Report

Semantic Annotations in Today's Web

Alexander Trautsch

18th July 2016

Contents

1 Introduction

A wide spread semantic web has, since Tim Berners-Lee's article in 2001 [30], remained a dream. Although making documents on the web machine readable clearly has its benefits, it had always the problem of limited support by the people creating websites. People like Aaron Swartz [28] blamed the complex standards that were first introduced for the slow to non-existent adoption of semantic web technologies. The base of semantic web technologies was (and remains) the Resource Description Framework (RDF) [27]. The power of this framework was previously only usable by using its eXtensible Markup Language (XML) or N3 representations. This meant that to contribute to the semantic web, publishing the semantic information separate from the HTML documents via a separate endpoint was necessary.

Some technologies targeted this weak spot by combining the semantic information with the already existing content, e.g., Microdata [3], Microformats [19] and finally Resource Description Framework in Attributes (RDFa) [5]. This mostly solved the problems of complexity but the problem of motivation for the content producers remained. In recent years, due to the adoption of semantic annotations and cooperation by search engines in the Schema.org initiative [23], the motivation for content providers are now mostly provided by search engines, as the main data consumers. With the difficulty and motivation problems at least partially solved, semantically annotated content has seen a steady increase in recent years [13].

In this report we will focus mostly on RDFa and Javascript Object Notation for Linked Data (JSON-LD) as these are the newest and JSON-LD is the recommended format by Google [18]. RDFa is an annotation format for HTML5 and XML that can be serialized into an RDF graph. JSON-LD is strictly speaking not an annotation format as it can be used completely separate from the content. JSON-LD can also be serialized to RDF. We will first introduce some underlying standards that build the foundation of the semantic web. Then we will describe the annotation formats in use today: Microdata, Microformats, JSON-LD and RDFa. Structural semantic HTML tags that were introduced with HTML5 will also be briefly presented. After that we will look in depth at some example applications for semantic annotations to improve search result displays in the Google search engine. At the end we will present a small case study containing real world data and then close with a short conclusion.

1.1 Foundations

The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation [30]. At the core of the semantic web is the RDF graph, Web Ontology Language (OWL), the Universal Resource Identifier (URI) [29] and International Resource Identifier (IRI) [26]. The main purpose of the RDF graph is expressing information about resources. It has a triple data model consisting of subject predicate object.

```
<Göttingen> <is a> <City>.
<Göttingen> <is located in> <Germany>.
```

In the RDF graph example above, Göttingen is the subject, is a, and is located in are the predicates and City and Germany are the objects.

Now we have expressed some information about resources but it is not well-defined, for that we need OWL [2] and IRIs. OWL is based on Resource Description Framework Schema (RDFS) and provides a way to define ontologies. An ontology describes concept but can also include rules which may be used for inference or integrity checking. In the OWL description for our example would be the predicates and the term City. It may also include a rule that a city can be located in a country but not vice versa. To uniquely identify the term city the IRI is used. City will not appear as a string in the document but as a reference to a unique identifier, e.g., http://schema.org/City. Moreover, Göttingen and Germany may also be referenced by unique IRIs.

For querying the graph there is another standard called Simple Protocol And RDF Query Language (SPARQL) [4]. SPARQL is a SQL like query language for the semantic web which we can use in our example to query all cities located in Germany.

Lets look at a more complex example of an RDF graph in n-triple format.

```
@prefix s: <http://schema.org/>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.

<http://www.example.org/bob#me> rdf:type s:Person.
<http://www.example.org/bob#me> s:name "Bob".
<http://www.example.org/bob#me> s:address _:aid.
_:aid rdf:type s:PostalAddress.
_:aid s:streetAddress "1600 Amphitheatre Pkwy".
_:aid s:addressLocality "Mexico Beach".
```

Here we can see that we have some vocabularies in use, schema.org and rdf. We also have 3 literal objects, one for the name and two for the address information. In the example there is only one blank node _:aid which refers to the address. It is only used to assign a

PostalAddress to our bob subject therefore it does not need to be referenced anywhere else. That is a typical use case for blank nodes.

Now we extend that example with a typed literal and another vocabulary.

```
@prefix s: <http://schema.org/>.
@prefix foaf: <http://xmlns.com/foaf/0.1/>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.

<http://www.example.org/bob#me> rdf:type s:Person.
<http://www.example.org/bob#me> s:name "Bob".
<http://www.example.org/bob#me> s:address _:aid.
<http://www.example.org/bob#me> foaf:birthday "1980-01-01"^^xsd:date
_:aid rdf:type s:PostalAddress.
_:aid s:streetAddress "1600 Amphitheatre Pkwy".
_:aid s:addressLocality "Mexico Beach".
```

In this example we added the Friend of a Friend (FOAF) vocabulary to specify a birthday and we also included the XMLSchema vocabulary to add a typed literal. The RDF graph now contains not only the birthday but also the information that the literal is in fact a date.

2 Annotation Formats

There are different annotation formats to semantically annotate content. HTML5 is included here as it also provides a semantic upgrade over older HTML and XHTML formats. The semantics it adds to HTML documents are mostly of structural nature concerning document structure and not annotations to the content.

