seems to contain two pages that mention BBC iPlayer (the `foaf:topic` predicate suggests this was found via entity extraction) and whose primary content is an item with `rdf:type` "RadioEpisode" in the Schema.org[1] vocabulary. This shows hints of a strong complement between semantic web data and entity extraction confirming that textual content on the page correlates with any semantics also embedded in the page.

Interestingly, this cluster is also strengthened by the fact that both pages have a common thumbnail (the `ogp:image` predicate indicates a thumbnail when used to share on Facebook). An examination of the two content pages in question confirms that they are two episodes of the same "5 Live Breakfast" programme on the *BBC Radio 5 live* radio channel.

The other two predicates for the small cluster are less clear without some explanation as to their origin. The `meta:twitter:card` predicate is a documented[2] HTML meta tag that to indicate a page has included further meta tags to control which metadata to include in the so-called *Summary Cards* Twitter generates as a summary of page a user is sharing in a post to the website. This overlaps some of the features provides by the *Open Graph protocol*[3] vocabulary used by Facebook for similar means.

This overlap between Facebook and Twitter highlights both:

- a failure for industry to adopt (or for academia to convince them to adopt) semantic web initiatives such as RDFa, Microdata, Microformats or Schema.org – designed to provide machine-readable semantics in ways richer than simple meta tags; but also
- a critical opportunity for research and development in semantics to build on top of the real, commercial pressure created by Facebook, Twitter, Google et al to enrich pages with semantic data so that content providers can control how their content is summarised when shared on their platforms and also how their content is analysed for searching and extrapolation of real time trends in social media conversation.

That predicates for Facebook and Twitter are appearing in this experiment so easily also shows evidence that the BBC is among the media organisations that now have content producers providing semantic enrichment alongside their content production or at least that automated generation of in-page semantics from metadata already present in content management systems is being prioritised alongside other work to develop features more visible to human users. It is not unreasonable to suppose this drive is at least partly driven by content producers' and content promoters' desire to control how content is displayed and discovered on third party social media platforms.

[1] http://schema.org/
[2] https://dev.twitter.com/cards/types/summary
[3] http://ogp.me/

$N = 2$	$N = 9$	$N = 265$
ogp:image=”[. . .]”	rdf:type_schema:RadioEpisode=true	md:item_rdf:nil=true
foaf:topic_dbpedia:BBC_iPlayer=true	schema:url=”[. . .]”	foaf:topic_dbpedia:Digg
rdf:type_schema:RadioEpisode=true	foaf:topic_dbpedia:JavaScript	twitter:card:”summary_large_image”
rdfa:usesVocabulary_schema=true	meta:twitter:card=”summary”	foaf:topic_dbpedia:LinkedIn
meta:twitter:card=”summary”	foaf:topic_dbpedia:Listen_(Beyoncé_Knowles_song)”	meta:apple-mobile-web-app-title=”BBC Sport”

Table 5.2: Ten most common features across the smallest, the largest and a mean-sized cluster produced by the enriched union of dereference and entity extraction

Mean-sized Cluster

Returning to the mean-sized cluster in table 5.2, we see a cluster where the most common items are identified as pages each for a single episode. There is no indication in the top five features that they are episodes of the *same* programme, however. This could well be a strength of this system variant in that it is able to suggest other radio programmes on the basis it has understood they all share a common type, but it is open to suggesting episodes of other programmes for greater diversity. In fact, this may be a necessary feature in any "related content" feature as it could be safely assumed that any BBC page for a single episode of a radio programme already contains clear and sufficient navigation links to find more episodes of the same programme; machine learning adds no value to the user here.

Again, there is potential to call out a real benefit in the combination of in-page semantics for this "hard" categorical data (i.e. the page's content is of type "Radio Episode") with entity extraction for a looser, schemaless vocabulary of words that more qualitatively describe the topics covered in the content.

In this case, the topics extracted appear to be the JavaScript programming language and the song *Listen* by music artist Beyoncé Knowles. A manual review of the pages involved finds them to be radio episode pages again, but with no clear indication that they mention either topic. Deeper analysis of the HTML source for these pages finds two "mistakes" the system has made:

- the word JavaScript appears in the page only as part of the MIME type `text/javascript` in markup around JavaScript *code* supporting the page; and
- the word Listen appears on *every* BBC page concerning a radio programme due to the fact they all contain a "Listen" button allowing the user to listen to the live radio stream for that particular radio station. DBpedia Spotlight has assumed this word is a mention of a song titled *Listen*.

Largest Cluster

Finally, the largest cluster has unique quirks of its own. The most popular feature across all members was a rather strange `md:item_rdf:nil=true` whose original is unclear with some explanation. This triple appears to have originated in the RDFLib Python library as an artefact of the Microdata parser (run on any HTML response) having found no Microdata items in the page. This is likely to be a bug in the library where it may have been more useful not to say anything.

The result of this bug or unexpected behaviour is that variants of the data pipeline that use deferencing as an extraction method will have this propery for any page that does not use Microdata simply because of the choice of RDF parsing library. There appears to be enough mix of BBC content pages that do and do not use Microdata such that this feature passed the feature selection filter that should have removed any feature with the same value across the whole corpus.

This is part of a bigger behavioural pattern where embedded semantics appear to tell us a lot about how pages are built as well as describing the content

contained therein. This is not necessarily a drawback and is discussed further in section 5.2.1.

