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

Ross Fenning

November 11, 2015

# CONTENTS

# 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.

# 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*[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 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   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.

Notably, Kiefer, Bernstein and Locher[2] proposed a novel approach called SPARQL-ML – an extension to the SPARQL[4] 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.

---

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

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:

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

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

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 F urnkranz[3]. 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.

## 2.2   RDF in the enterprise

# 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

## 3.2 Use Cases

## 3.3 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 5, 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.

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 2.2 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[5] syntax will be used.

```
{
  "@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

8

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 **??**.
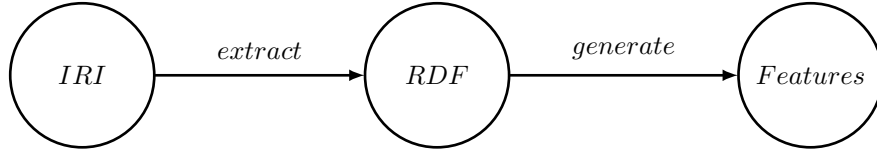


Figure 3.1: Null feature generator

Note that Figure **??** 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.
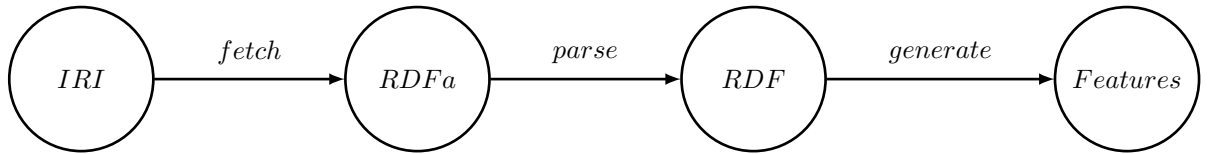


Figure 3.2: Semantic web content miner

## 3.4   Technical Architecture

Figure 3.3: Semantic web content miner with DBPedia enrichment

Figure 3.4: Maximal web content miner

# FOUR

## IMPLEMENTATION

# FIVE

# EVALUATION

CHAPTER

# SIX

# CONCLUSION

# BIBLIOGRAPHY

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

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

[3] 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.

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

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