In this chapter we will describe the different formats that can be used for annotation and explain the advantages and disadvantages of each.

2.1 HTML5

HTML5 introduced some new HTML tags to give more semantic structure to the document. Before HTML5, most layout of the content was done with tags. This is not semantic because the document structure may consist of many parts with different meaning, e.g., navigation, footer, sidebar. In HTML5 there are new tags supporting a more semantic document structure. There is, for example, a <nav> tag to use for navigation elements for the page.

Additional semantic tags are: <aside> for a sidebar, <footer>, <header> for document headers, footers. Also included are more elements to structure content, e.g., <article>, <section>. This is, semantically, a big improvement over the previous HTML versions. As it allows to describe the structure of the document in more detail.

2.2 Microformats

Microformats [19] do not introduce new attributes or tags, instead they introduce a convention and rules to use existing attributes. According to the Web Data Commons [13] 2015 Corpus Microformats are still used often (see Table 1). This may have to do with their simplicity and the clear communication by the team behind them. There is, for example, a dedicated page describing the support for different Microformats by different search engines [20].

This format is not serializable to RDF per-se but there exist some XSLT for transformation of specific Microformats [1].

```
<span class="vevent">
    <span class="summary">The microformats.org site was launched</span>
    on <span class="dtstart">2005-06-20</span>
    at the Supernova Conference
    in <span class="location">San Francisco, CA, USA</span>.
</span>
```

Listing 1: Microformat Example 1

This example shows an event with annotated location, summary and startdate. The values of the class attributes here are defined names by the Microformats team. It also allows the HTML attributes rel and rev, which defines a forward relationship and a reverse relationship.

```
<span class="vcard">
  <span class="fn">Max Mustermann<span>
  <div class="label" style="display:none">
    <span class="type">home</span>
    42 Plantation St.<br>Baytown, LA 30314<br>United States of America
  </div>
  <div class="adr">
    <span class="type">Home</span> Address:<br>
    <span class="street-address">42 Plantation St.</span><br>
    <span class="locality">Baytown</span>, <span class="region">LA</span>
    <span class="postal-code">30314</span><br>
    <span class="country-name">United States of America</span>
  </div>
</span>
```

Listing 2: Microformat Example 2

This example shows the integration of vcard with the hcard format of Microformats. Notable here is that the label is part of the metadata but not shown to the visitor of the website.

HTML attributes used by Microformats

- class
- title
- rel
- href
- rev

The simplicity is a big plus of this approach but there are also some drawbacks. On of them is that it can not be serialized directly to RDF. It also does not have support for language annotations, e.g. @en-US. It is not extensible by the use of Schemas, e.g. the rel attribute in a vcard for a person can contain friend, colleague but these are not extensible. Furthermore, Microformats are a format convention not a standard created by a standardization body like the W3C.

2.3 Microdata

Microdata [3] was created for the purpose of annotating HTML. It introduces new attributes for this task, e.g. itemscope, itemprop. As mentioned already Microdata can be serialized to RDF [25] though there are some drawbacks and it is not fully compatible.

At a high level, Microdata consists of a group of name-value pairs. The groups are called items, and each name-value pair is a property [3].

```
<section itemscope itemtype="http://schema.org/Person">
  Hello, my name is
  <span itemprop="name">John Doe</span>,
  I am a
  <span itemprop="jobTitle">graduate research assistant</span>
  at the
  <span itemprop="affiliation">University of Dreams</span>.
  My friends call me
  <span itemprop="additionalName">Johnny</span>.
  You can visit my homepage at
  <a href="http://www.JohnnyD.com" itemprop="url">www.JohnnyD.com</a>.
  <section itemprop="address" itemscope itemtype="http://schema.org/
      PostalAddress">
    I live at
    <span itemprop="streetAddress">1234 Peach Drive</span>,
    <span itemprop="addressLocality">Warner Robins</span>,
    <span itemprop="addressRegion">Georgia</span>.
  </section>
</section>
```

Listing 3: Microdata Example 1, Source: [8]

Notable in the example above is the use of the schema.org ontology as the itemtype. This declares the itemscope to be of the type schema.org/Person. This in turn defines the attributes and relations that can be used to describe this itemtype. Here there exists a nested itemscope which defines a PostalAddress for the Person.
Attributes used in Microdata are:

- itemid
- itemprop
- itemref
- itemscope
- itemtype
- content
- src
- href
- data
- datetime

Microdata is an easy to use format to annotate HTML with semantic information. It was the most used annotation format in the Web Data Commons [13] 2015 corpus. It supports referencing items via IRI and it also supports the use of ontologies.

A disadvantage of Microdata is that it only supports (X)HTML. Although to be used in XHTML some modifications, e.g. itemscope converts to itemscope="itemscope", have to be made. Another maybe more relevant disadvantage is that Microdata does not support typed literals, e.g., integers dates.

2.4 RDFa

RDFa [5] consists of different standards documents. RDFa-lite [12] which describes a minimal subset of RDFa, RDFa-core [10] which describes the syntax and processing rules for embedding RDF through attributes and some language specific descriptions, e.g., RDFa for HTML [11]. As one would have guessed regarding these documents RDFa is much more complex than the previously described formats. But there are also advantages, RDFa is available for a lot of host languages.