Two other features in the five most common across the cluster suggest many of the content items mention the social network, LinkedIn, and news aggregator, Digg. These come from a similar issue with the *JavaScript* and *Listen* entities for the mean-sized cluster in that they are mentioned in the page as part of controls inviting users to "share" the content on social media sites. These are also clear false positives that reinforce the importance of ensuring entity extraction is performed in relevant textual information. This is discussed in section 5.2.2.

The fifth most common feature for this cluster is genuinely something informative about the content in the page – it appears to be a meta tag informing of a title to use for the page on iOS devices. This is again likely to be an artefact of how the BBC Sport section of the BBC website is developed with requirements relating to iOS devices, but it also – perhaps inadvertantly – tells us something semantic about the page in that it is part of BBC Sport.

This may not be groundbreaking insight as there are clearly more definitive ways to distinguish BBC Sport content (e.g. all the URLs paths start with the word "sport"), but it starts to showcase a potential advantage of unsupervised learning on semantics that it can use subtle clues based on how the pages are constructed to determine something about them – if this is deemed useful, that is.

It is also a showcase of how an unsupervised data mining process can be said to have "learned" that these are all BBC Sport articles without having to be instructed through bespoke business rules. This hints at some need for further research as to whether enterprises can truly adopt semantics and data mining to derice business rules that survive organisational and systems changes due to being less brittle than bespoke enterprise integration software based on static business rules (i.e. rules that are true today but need updating when the business context changes or risk breaking).

The more qualitative evaluation in chapter 6 will provide some idea of which properties ultimately end up actually being useful.

5.1.3 Impact of Noise

The previous section highlighted many cases of "false positives" and errors made by this experimental system. The intent of the designed system is to provide an initial baseline capable of processing any piece of media content, perhaps at the expense of the depth that would be achieved by explicit data adapters developed against an internal data API. Thus it is useful to analyse where the system performs less well because a goal for any initial, baseline application should be to evolve and grow it to adapt to errors as they find them.

An open question still for further research is whether these errors are real barriers to gaining useful insight via data mining. The open world assumption and loose nature of the RDF model lends itself well to embracing the patchy and noisy nature of the open web of data and that can be a strength even across a single organisation's own content – particularly in large and fragmented organisations.

In a rigid, closed-world approach, we may see bespoke data adapters fail altogether the first time a team responsible for a data API changes, say, the

XML structure of its query responses without notifying all clients. We may also see data mining processes lacking the ability to find trends in the entire corpus since the data ingest written specifically for custom APIs has only been built for a subset of all the data stores in the organisation.

The nature of machine learning algorithms is they are intended to learn for themselves which data properties are useful and which are not. In a classification model, we would expect the learning to place more weight on features that produce the answers desirable by those that use the model and some of the noise naturally falls away.

In the current experiment around unsupervised clustering, not only was feature selection implemented to throw away features that do not vary and features that vary too much, but clustering algorithms should favour where items where they share a number of properties, not group content together simply because it shares one erroneous attribute.

Thus it can be concluded that the mere existence of noise is not enough to fault a particular configuration of our semantic data mining system, but there is a need to answer the question as to whether the noise is enough to overwhelm and confuse any machine learning attempts. The question we look to answer in this research is which of the variants of this experiemental system are more or less susceptible to noise and errors and recommend how we might research overcoming any false positives that do arise.

At the very least, there are variants of the system that are clearly failing very quickly, e.g. the hyperlink relationships producing only two, unusably large clusters shows a clear need to overcome the causes of those problems before even considering using it in a real system.

5.2 Strengths and Weaknesses of Different Approaches

The analysis in the previous section walks through some exploratory reviews of a small number of clusters, whereas in this section an overview of how each approach appears to show strength and failures is given.

5.2.1 Embedded Semantics Extraction

Some of the analysis in section 5.1 touched on some of the side effects of semantics being tied to *how* a page was built rather than *what* the page is about. It was also shown semantics are being included due to pressures from social media organisations for content providers to describe summaries of their content for better display and discovery. Based on this and other observations in the clusters produced, clustering based on deferencing URLs and reading their embedded semantics has the following advantages:

- Many media companies are providing semantic data for the purposes of display and other features in a social media context. This means there is a base level of metadata to be extracted without asking those building the pages to add in more semantic markup.
- Some parts of the BBC website have chosen to include much richer semantics. This is observed in section 5.1 where pages about radio episodes

where described as such. Some parts of the BBC Sport website also mark up entities such as sport teams as well.

We can also observe the following challenges or drawbacks:

- There can be a lot of noise from meta information about the semantics, e.g. the RDFLib library declaring that a page contains *no* Microdata as observed in section 5.1 or RDFa declarations that inform that the page is using a particular vocabulary by default [4]
- Some parts of the BBC website provide only basic metadata (e.g. for social media as above) and some have chosen to include far richer semantics. This leads to items varying substantially in terms of the size of the RDF graphs we can extract. The unbalance show signs of causing a bias where clusters may not be likely to a mixture of such items, instead preferring to group the semantically-rich items together for example.
- There are very general-purpose assertions such as `ogp:site_name="BBC"` provided simply to tell Facebook that a given page is part of the overall BBC website. This is not useful if the whole corpus is from the BBC website, but does not get filtered out due to pages being built differently not all development teams behind individual parts of the website choosing to include that property. Thus the varied usage and non-usage of this feature implies to a machine learning algorithm there is information in the feature's presence or non-presence.

