# Improving content discovery through combining linked data and data mining techniques

Ross Fenning
British Broadcasting Corporation (BBC)
MediaCityUK
Salford, United Kingdom
ross.fenning@bbc.co.uk

## ABSTRACT

A comparative study of multiple approaches for extracting linked data from web content published by the BBC for the purposes of applying machine learning to cluster related content.

This project designs and implements a process by which semantic data about any piece of online media content can be extracted into RDF models in multiple ways and feature sets for machine learning can be generated from that data. The clusters produced by fourteen different permutations of techniques are evaluated and a potential application of suggesting related content is mocked up for qualitative evaluation.

It is concluded organisations can very easily make use of embedded semantics within web pages due to better semantic properties being made available in recent years because of pressure from social media sites such as Facebook and Twitter. These embedded semantics provide a good baseline for simple learning that can group content based on broad categorisation.

A simple application of entity extraction also shows promising results in evaluation and is highlighted as an area for organisations to explore tuning to provide a powerful complement to embedded semantics.

## 1. 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. Examples include providing a text-based search, hierarchical categorisation and tailored recommended content based on past behaviour.

### 1.1 Problems

- Content across multiple content management systems, in differing formats and data models.
- Content items are in fairly opaque formats, e.g. video and audio content.
- Content is being published continuously, so any analytical process that operates over all content (e.g. machine learning) may need to be run periodically or in an incremental fashion.

### 1.2 Hypothesis

- Research and software tools around the concept of *Linked Data* can aid us in rapidly acquiring a broad view of an organisation's content.
- We can establish at least a naïve mapping of an RDF graph representing a content item to an attribute set suitable for data mining.
- 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 make use of semantic metadata added to pages to provide better presentation of that content as it is shared on social media.

## 2. BACKGROUND

Data mining activities such as machine learning rely on structuring data as *feature sets*[1] – 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 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 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 Fürnkranz proves to be a promising strategy given we can acquire an RDF graph model for content items in the first place.

## 3. DESIGN

An experiemental system was designed and built around the following high-level requirements:

1. gathering (meta)data about all of an organisations content items;
2. further enriching that metadata with information not explicitly present in the content item itself; and

3. applying machine learning to that content metadata to gain new insights about that content.

The system was also designed with the end goal of being able to present users with alternative content items deemed similar to any given item they are currently ready or watching. This provided a focus to ensure some industrial application of the experiment and also a means by which to evaluate the results in that results can be scored on how they perform at driving such a website feature.

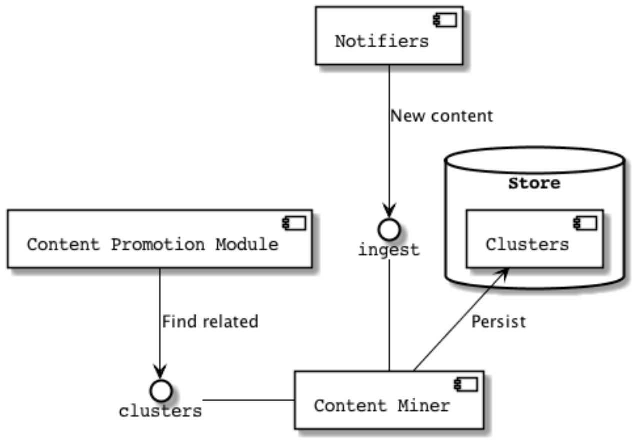

Figure 1: High-level component diagram with interfaces for each use case

Figure 1 shows a high-level view of a "Content Miner" software application. 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. Once the application has meaningful clusters of content, we can imagine another system that can query this miner application to provide some view of related content given some initial content item.

## 3.1 Data Pipeline

This input side of the miner was conceptually modelled as a data "pipeline" that takes an IRI identifying a piece of web content at the source end and produces feature sets at the other end. In this pipeline, the three primary data structures employed are the *IRI*[1] (in many cases the public URL for the content's primary page is sufficient), the *feature set* (modelled as a schema-less set of key-value pairs) and the *RDF graph* as an intermediate structure for manipulating along the pipeline.

The data pipeline was then designed inductively, introducing potential methods for creating RDF graphs from web content and also enriching those graphs.

Listings 1 shows a SPARQL query inspired by Paulheim and Fürnkranz[2]. This query generates a boolean `true` value for any properties that match and implies `false` for those that do not and we then flatten this into a simple feature format: `?p_?o=true`, e.g. `rdf:type_schema:Article=true`

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

```
SELECT ?p ?o
WHERE { ?iri ?p ?o . }
FILTER isIRI(?o)
```

Listing 1: Generates field `content_?p_?v` with value `true`

Note that this ignores triples whose objects are literals, for which case the similar strategy as shown in listings 2 was employed to produce features of the form: `?p=?v`.

```
SELECT ?p ?v
WHERE { ?iri ?p ?v . }
FILTER isLiteral(?o)
```

Listing 2: Generates field `content_?p_?v` with value `true`

With a feature generation strategy in place, we can inductively build up the pipeline of how we create RDF graphs for that generation.

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

Listing 3: 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. This provides our base case in the inductive design. This enables a *null* pipeline as shown in Figure 2 that only produce feature sets with a single property containing the item's IRI.