- XHTML1
- HTML4
- HTML5
- XHTML5
- XML
- SVG
- ePub
- OpenDocument

As can be seen in the list above, RDFa is available for a lot of HTML / XML based languages. This of course increases its usefulness as it can be used, for example, not only to annotate the HTML document but also the OpenDocument files that are linked there. In this report we will concentrate on its use in HTML.

RDFa uses these attributes.

- about
- datatype
- profile
- prefix
- property
- resource
- typeof

- vocab
- content
- href
- rel
- rev
- src

The attributes content, href, rel, rev, src are already in HTML and are re-used. In the following examples the RDFa only attributes are in red.

```
<div vocab="http://xmlns.com/foaf/0.1/" typeof="Person"><p>
  <p>
    <span property="name">Alice Birpemswick</span>,
    Email: <a property="mbox" href="mailto:alice@example.com">alice@example.
        com</a>,
    Phone: <a property="phone" href="tel:+1-617-555-7332">+1 617.555.7332</a
        >
  </p>
</div>
```

Listing 4: RDFa example 1

In this simple example we can see how a basic object with a few properties is constructed. We have a blank node of type foaf:Person with attributes foaf:name, foaf:phone.

2.5 JSON-LD

JSON-LD [6] is the format recommended by google [18]. It is embedded into the website as a JavaScript HTML tag, although with its own type.

This allows it to exist separate from the content that is shown to the visitor of the site. Search engines like Google have to be aware of this and check if the content that is present in the JSON-LD is also present in the HTML markup that is visible for the user. Otherwise search engine spam would become a major problem (or even bigger than today).

The syntax of JSON-LD is based on Javascript Object Notation (JSON) [24] and it includes additional keywords to support linked data features. Here is a small example.

```
<script type="application/ld+json">
{
  "@context": "http://schema.org/",
  "@type": "Person",
  "name": "Jane Doe",
  "jobTitle": "Professor",
  "telephone": "(425) 123-4567",
  "url": "http://www.janedoe.com"
}
```

```
</script>
```

Listing 5: Example of JSON-LD

Which serialized into N3 format looks like this:

```
_:b0 <http://schema.org/jobTitle> "Professor" .
_:b0 <http://schema.org/name> "Jane Doe" .
_:b0 <http://schema.org/telephone> "(425) 123-4567" .
_:b0 <http://schema.org/url> <http://www.janedoe.com> .
_:b0 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://schema.org/Person> .
```

We can see that the rdf-syntax-ns for @type is automatically included and schema.org is prefixed automatically.

The second example extends the first with more vocabularies and a typed literal. Of note here is that we extend the shortened @context from above and now we have to specify which vocabulary to use in every place. We also introduced a typed literal here (birthday) which allows us to define the type of the value of this literal.

```
<script type="application/ld+json">
{
  "@context": {
    "s": "http://schema.org/",
    "xsd": "http://www.w3.org/2001/XMLSchema#",
    "foaf": "http://xmlns.com/foaf/0.1/"
    },
  "@type": "s:Person",
  "s:name": "Jane Doe",
  "s:jobTitle": "Professor",
  "s:telephone": "(425) 123-4567",
  "s:url": "http://www.janedoe.com",
  "foaf:birthday": {
    "@type": "xsd:date",
    "@value": "1980-01-01"
  }
}
</script>
```

Listing 6: Example of JSON-LD with typed literal and more vocabularies

```
_:b0 <http://schema.org/jobTitle> "Professor" .
_:b0 <http://schema.org/name> "Jane Doe" .
_:b0 <http://schema.org/telephone> "(425) 123-4567" .
_:b0 <http://schema.org/url> "http://www.janedoe.com" .
_:b0 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://schema.org/Person> .
_:b0 <http://xmlns.com/foaf/0.1/birthday> "1980-01-01"^^<http://www.w3.org/2001/XMLSchema#
```

In the example applications in the next chapter we will present a little more complicated JSON-LD with their respective N3 representation.

2.6 Choosing a Format

The annotation format should be chosen according to the expected data consumers. For example, if those are search engines RDFa or JSON-LD should be chosen (as they are officially supported) and the schema.org vocabulary should be used. According to Web Data Commons [13] the most used format at the moment is Microdata with RDFa coming in a close second, Microformats third and JSON-LD last. As JSON-LD is the format which Google recommends it is bound to grow in adoption in the near future.

Format	Domains	URLs	Triples
Microformats	1,528,354	143,411,821	1,594,758,346
Microdata	1,100,783	312,229,919	13,224,134,881
RDFa	512,806	196,336,975	1,598,114,462
JSON-LD	596,229	35,486,192	382,896,204

Table 1: Web Data Commons Crawling Results [13], November 2015

The vocabulary also depends on the data consumer. Like mentioned above, schema.org should be chosen for search engines. It also has an open development model, they have a github account and there are also extensions to the vocabulary possible that include other vocabularies.