The last drawback above suggests a bigger question here as to whether there is a more general challenge with machine learning and the open world assumption of linked data and semantics. A classification model surely has to choose between non-membership or membership of a category with some degree of certainty, so will mistake the lack of knowledge of an attribute that would place it in a category as if that item genuinely lacks that defining feature.

Similarly, a clustering algorithm can only group items on what is known. Perhaps if features are lossy in a random sense, then other features will still link two related items, but if, say, a whole section of a website fails to provide semantics on genre classification of its content, then content items in that section will never be paired up with pages in another section of the site that does provide useful information above genres.

It should be noted that this is still a problem in trying to integrate information between two bespoke databases that follow a closed-world assumption. If one content management system encourages authors to categories their work into genres and another does not, then we still cannot link up content. This situation is arguably worse in that two systems that *do* provide genre information, but using wholly different names for the properties and a different taxonomy of values, still cannot easily unify the data models without explicit and possibly extensive rules translating between the two.

It could be said that the Linked Data concept and by extension the idea of Linked Enterprise Data allows two different data sets to be developed independently for their own use cases so long as they (or a third party) is encourage to provide additional semantics that link the vocabularies. This is even easier still

[4]The feature `rdfa:usesVocabulary_http://schema.org/=true` is very common

if the two systems use different, but well-known vocabularies, as the mappings may already exist.

5.2.2 Entity Extraction

Inferring semantics from entity extraction has some clear advantages:

- Extracting real things from textual content gives a far stronger feel for the topics covered therein. Note this is essentially the basis of any keyword-driven search engine such as Google.
- Implementation in the enterprise is relatively simple given the vast number of technologies already available and the long history of research into entity extraction. Reuse of an existing solution – as implemented in this research – is sufficient for many use cases.

However, there are still many drawbacks and challenges to overcome:

- Entity extraction classically suffers from a disambiguation challenge (e.g. is it Paris, France or Paris, Texas?), but there is also much literature on solving and overcoming these problems. The naïve, simple usage of DBPedia Spotlight used in this experiment benefitted from no customisation of the parameters used nor from any attempts to solve ambiguity problems.
- It is less clear as to whether we can narrow down which are the primary topics or entities in a piece of content. Some search engines use heuristics such as preferring entities in titles and opening paragraphs, but this is still a fuzzy approach. In this research, a simple "topic" predicate from the FOAF vocabulary was used and thus gave equal status to all entities, assuming they are each equally a topic of the content. Better research in this area might provide more effective ways to discern a "topic" relationship from, say, a looser "mentions" relationship.
- All entity extraction technologies mentioned so far have focused around text content only. The ability to extract topics from audio or video content is much more complex and is the subject of whole other fields of research. Some very promising techniques were prototyped by Raimond et al.[13] to extract semantic data from the BBC World Service archive using a novel combination of text-to-speech analysis, machine tagging and crowdsourced user feedback at scale. There could be much opportunity to applying such research more generally, for example across BBC iPlayer catch-up content, which can then feed into better data mining.
- Effective extraction of the content "body" text must be performed to prevent entities being found in supporting page furniture such as site-wide navigation links. A single failure to remove an element common to multiple pages will result in all those pages being "tagged" with that concept.

The final drawback is worsened when elements are across just one class of pages (as observed with the *Listen* problem in section 5.1) as at least entirely global entities will not pass feature selection. When a navigation element only

appears in part of the corpus, the system might erroneously decide this is an informative feature.

As noted in section 5.2.1 with semantics that expose the nature of the construction of the page, a supporting element on a particular class of pages might actually be useful information. To refer back to the *Listen* example – where a "Listen" button for radio stations on radio episode catch-up pages is mistaken for a song with that title – it can be argued that the clustering model "learned" something common between all these pages. It could be said that we need not worry about the technically incorrect semantics that it "thinks" the pages are about that song if the end result is a useful feature that understands there is some commonality between radio episode catch-up pages and it is suitable to consider them as related content suggestions when a user starts on such a page.

This is an overall advantage to applying semantics to data mining: we do not need our machines to gain a perfect "understanding" of the content so long as the so-called understanding is *good enough* that applications of the models – such as suggesting related content – ultimately appear to be behaving intelligently in the average case.

This conclusion is reminiscent of John Searle's famous *Chinese Room* thought experiment[17] that illustrates a computer programme's potential to appear intelligent even when following deterministic rules and lacking the deep understanding and consciousness that we perceive as uniquely human.

5.2.3 Hyperlink Relationships

Exploratory analysis does not highlight any major strengths from hyperlink exraction alone. Common properties centre almost entirely around links to common pages such as FAQs, help pages, etc. due to the lack of constraints in the system implemented to narrow the extraction to "interesting" links from within the content itself.

This extraction technique might well perform better on a website such as Wikipedia where articles contain a large number of links to other articles, but BBC content employs links within article bodies very rarely. Other news websites such as The Guardian make more use of hyperlinks within articles to other articles and topic aggregate pages, so results for content from The Guardian website might look very different to those for BBC content.