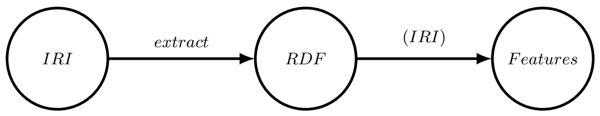
$$\begin{tikzcd} \textit{IRI} \arrow[r, "\textit{extract}"] & \textit{RDF} \arrow[r, "(\textit{IRI})"] & \textit{Features} \end{tikzcd}$$

Figure 2: Null feature generator

We can then add in, one-by-one, the following potential extraction techniques:

- Extraction by dereference of the IRI, parsing any RDFa or microdata semantics found embedded in the response.
- Use entity extraction such as DBpedia and infer triples of the form $(IRI, rdf{:}about, Entity)$
- Infer a relationship between pages if they have hyperlinks to each other: $(IRI_1, ex:related, IRI_2)$

We also propose enrichment techniques such as the following:

- Dereference every object IRI for triples of which the given content item is the subject.
- Infer facts based on OWL/RDFS ontologies and schemas, e.g. infer an item is about animals if the entity *Dog* is found and we have an ontology that states that *Animal* is a superclass.

This could lead to a theoretical maximum pipeline as depicted in Figure 3.

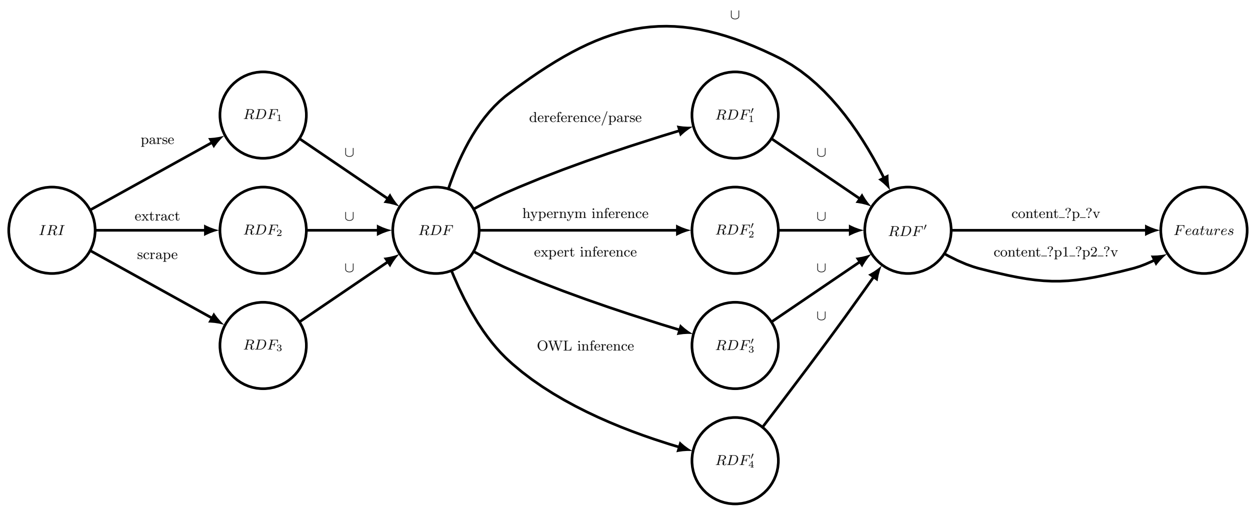

Figure 3: Maximal Data Pipeline

## 4. IMPLEMENTATION

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[3] of doing one job per command and that they were composable via Unix pipes. This allows pipelines to be composed, e.g.:

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

that resemble the entire theoretical data pipeline described in section 3.1 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.

## 5. ANALYSIS AND EVALAUTION

Clusters produced were analysed both by an exploratory analysis of the results and also a survey given to 30 respondents to evaluate mock-ups of suggested, ”related” content based on a hypothetical starting page and promoted links to other pages in the same cluster.

Exploratory analysis showed a large variety in the sizes of clusters produced. Hyperlink relationships fell down on finding links to common pages like FAQs which then allowed the greedy clustering algorithms to spend too much time grouping all pages that linked to those very common links.

Embedded semantics did well where annotations were put in place for social media sharing. There was some difficulties with noise from semantic vocabularies concerned with describing structure of the page or even schema properties about the properties themselves.

Entity extraction was strong at finding larger numbers of triples, but required some monitoring to avoid false positives from words common to many pages (a large number of BBC pages mention Twitter, for example, where there are controls prompting users to share content on that platform). Other examples included a “Listen” button present on all BBC pages about radio programmes.

Enrichment by dereference had the same shortcomings of embedded semantics extraction and whilst it did add value in some cases, it generated an explosive increase in the number of triples that had a large impact on system performance.

Results from the survey supported the idea that embedded entities pleased users where it was able to group things on very structural, categorical aspects of the data (e.g. BBC News vs. BBC iPlayer or even “Football” category within BBC Sport.)

However, many users preferred items to be linked more strongly by theme or topic. Some useful discussion with user experience experts supported this idea and in some cases this was achieved due to entity extraction.

## 6. CONCLUSIONS

Extracting embedded semantics provided enough initial benefits for machine learning in that a model can be produced that understands the broad categorisation of media content. That it is far simpler to implement than bespoke enterprise integration projects, proves its worth as an initial iteration of a more intelligent system.

Organisations would benefit from implementing this baseline for gaining insights into its content and then spend later iterations looking at introducing entity extraction, monitored carefully for appropriate tuning.

Later iterations can look at some enterprise integration work to improve the quality of the most poorly-performing content items. there is certainly need for future work to evaluate some of the more complex enrichment approaches.

## 7. REFERENCES

[1] C. M. Bishop. *Pattern recognition and machine learning.* springer, 2006.

[2] H. Paulheim and J. 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.

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