3 Example Applications

The biggest data consumers of semantic annotations from websites are search engines. This is also the data consumer which adds some incentive to data producers. Therefore we present three example applications of semantic annotations that are used by data producers to improve their search result displays in the Google search engine. We present every example with an image accompanied by JSON-LD code to produce the image, we also include the resulting RDF graph in N3 representation. We chose JSON-LD for this because it is the most concise format.

The presented examples are in no way complete. Google supports a lot of different types of information from the schema.org vocabulary. The content types for which the search results render differently can be viewed in the Google search gallery [17].

3.1 Google Search Results - Product

This is how a fully annotated product looks in the Google search results.

Figure 1: Google search: Product display

This is the annotation code in JSON-LD for this example.

```
<script type="application/ld+json">
{
  "@context": "http://schema.org/",
  "@type": "Product",
  "name": "Lava Lite Classic Lava Lamp Purple/Blue",
```

```
  "aggregateRating": {
    "@type": "AggregateRating",
    "ratingValue": "3.5",
    "reviewCount": "60"
  },
  "offers": {
    "@type": "Offer",
    "priceCurrency": "USD",
    "price": "13.17",
    "availability": "http://schema.org/InStock"
  }
}
</script>
```

Listing 7: JSON-LD Product

The RDF graph serialization in N3:

```
@prefix s: <http://schema.org/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

_:b0 s:aggregateRating _:b1 .
_:b0 s:name "Lava Lite Classic Lava Lamp Purple/Blue" .
_:b0 s:offers _:b2 .
_:b0 rdf:type s:Product .
_:b1 s:ratingValue "3.5" .
_:b1 s:reviewCount "60" .
_:b1 rdf:type s:AggregateRating .
_:b2 s:availability "http://schema.org/InStock" .
_:b2 s:price "13.17" .
_:b2 s:priceCurrency "USD" .
_:b2 rdf:type s:Offer .
```

As we can see it is not hard to annotate content this way and a lot of useful information for the user can be displayed before he visits the website. Price and rating are important information for every user who looks to buy a product. It should be noted here that Google does not include typed literals in its example code anywhere nor does it state that it is supported. Although it makes a lot of sense for things like price, we can not be sure if it would work for the Google crawler.

3.2 Google Search Results - Recipe

Again, first we can see in figure 2 how a recipe can look in a search result. This is from a mobile search therefore the card view is shown.

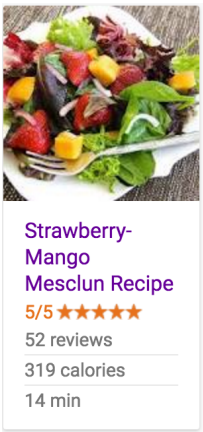


Figure 2: Google Search: Recipe display

This is the JSON-LD code for this recipe.

```
<script type="application/ld+json">
{
  "@context": "http://schema.org/",
  "@type": "Recipe",
  "name": "Strawberry-Mango Mesclun Recipe",
  "image": "http://images.media-allrecipes.com/userphotos/600x600/1116471.
     jpg",
  "aggregateRating": {
    "@type": "AggregateRating",
    "ratingValue": "5",
    "reviewCount": "52"
  },
  "totalTime": "PT14M",
  "nutrition": {
    "@type": "NutritionInformation",
    "calories": "319 cal"
  }
}
</script>
```

Listing 8: JSON-LD Recipe

This JSON-LD serializes into this N3 format.

```
@prefix s: <http://schema.org/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

_:b0 s:aggregateRating _:b1 .
```

```
_:b0 s:name "Strawberry-Mango Mesclun Recipe" .
_:b0 s:nutrition _:b2 .
_:b0 rdf:type s:Recipe .
_:b1 s:ratingValue "3.5" .
_:b1 s:reviewCount "60" .
_:b1 rdf:type s:AggregateRating .
_:b2 s:calories "319 cal" .
_:b2 rdf:type s:Nutrition .
```

This may be essentially all a user may need to decide if he wants to know more about this recipe. The schema.org recipe annotation type supports a lot more information, for example, we can also annotate the ingredients needed and the cooking steps.

3.3 GMail integration Event Reservation

This example application is not from the Google search. It is an application from their E-Mail service GMail. Google GMail allows adding semantic markup to the E-Mail itself which in turn changes the appearance of the E-Mail in the GMail web interface and may allow additional actions. We are going to pick an event reservation as an example here.

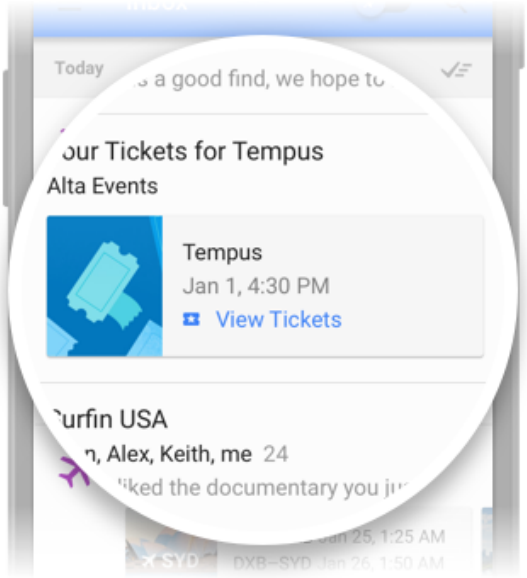


Figure 3: GMail Event Reservation [9]