5.2.4 Enrichment by Dereference

The most striking observation of all clusters from enriched RDF graphs is that they can have orders of magnitude more features than their unenriched counterparts. This was especially true in the case of enriched RDF graphs obtained via entity extraction. This is due to the fact that any lengthy article may yield ten to fifty entities, perhaps more. As DBPedia URLs were used for entity tags in this experiment and DBPedia can have very rich graphs for popular entities, it is not difficult to extrapolate an assumption that the final feature sets will be very large.

The size of these feature sets makes it challenging to analyse in an exploratory sense, but the comparative evaluation in chapter 6 hopefully proves whether the size of the RDF graphs and resulting feature sets is ultimately harmful to the data mining process.

Enrichment via dereference was less dramatic for in-page semantics themselves extracted via dereference. Whilst not called out specifically in section 5.1, a general trend was for those semantics exclusively to contain literals as values.

This is a consequence of relying on HTML meta tags and other such mechanisms for social media sites. Richer markup approaches such as RDFa and Microdata allow for entities identified by IRIs and URLs to be the objects of any triples, but since these were employed rarely in relation to metadata for social media, they appeared less often in the resulting clusters.

Finally, inferred semantics from hyperlinks would solely contain other BBC page URLs as objects (because the implementation explicitly filtered for BBC URLs only) and thus gained all the advantages and difficulties with extraction via dereference. A casual inspection of the clusters suggests that enrichment here did not therefore add as much additional value as was the case with entity extraction.

5.3 Recommendations for Production Use

It can be argued that building an experimental system full of flaws and drawbacks is not just more desirable, but essential for research and learning. However, if a similar system were to be built as a production-ready application delivering real benefits to customers in industry, there is clear need to do more work to overcome practical limitations.

Thus this section concludes this chapter with some recommendations based on all learnings so far on how a media organisation might want to start applying data mining of their content using semantics.

5.3.1 Embedded Semantics

The Semantic Web movement provoked some pragmatic scepticism[9] over the years, mainly focused around the expectation that content authors are unlikely to create alternative data for machines on top of their primary objective of creating content to be consumed by people.

The findings and analysis presented so far in this chapter suggest there are modern incentives around summarisation of content (e.g. Twitter "cards" when content is embedded in a post) and discovery (both Facebook and Twitter show trends in topics discussed in news items posted to their networks). Recognition of this shift in incentive and consequently the presence of more metadata than before does not appear to be noted so much in the academic literature.

Haas et al.[7] provided evidence in 2011 that fewer than four percent of pages retrieved via Yahoo US search contained RDFa markup. In a media organisastion that has no culture of technical employees with an interest in the semantic web or linked data, there may be no pages with semantic metadata purposely added. However, if we embrace meta tags intended for social media websites, exploratory analysis of BBC metadata in this research suggests a ubiquity of such semantics driven entirely without the academic push behind the Semantic Web and Linked Data.

We can therefore see social media annotations as compromise between benefits of semantic markup and the pragmatic concerns that authors will not add hidden metadata without clear, immediate benefits. It is hoped that the reults

of this research show that there are some baseline benefits to an initial round of data mining on those semantics alone and gaps and failures can create the incentives for web page authors and engineers to improve their semantics and see immediate benefits (e.e. their pages get promoted more readily in pan-site signposting such as sets of "related content" suggestions).

In summary, the recommendation for the media industry here is to lead with building insights based on what data are already in place and look to improve over time once the incentives to do so are in place. Gaps to consider are where semantics are missing altogether and a large gap observed in BBC content alone is a lack of linked data between distinct areas of the website.

For example, programme information and catch-up pages are excellent at describing themselves as being TV or radio programmes and BBC News has the early stages of pages that aggregate News over a given topic, but there is no cross-linked data where concepts are covered both in a news article and in a programme. This will emerge in organisations where distinct teams and divisions work on very distinct parts of their output, so any data mining concerned with the whole corpus needs to highlight issues from lack of cross-linking as much as possible, particuarly if that data mining is driving a real website feature that will see immediate improvement if such cross-linking is implemented.

5.3.2 Entity Extraction

The clearest requirement here is to ensure that entity extraction is run on only the main textual content of any page. With sufficient experimentation, an organisation might be able to tolerate the false positives on a more naïve implementation that processes the whole page. Over time, the system could evolve with bespoke content adaptors common in enterprise integration so as to extract the pure textual content directly from the source system and thus bypass the need to remove noise from around the article.

Also, there will be a need to spend time tuning and configuring the entity extraction to produce the right quantity and quality of entities for each content item. There is no clear single approach from this initial research alone and organisations will need to experiment on their own content.

5.3.3 Hyperlink Relationships

The only conclusion from this particular experiment is that hyperlink relationships are a poor indicator of the nature of the content in the BBC content corpus. An organisation might only want to make use of these relationships if its content authors habitually include links within articles to other articles or pages.

CHAPTER

SIX

EVALUATION

This chapter builds on the more immediate analysis in chapter 5 of the clusters produced by evaluating the results across a group of people. The system designed and built is also a subject of evaluation, particularly any flaws or potential improvements that emerge from having run the software and reviewed its results.

A higher-level goal of this research was to perform some more qualitative evaluation of those clusters by obtaining human feedback on a potential application thereof. The application evaluted was a feature that would take a page a user is currently reading and suggest related content based on their being members of the same cluster. Mock-ups of such a feature were generated and evaluated via a survery as described in section **??**.

The remaining sections of this chapter then discuss some issues with the design and difficulties arising during implementation respectively.

6.1 Qualitative Survey

An online survey was produced where a user is shown a sample page from the BBC website and for each cluster of which that page is a member, a row of up to four "suggestions" was presented to users.

In this section, some details around the design and sampling criteria for the survey are discussed initially, followed by analysis of the results. The section concludes with further qualitative feedback and observations that arose in survey responses.

6.1.1 Survey Design

An initial challenge for generating the survey was that it was necessary to present to users a view starting from a particular content item. This as effectively an inversion of the analysis from chapter 5 where a top-down overview of the clusters was performed, with some drill-down into more depth in some cases.

A reasonably arbitrary sample size of 20 content items was chosen for which it was therefore necessary to choose 20 *pages* that were known to be members of one or more clusters. It was also desirable to ensure that multiple clusters

are evaluated in one survey question, so items that were placed into clusters by multiple instances of the data pipeline were preferable.

It was also important to ensure that the overall set of items chosen provided coverage across all permutations of the data pipeline and that some diversity of content was present (e.g. all items are not simply news articles without any representation of TV and video content).

A further additional property chosen for the sample set was that a diverse range of cluster *cohesion* values were present in the set. This arguably allows the survey to present for evaluation clusters that the machine learning applied itself believes to be poor as well as ones it believes to be cohesive. This controls for coincidences where only good clusters were chosen for one approach and another had its least cohesive results chosen. It also has potential not just to compare which approaches receive the most relevance votes, but also to look for where those approaches' results produce clusters whose cohesion measure correlate with anything meaningful to humans.

The above sampling heuristics were applied programmatically by first inverting the cluster data structures such that there is a known list of all content IRIs across the whole results set and the list of clusters of which they are members are keyed against those IRIs.

A composite uniqueness key was then created for each membership those content IRIs have in the form:

$$u(iri, cluster) = (approach, cohesion(cluster), category(iri))$$

where *approach* is a unique key referring to a particular permutation of the data pipeline being evaluated and *category* is a rough, heuristic function that tries to assess which part of the BBC website "owns" that content item, e.g. BBC News, iPlayer, Weather, Sport. Each item would have a uniqueness key attached to it for each cluster of which it is a member.

Content items were initially scored based on how many clusters of which they were a member:

$$score(iri) = 10 - |5 - |\{c \in C \mid iri \in c\}|\,|$$

where C is the set of all clusters produced. This gives us a score of 10 when the number of clusters in which the item appears is the ideal value of five. The justification for this optimum is based on the lower end of the memory capacity limits suggested by Miller[10] (Miller's Law).

The item is thus "punished" when it appears in only one cluster (thus the question would only be testing one approach and has less value as a question) or if the item appears in too many clusters (asking people to compare between too many items might be overwhelming and reduce the potential for people to answer accurately). As this is only a heuristic, the sampling code still chose some items nearer this extremes, thus an absolute cap was implemented in the survey itself to ensure a random sample of seven clusters was chosen for evaluation when items appeared in a number greater than seven.

Starting with the highest scoring so far, the sampling algorithm then updated each score based on the number of unseen unique keys it would introduce to the final sample if it were chosen:

$$score'(iri) = score(iri) \times \frac{15 - |keys(iri) \cap K|}{2}$$

where $keys(iri)$ is all uniqueness keys for an IRI:

$$keys(iri) = \bigcup_{c \in C} u(iri, c)$$

and K is the cumulative union of all keys so far generated by invocations of of the *keys* function so far, with its initial value being the empty set.

Some of the constants used were tweaked based on what appeared to generated a desirable, mixed set of sample results such that they are a little better than arbitrary values. Once all IRIs were updated with the new scores, the final sample of results for evaluation was simply the content items with the twenty highest scores.

6.1.2 Survey Results

TODO

6.1.3 Further Observations and Discussion

TODO

6.2 Design Limitations

Throughout chapter 3, the design of the data pipeline and overall software architecture was outlined and known trade-offs and limitations were highlighted. Further design trade-offs due to the systems intended use to be experimental rather than production-quality in an enterprise were also covered.

This section highlighted some design issues that were noted only after the system was built and how research and industry alike might learn from these problems. Such problems might be addressed if the results of this experiment are to be researched further or indeed if such a system were to be built in a real media organisation. Such learnings for production use should follow on from the recommendations in section 5.3 around applying the data extraction techniques used and provide more general software design recommendations.

6.2.1 Performance

The experimental system ran far slower than would be acceptable in an enterprise system and even to the extent that it would limit developing this research to the scale where it is processing millions of items – a target that would be ideal for further proving the suitability of semantics and machine learning on real media content corpora.

A better-performing system might have allowed even this experiment to consider larger numbers of items, although even a relatively small sample size should sufficiently indicate which techniques look more promising for developing a more efficient system that focuses thereon (and save time by dismissing techniques that show no promise).

The performance difficulties that arose from the design and concept broadly break down into two problems:

1. lack of consideration for data structures for the feature sets and clusters; and

2. network latency bottlenecks.