This is how the JSON-LD markup looks for this.

```
<script type="application/ld+json">
{
  "@context": "http://schema.org",
  "@type": "EventReservation",
  "reservationNumber": "E123456789",
  "reservationStatus": "http://schema.org/Confirmed",
  "underName": {
    "@type": "Person",
    "name": "John Smith"
  },
  "reservationFor": {
    "@type": "Event",
    "name": "Tempus",
    "startDate": "2016-01-01T19:30:00-08:00",
    "location": {
      "@type": "Place",
      "name": "AT&T Park",
      "address": {
        "@type": "PostalAddress",
        "streetAddress": "24 Willie Mays Plaza",
        "addressLocality": "San Francisco",
        "addressRegion": "CA",
        "postalCode": "94107",
        "addressCountry": "US"
      }
    }
  }
}
</script>
```

Listing 9: JSON-LD GMail markup

Although a nice idea, this is a feature that is only supported by GMail. No other E-Mail provider or program interprets semantic markup in E-Mails.

4 Case Study

The idea was to use the data collected from Web Data Commons [13] which extracted semantic information from the content collected by Common Crawl [15]. The Common Crawl contains the crawlable Internet as a whole and is therefore an interesting data source for real world usage data. Of the 1.77 billion pages contained in the crawl data, 541 million contained some kind of semantic annotations. These originate from 2.72 million domains and contain altogether 24.38 billion RDF N-Quads, for reference the DBpedia project consists of about 3 billion RDF triples [16].

The Web Data Commons project has split the data into the different formats, RDFa, JSON-LD, Microdata and different Microformats. The collected RDFa data is 40GB compressed which is 440GB uncompressed. The collected JSON-LD Data is 6.5 GB compressed which is 71.5 uncompressed. These are the formats we include in our case study, Microdata would also be interesting but due to space constraints we can not read the additional 3179 GB uncompressed. Although, as we will see later we only succeeded in reading the JSON-LD dataset completely. The RDFa dataset is only partially read (66 from 397 files).

This data should be fed into a triple store which can then be used to run SPARQL queries on the data to extract some insights. As the Common Crawl [15] collects data from the whole Internet, it would be interesting to use this data and, for example, to look at what kind of semantic annotations come with it.

4.1 System

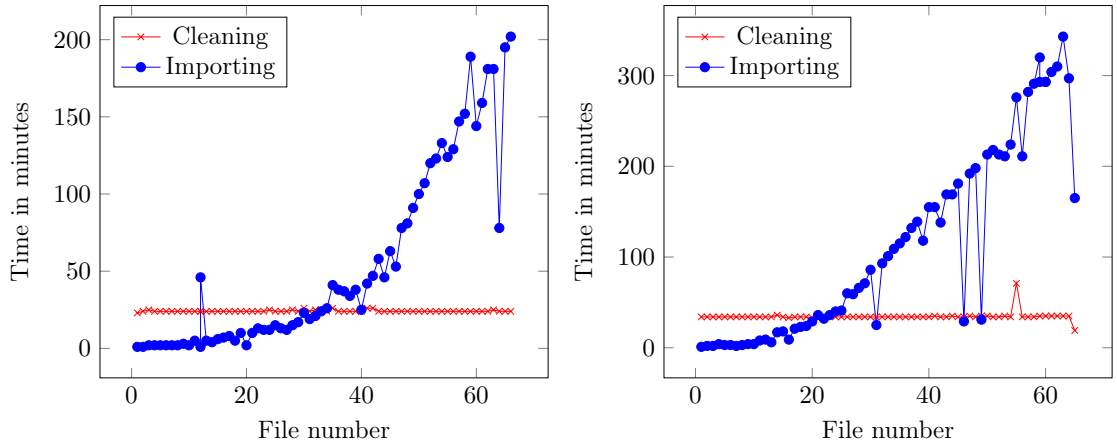
Now we describe briefly the system we implemented to read in all the data. The N-Quads data is first extracted from the compressed file and then read into an in-memory graph. For this Python and RDFLib [22] are used. In this step the data is cleaned and recoverable errors are fixed, e.g. the language tag divider (see Section 4.2 Problems). To minimize lost data each N-Quad is separately injected into the in-memory graph, if this fails due to formatting errors the N-Quad is dropped. Otherwise it is appended to the output N-Quads file.

After the complete input file is cleaned and saved to an output N-Quads file, it is stored into the Apache Jena TDB. For this, the bulk loader is used (directly from the command line), which works directly on the files of the TDB. The bulk loader should be the fastest method of adding data to the Apache Jena TDB but in our case study it slowed down somewhat later as we will explain in the next section.

4.2 Problems

The first and simplest problem was that the format of the data was not really N-Quads as defined by the W3C [7]. The language tag divider between language and locality was wrong. In the W3C standard it is - and in the data it was _. This was easily solved by a regular expression which converted the data on reading into the in-memory graph. A related problem was that the language tags were not valid in some other cases. One example was @fr2 which is not valid according to the W3C because numbers are not allowed in the first part. Some of these problems can be fixed, we can change the language divider to - and throw away empty language tags. But some of these fixes would have to be done by hand, e.g., and match @fr2 to @fr which is not feasible for this data size. Therefore we used our approach of trying first to store each N-Quad into an in-memory graph described in the previous section.