The first problem stems from the fact that the design was precise around the in-flight data in the pipeline, less consideration was given for how to store the resulting feature sets and also the clusters. The feature sets were stored as simple JSON objects on files on disk, which is perfectly fast for writing and reading, but there was no pre-calculated distance matrix between those objects. The clusters were also stored in a naïve fashion of larger JSON objects that embedded their members' data for quick display of those clusters and calculation of things like cohesion.

A distance matrix is a $O(n^2)$ operation in terms of computational complexity, which scales poorly, particularly for a continuous ingest of data where each new item invalidates the matrix. The design of the system implemented in this research was built around an assumption that we wish to have a continuously-running data pipeline that can update as new content is published. Recalculating the distance matrix on each new item would effectively push the complexity up to $O(n^3)$ which is clearly undesirable.

With this in mind, a cut-off for content ingest had to be determined so that the distance calculations and then the clustering could be invoked. The system implemented does not save any intermediate data such as distances, so adding in a single new item or cleaning out any duplicates found required all the results to be regenerated from the start.

A production version of this system would have to look at a distance matrix and cluster set structure that allows for real time updates of new and updated content, efficiently changing only parts of the matrix affected (e.g. a new item only needs to calculate its distances with each existing item and does not affect existing pairs). This could be a promising direction for further research.

The other issue of network latency became very apparent in particular when enrichment via object dereferencing was enabled. Whereas an enterprise integration effort between two different data stores operates on bespoke software engineered in a controlled network, extracting RDF data via a URL dereference is susceptible to network latency over the public Internet. Also, HTML pages with embedded semantics have a very poor ratio of semantic metadata to irrelevant information – a document DOM focuses a lot on marking up the content for web browsers.

There is still a compelling case for the *decoupling* achieved by using standards like RDFa and the *abstraction* HTML and HTTP provide over the underlying data stores, reducing the need for bespoke clients. However, any extraction of that data needs to tolerate or factor in the increased latency for this approach.

For extraction, the latency of a single HTTP request (or two or three if HTTP redirects are issued) will only cause a bottleneck if new content is published or updated at a rate faster than the pipeline can process. This seems unlikely even in an organisation like the BBC, particularly if the system implements structures such as queues to handle bursts of input, processing them in quieter intervals.

However, for enrichment via deference, an RDF graph produced might have hundreds of objects needing dereferenced via HTTP request. A simple cache

layer might prevent the system repeatedly requesting for the same URL in a short period, but this kind of latency still means it could take of the order of minutes for a single item to get through the pipeline.

It is not clear that there is a single answer to handle this as it is likely a combination of factors need optimised to make this form of enrichment acceptable:

- It might be that hundreds of objects linked via predicates creates too much noise in general. If those objects have come mainly from entity extraction, should entity extraction be tuned to bring back only the "main" topics of a page?
- Chapter 5 discussed how many of the extraction techniques (particularly deference) produced a lot of irrelevant facts (e.g. about the page itself, not the content) all of which would be candidiates for deference of the objects are IRIs. Effectively filtering out such noise would reduce the work spent enriching irrelevant things further.
- A simple architectural solution is to parallelise the HTTP calls as much as possible such that an item's latency is only as long as the slowest response time. There are obvious limits to this (we should avoid overloading the server even if it is scaled for millions of visitors per day), but any parallelism is clearly better than a hundred sequential HTTP requests. Parallelism includes distributed architecture over multiple servers as much as concurrent programming paradigms.
- An organisation should probably ensure a system is physically near any servers serving HTTP requests to reduce the overall TCP packet latency. This optimisation will add up over time, but is a challenge for a large organisation such as the BBC with systems of varying ages distributed over multiple locations [1]

Even with all the optimisations possible, it is unescapable that multiple HTTP requests to assemble an RDF graph is slower than a single query to a data store's query engine (e.g. SQL) that returns all the information needed in a single response. The benefits of using semantics, HTTP and other standards – such as decoupling, simple enrichment, reuse of standard technologies – need to be effectively researched and promoted, otherwise enterprise organisations will continue to reach for the seemingly obvious solution of building software that queries one database to push to another.

6.2.2 Noise

The semantic web and more recently Linked Data are arguably built around the idea that they embrace the loose architecture of the World Wide Web. With that looseness – open world assumption, no schema constraints, etc. – we

[1] Industry has long outgrown the notion that enterprises like the BBC have a single "www" server as in the early days of the World Wide Web. However, the www hostname still needs to resolve to a single set of IP addresses for something that fronts the rest of the network like a load balancer. Effective caching in this front layer and ensuring our data pipline lives close to that cache might be sufficient in this case. An example of a front cache might be something like the Varnish HTTP cache: https://www.varnish-cache.org/

cannot assure strict quality and integrity of the data as would be possible in a closed-world database.

The data pipeline design in chapter 3 has no requirements nor consideration for how to handle facts that are erroneous, missing nor meaningless (e.g. see the discussion in section 5.2.1 of the system finding facts about the pages' structure rather than content metadata).

In section 5.1.3, it is argued that noise might not have as much impact as one would assume given the fuzzy nature of machine learning in general. This argument is important to consider, but here we explore the issue of noise on the assumption that it *is* a problem as it is important to evaluate how noise fed into the system in the first place and how we might tackle it.

Erroneous facts may be non-trivial to distinguish from real facts, but there is scope for research on effective methods for reducing that noise. In an enterprise, we need not exclude the benefits of enterprise integration development to *support* this approach based on semantics and web standards.

Whilst there is much gained from being able to reason about the entire breadth of an organisation's content using semantics, it does not preclude writing bespoke business rules that strip out known, erroneous data or provides a framework against which to evaluate the likelihood of a fact being useful or true. At the very least, there could be some work to develop ontologies and schemas that detect inconsistencies in the data, but in a way that adds value to a working system rather than such modelling being a barrier to building the first iteration of the system.

Facts that are missing are arguably not a "noise" issue, but still a data integrity concern. If an item lacks key information, it may be categorised incorrectly. It should be noted that the data pipeline designed here did nothing to address this deliberately and the desire was to evaluate the open world characteristics of semantics without trying to fill in the gaps.

As with the erroneous facts discussion above, there is much case to be made for designing such a system so that it is extensible enough with customisations and bespoke software to aid where it appears to be struggling – once its initial behaviour is evaluated. In the case of the data pipeline designed in chapter 3, any in-house databases and APIs can be integrated as additional extraction techniques to be joined via union to more general approaches such as dereferencing.

Data being irrelevant might be the larger of the three concerns here as that greedy machine learning algorithms can get stuck on such noise. In section 5.1.1, we see how one approach caused the clustering to get stuck on the fact that all items linked to a common FAQ page.

Other data attributes such as the Twitter Cards metadata (see section 5.1.2) can contain very broad things that are declared on all pages that use this vocabulary (e.g. `card=summary` is always used to indicate that a given page is using these meta tags) might cause a greey algorithm to focus too long on grouping together all pages that happen to choose to implement this Twitter feature with less concern about the more interesting properties.

CHAPTER

SEVEN

CONCLUSIONS AND FUTURE WORK

7.1 Challenges and Opportunitues for Semantics in the Enterprise

7.2 Suitability for Semantics and Data Mining in Media Organisations

7.3 Suggestions for Initial Implementations in the Enterprise

7.4 Future Research

7.4.1 Better Enrichment

7.4.2 Patterns for Reducing Data Noise

7.4.3 Deeper Study of Machine Learning on Semantics

BIBLIOGRAPHY

[1] Tim Berners-Lee. Linked data-design issues (2006). *URL http://www. w3. org/DesignIssues/LinkedData. html*, 2011.

[2] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[3] Frederick P Brooks Jr. The mythical man-month (anniversary ed.). 1995.

[4] Joachim Daiber, Max Jakob, Chris Hokamp, and Pablo N. Mendes. Improving efficiency and accuracy in multilingual entity extraction. In *Proceedings of the 9th International Conference on Semantic Systems (I-Semantics)*, 2013.

[5] Brian S. Everitt, Sabine Landau, Morven Leese, and Daniel Stahl. *Hierarchical Clustering*, pages 71–110. John Wiley & Sons, Ltd, 2011.

[6] Roy Fielding and Julian Reschke. Hypertext transfer protocol (http/1.1): Semantics and content. 2014.

[7] Kevin Haas, Peter Mika, Paul Tarjan, and Roi Blanco. Enhanced results for web search. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 725–734. ACM, 2011.

[8] Christoph Kiefer, Abraham Bernstein, and André Locher. *Adding data mining support to SPARQL via statistical relational learning methods*. Springer, 2008.

[9] Catherine C Marshall and Frank M Shipman. Which semantic web? In *Proceedings of the fourteenth ACM conference on Hypertext and hypermedia*, pages 57–66. ACM, 2003.

[10] George A Miller. The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological review*, 63(2):81, 1956.

[11] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[12] Heiko Paulheim and Johannes Fümkranz. Unsupervised generation of data mining features from linked open data. In *Proceedings of the 2nd international conference on web intelligence, mining and semantics*, page 31. ACM, 2012.

[13] Yves Raimond, Tristan Ferne, Michael Smethurst, and Gareth Adams. The bbc world service archive prototype. *Web Semantics: Science, Services and Agents on the World Wide Web*, 27:2–9, 2014.

[14] Eric S Raymond. *The art of Unix programming*. Addison-Wesley Professional, 2003.

[15] Giuseppe Rizzo and Raphaël Troncy. Nerd: a framework for unifying named entity recognition and disambiguation extraction tools. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 73–76. Association for Computational Linguistics, 2012.

[16] Stuart Russel and Peter Norvig. *Artificial Intelligence: A Modern Approach, 2010*.

[17] John R Searle. Minds, brains, and programs. *Behavioral and brain sciences*, 3(03):417–424, 1980.

[18] Toby Segaran, Colin Evans, and Jamie Taylor. *Programming the semantic web*. " O'Reilly Media, Inc.", 2009.

[19] Manu Sporny, Dave Longley, Gregg Kellogg, Markus Lanthaler, and Niklas Lindström. Json-ld 1.0. *W3C Recommendation (January 16, 2014)*, 2014.

[20] Ian H Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005.

APPENDIX

A

PROPOSAL

APPENDIX

B

ETHICS FORM

Please type

UNIVERSITY OF BRADFORD
MAKING KNOWLEDGE WORK

EC ……

COMPUTING PROJECT ETHICS CHECKLIST

This checklist is designed to help you to decide whether or not ethics approval is required and, if required, to decide on the appropriate ethics review procedure –
please read Annex 0 and 1 on page 5 before you complete this form

Please Note:

α) This Checklist should be completed for **all** projects.