A scale related problem we ran into was the slowing down of the import into the Apache Jena TDB as can be seen in the figure.



(a) RDFa import into Apache Jena TDB *(b) JSON-LD import into Apache Jena TDB*

Figure 4: Time for importing and cleaning data

The JSON-LD import was complete, we imported the whole 65 compressed files provided by Web Data Commons. The size of the storage on disk for the TDB was 120 GB. The size of the store in triples is 355,602,390. The RDFa import was not complete, we only imported 66 compressed files due to time constraints. Although, it is probably not feasible to import the rest of the files, as the RDFa dataset consists of 397 files and the TDB import would probably fail at some point. The drops in the imports are due to the bulk loader failing which, again due to time constraints, we ignored for the time being and just jumped to the next file. The last drop in the JSON-LD import is due to the file containing less data.

4.3 Data RDFa

In this section we present some overview over the RDFa dataset. Some of these values are provided by the Web Data Commons therefore here the complete dataset is used. The incomplete imported RDFa data is used only for the SPARQL queries in section 4.4.

Vocabulary	Domains
http://opengraphprotocol.org/schema/	204,487
http://ogp.me/ns#	106,941
http://www.facebook.com/2008/	75,086
http://rdf.data-vocabulary.org/#	69,664
http://xmlns.com/foaf/0.1/	60,427
http://purl.org/dc/terms/	57,582
http://purl.org/rss/1.0/modules/content/	42,652
http://rdfs.org/sioc/ns#	40,870
http://ogp.me/ns/fb#	40,734
http://www.w3.org/2000/01/rdf-schema#	13,510

Table 2: Semantic vocabularies used, top 10 domains, Source: Web Data Commons [13]

As can be seen in the table, we have a lot of older vocabulary URI's still in use. Some of these do not even exist anymore, e.g., rdf.data-vocabulary.org. Facebook's open graph protocol [21] is included 3 times in different versions in the table. There seems to be significant inertness to changing the semantic annotation vocabulary once it is included into the HTML. That is surprising as HTML code is seldom written by hand anymore but could be indicative of old software in use to generate the HTML.

Class	Domains
http://rdf.data-vocabulary.org/#Breadcrumb	63,021
http://xmlns.com/foaf/0.1/Image	46,984
http://xmlns.com/foaf/0.1/Document	44,609
http://rdfs.org/sioc/ns#Item	28,048
http://www.w3.org/2004/02/skos/core#Concept	10,306
http://rdfs.org/sioc/ns#UserAccount	9,865
http://rdf.data-vocabulary.org/#Review-aggregate	5,278
http://rdf.data-vocabulary.org/#Rating	3,966
http://rdfs.org/sioc/ns#Post	3,654
http://www.w3.org/1999/xhtml	3,008

Table 3: Semantic classes used, top 10 domains, Source: Web Data Commons [13]

Here, we can see a little bit more classes from other vocabularies. The old version of the schema.org vocabulary (rdf.data-vocabulary.org) is 3 times in the table. FOAF, Simple Knowledge Orientation System (SKOS) and Semantically-Interlinked Online Communities (SIOC) are also interesting. These indicate that not only search engines are the expected data consumers for sites annotated with RDFa.

Property	Domains
http://opengraphprotocol.org/schema/title	200,411
http://opengraphprotocol.org/schema/urls	198,880
http://opengraphprotocol.org/schema/site_name	186,085
http://opengraphprotocol.org/schema/image	140,173
http://opengraphprotocol.org/schema/description	110,229
http://ogp.me/ns#title	97,129
http://ogp.me/ns#url	84,741
http://ogp.me/ns#description	79,620
http://ogp.me/ns#image	79,544
http://ogp.me/ns#site_name	75,560

Table 4: Semantic properties used, top 10 domains, Source: Web Data Commons [13]

Table 4 shows only open graph protocol [21] properties. We can assume that RDFa is mostly used to add this kind of annotation properties to websites. An important factor in this may be that when Facebook started the open graph vocabulary it used RDFa from the start. It also does not have that much properties as it is only used to annotate meta information on the page, e.g., name, which picture to include on Facebook if the page is shared.

4.4 Data Queries RDFa

In this section we present the SPARQL queries and results that we used to extract information about real world usage of the example applications we presented in section 3. The query contains the GRAPH component as we read the data into the TDB store in N-Quad format and kept the graph. The graph of the Web Data Commons data refers to the website where the semantic data was extracted from.