β) All questions on this checklist should be completed.

χ) Contact details (email address) should be given for supervisor and student.

δ) Supervisors should read and sign this checklist (in the correct box – EITHER/OR – not both boxes) BEFORE it is submitted to the Ethics Administrator for sign off by the Chair of the Research Ethics Panel.

ε) Guidance on the 2 different ethics review procedures that together make up the University's Ethics Review System (i.e. 'University' and 'NHS') is available on the University Ethics website.

φ) If your project will involve human tissue/biological fluids you should contact the UoB Designated Individual for the HTA licence, Dr Sue Boyce for advice (s.g.boyce@bradford.ac.uk or on 01274 235879)

γ) **If this Checklist is NOT correctly completed, it will be returned to you unauthorised.**

Project Title:……

Name of Supervisor: .…….....

Contact Details – email address ……

Department/School ……

Name of Student :.…….....……

Contact Details – email address ……

Has student attended appropriate ethics training (Project Ethics lecture): Yes ❒ No ❒

Please give summary of project (max 150 words)**:** *to be typewritten*

Q1	**Is the proposed project an <u>empirical research</u> project involving people?** • will the project include primary data collection from human subjects, their data or their tissue? • Will it constitute an 'investigation undertaken in order to gain knowledge and understanding'? (this includes work of educational value designed to improve understanding of the research process) If you answer 'Yes' to Q1 ethical approval may be required, move to Q2. If you answer 'No' to Q1 then a research ethics review is <u>not</u> required and you are not required to submit this checklist. You should still complete the form as Appendix B of your dissertation. ***Note**: there may be occasions where a project is not defined as research but still raises ethical issues – please submit for review if you think this is the case.*	Yes ❒ No ❒ Yes ❒ No ❒ Yes ❒ No ❒
Q1a	**Is the proposed project an audit involving humans?** A more detailed definition of Research, Audit and Service Evaluation is available on the University Ethics website. If you answered 'Yes' to Q.1a then ethical review is required.	Yes ❒ No ❒
Q2	**Will the project involve the <u>NHS</u>?** ee Research Ethics and Governance in NHS and Social Care page of the website	Yes ❒ No ❒

	If you answer 'No' to Q2 move on to Q3 If you answer 'Yes' to Q2 ethical approval will be required by NHS Research Ethics Committee (REC)	
Q3	**Will the project involve any of the following in the UK:** Testing a medicinal product Investigating a medical device Taking samples of human biological material (e.g. blood, tissue) Prisoners or others in custodial care (e.g. young offenders) as participants Adults with mental incapacity as participants Other vulnerable groups (e.g. vulnerable children) as participants If you answer 'Yes' to Q3 ethical approval will *usually* be required through Ethical Tissue or NHS Research Ethics Committee (REC) or where the project includes participants which need approval under the Mental Capacity Act approval will be required by the Social Care REC. If you wish to source material from Ethical Tissue at the University, they can be contacted on 01274 235897 or visit www.ethicaltissue.org See information specific to research in Social Care on the University Ethics website *If you answer 'No' to Q3 move on to Q4*	Yes ❒ No ❒
Q4	**Will the project involve human participants and/or human data (*but not accessed through the NHS*)?** *If you ticked 'Yes' please give details of:* 1. *Interviews (how many, how long will they last),* 2. *who you intend to interview,* 3. *where interviews will take place and* 4. *attach interview guidelines or the questions you intend to ask:*	Yes ❒ No ❒
Q5 **Q5a**	**Will the project involve human tissue (but not requiring NHS approval – see Q3)?** If you answer 'Yes' to Q5 University ethical approval is required If you require advice on human biological material please contact Human Tissue Act (HTA) Designated Individual: Dr Sue Boyce [s.g.boyce@bradford.ac.uk] on ext 5897 or visit www.ethicaltissue.org If you answered 'Yes' to Q5, is the human material over 100 years old and archaeological? If 'YES' please refer to the Biological Anthropology Research Centre (BARC) guidelines at http://www.barc.brad.ac.uk/BARC_human_remains_policy.pdf	Yes ❒ No ❒ Yes ❒ No ❒
If you answer 'No' to Q5 and have answered 'No' to Q2, Q3 and Q4 ethical approval is not required.		

PLEASE COMPLETE and SIGN *ONE* of the two boxes below
(in the case of a student project, we do require a Supervisor's signature in whichever box is relevant, before we can have the checklist signed off by the Research Ethics Panel):

1. I have discussed this project with my student AND/OR
2. I confirm that there are no ethical issues requiring further consideration.

(Any subsequent changes to the nature of the project will require that the Panel are informed of all changes)

Signed by (Supervisor):

Signature: ……..…. Date: …….

PLEASE PRINT NAME ……

Signed by (Student):

Signature: ……..……. Date: …….

PLEASE PRINT NAME ……

OR

I confirm that there are ethical issues requiring further consideration and will either:

1. refer the proposal to Ethical Tissue, or,
2. fill in and submit a full ethics application to be considered by the appropriate Research Ethics Panel.

Signed by (Supervisor);

Signature: …….. Date: …….

PLEASE PRINT NAME ……..

Signed by (Student):

Signature: …….. Date: …….

PLEASE PRINT NAME …….....

APPENDIX

C

SUMMARY