We use multiple versions of this query to acquire usage data for our application example for recipes, the first queried only for type Recipe and name, second added the times and the third added the nutrition information.

```
PREFIX s: <http://schema.org/>
PREFIX os: <http://rdf.data-vocabulary.org/#>
SELECT (count(*) as ?rec)
WHERE {
```

```
    GRAPH ?g {
        {
          ?s a s:Recipe.
          ?s s:name ?name.
          ?s s:prepTime ?pt.
          ?s s:totalTime ?tt.
          ?s s:nutrition ?nutr.
          ?nutr a s:NutritionInformation.
          ?nutr s:calories ?cal.
        }
        UNION
        {
          ?s2 a os:Recipe.
          ?s2 os:name ?name2.
          ?s2 os:prepTime ?pt2.
          ?s2 os:totalTime ?tt2.
          ?s2 os:nutrition ?nutr2.
          ?nutr2 a so:NutritionInformation.
          ?nutr2 os:calories ?cal2.
        }
    }
}
```

Listing 10: SPARQL query Recipes

Recipes (schema.org)	72
Recipes (data-vocabulary.org)	4,316
Recipes with times (data-vocabulary.org)	181
Recipes with times (schema.org)	8
Recipes with times and calories (both)	0

Table 5: Recipes, Source: Web Data Commons [13]

As the results show, there are not many recipes to begin with and 0 with times and calories. There are also much more recipes with the old address of the schema.org vocabulary (rdf.data-vocabulary.org).

In the next step we take a look at the annotated data of products, to check if we find some of the annotations used in our example applications.

```
PREFIX s: <http://schema.org/>
PREFIX os: <http://rdf.data-vocabulary.org/#>
```

```
SELECT (count(*) as ?ret)
WHERE {
    GRAPH ?g {
        {
          ?s a s:Product .
          ?s s:name ?name .
          ?s s:offers ?o .
          ?o s:price ?p .
          ?o s:priceCurrency ?cur .
          ?s s:aggregateRating ?r.
          ?r a s:AggregateRating .
          ?r s:ratingValue ?rv .
          ?r s:reviewCount ?rc .
        }
        UNION
        {
          ?s2 a os:Product.
          ?s2 os:name ?name2.
          ?s2 os:offers ?o2.
          ?o2 os:price ?p2.
          ?o2 os:priceCurrency ?cur2.
          ?s2 os:aggregateRating ?r2.
          ?r2 a os:AggregateRating .
          ?r2 os:ratingValue ?rv2 .
          ?r2 os:reviewCount ?rc2 .
        }
    }
}
```

Listing 11: SPARQL query Products

Products (schema.org)	229
Products (data-vocabulary.org)	47,206
Products (both) with price and currency	0
Products (both) with price, currency and rating	0

Table 6: Products, Source: Web Data Commons [13]

In the RDFa dataset are only a couple of products and those are not richly annotated. No products have a rating and no products have a price. Again, we can see that there are more products annotated with the old schema.org version than the new one.

4.5 Data JSON-LD

As the JSON-LD dataset was the smallest we could import it completely. Like in the previous section we will start with a quick overview and then present some SPARQL queries on the complete JSON-LD data.

Vocabulary	Domains
http://schema.org/	597,156
http://www.w3.org/1999/02/22-rdf-syntax-ns#	40
http://purl.org/dc/terms/	2
http://www.w3.org/2001/XMLSchema#	2

Table 7: Semantic vocabularies used, top 10 domains, Source: Web Data Commons [13]

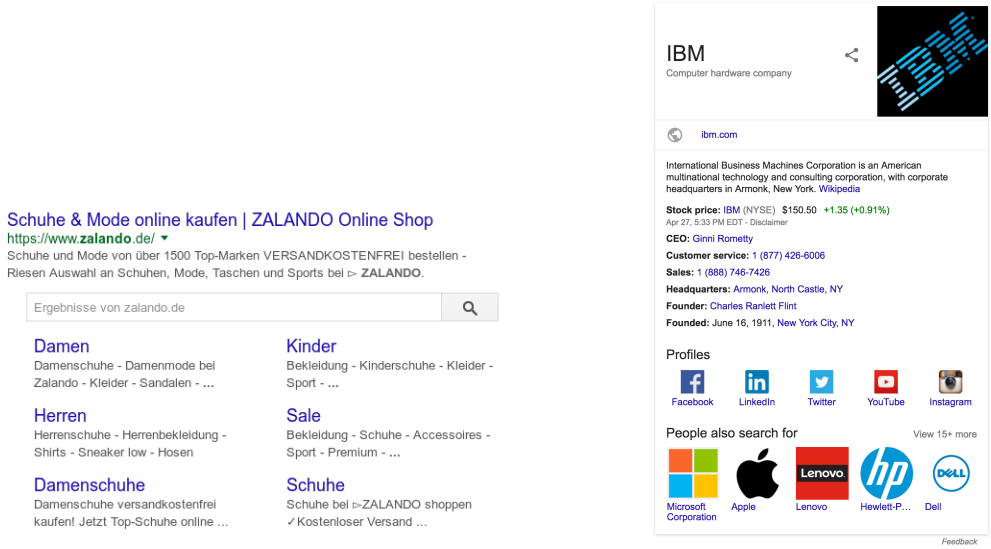
In this table we can see that only schema.org vocabulary is used and only 2 domains used Dublin Core (DC) annotations. XMLSchema and rdf-syntax-ns can be used to annotate typed literals in JSON-LD.

Class	Domains
http://schema.org/WebSite	568,457
http://schema.org/SearchAction	558,321
http://schema.org/Organization	92,316
http://schema.org/LocalBusiness	19,750
http://schema.org/Person	18,231
http://schema.org/Place	2,848
http://schema.org/PostalAddress	2,656
http://schema.org/Event	2,599
http://schema.org/ContactPoint	2,152
http://schema.org/Website	1,202

Table 8: Object classes used, top 10 domains, Source: Web Data Commons [13]

The used classes in JSON-LD are also a strong indicator of its usage primarily for search engines as data consumers.

SearchAction is used to supply meta information about an internal search on the web page, an example for this is shown in figure 5. The rest of the classes can be also be included into the Google Knowledge graph (also figure 5).



(a) Site Search Bar example *(b) Google Knowlege Graph card example*

Figure 5: Google Search result display example

Property	Domains
http://schema.org/url	577,669
http://schema.org/potentialAction	558,356
http://schema.org/target	558,351
http://schema.org/query-input	558,321
http://schema.org/name	486,478
http://schema.org/logo	91,089
http://schema.org/sameAs	66,062
http://schema.org/alternateName	35,435
http://schema.org/openingHours	19,214
http://schema.org/image	6,617

Table 9: Object properties used, top 10 domains, Source: Web Data Commons [13]

The properties used are also indicative of optimizations for search engine results as they are used with the classes described previously.

4.6 Data Queries JSON-LD

In this section we want to check how often the search result improvements we described in section 3 Example Applications appeared in the JSON-LD dataset and how they were annotated, e.g., has every recipe a nutrition property. To count the appearances we use the following SPARQL query.

```
PREFIX s: <http://schema.org/>

SELECT (count(*) as ?ret)
WHERE {
    GRAPH ?g {
        ?r a s:Recipe .
        ?r s:name ?name .

        ?r s:prepTime ?pt .
        ?r s:totalTime ?tt .

        ?r s:nutrition ?nutr .
        ?nutr a s:NutritionInformation .
        ?nutr s:calories ?cal .
    }
}
```

Listing 12: SPARQL query Recipes

To save space only the complete SPARQL query is shown here. We start with only the type check of the recipe and if it has a name set. Then add the prepTime and localTime and after that we query if the nutrition is set and if it contains calories.

Recipes	53,299
Recipes with times	23,189
Recipes with times and calories	49

Table 10: Recipes, Source: Web Data Commons [13]

As can be seen we have a lot of recipes but not a lot of recipes with nutritional information.

We now present the same for our example application for product annotations. Again, we print the complete SPARQL query.

```
PREFIX s: <http://schema.org/>

SELECT (count(*) as ?ret)
WHERE {
    GRAPH ?g {
        ?s a s:Product .
        ?s s:name ?name .
        ?s s:offers ?o .
        ?s s:aggregateRating ?r.
        ?r a s:AggregateRating .
        ?r s:ratingValue ?rv .
        ?r s:reviewCount ?rc .
        ?o s:price ?p .
        ?o s:priceCurrency ?cur .
    }
}
```

Listing 13: SPARQL query Products

Products	98,046
Products with price and currency	65,721
Products with price, currency and rating	24,058

Table 11: Products, Source: Web Data Commons [13]

The products query shows more product annotations are richly annotated with price, currency and a rating.

The examples schema.org (and Google) provides for product markup and the JSON-LD examples we provided in section 3 were without typed literals. This is also noticeable in the JSON-LD results as we will see in the next query.

```
PREFIX s: <http://schema.org/>

SELECT (count(*) as ?num)
WHERE {
    GRAPH ?g {
        ?s s:price ?price .
    }
FILTER(?price != "")
```

```
FILTER(isNumeric(?price))
}
```

Listing 14: SPARQL query Prices

Like before, we only include the complete SPARQL query. We do not require a product here we just ask for a price predicate.

Prices	193,430
Prices non empty	192,858
Prices non empty and numeric type	6,794

Table 12: Prices, Source: Web Data Commons [13]

Table 12 shows that only a small amount of prices in the JSON-LD dataset have a numeric type. With introducing typed literals the JSON-LD code grows a little bit but in the end it is not overly complex as shown in listing 6. It should be added to example code snippets that are provided by schema.org.

5 Summary

In this report we described some of the foundations of the semantic web. We talked about the RDF graph, OWL and SPARQL. We presented the 4 semantic annotation formats that are in use today and provided some details about their syntax and some examples. To give some real world examples of semantic annotation we described 3 example applications for the biggest data consumer the Google search engine. To provide further details on real world usage we present a small case study using the data provided by the Web Data Commons project. In the case study we first gave an overview over the data with some statistical information, e.g. which vocabularies were used the most. Furthermore, we include some SPARQL queries to extract real-world usage scenarios of our presented examples. From the statistical data and the queries we can determine that JSON-LD is almost only used with the schema.org vocabulary. Moreover, it is the dataset mostly tied to search engines as data consumers. The RDFa dataset had some other vocabularies like FOAF or SIOC which are more closely connected to social web semantics.

The lack of typed literals for some properties, e.g., price may be an indicator that most data producers just copy the provided code snippets for their usage scenario. Providers of such code snippets like schema.org or Google should at least optionally include the needed code for typed literals. As an extension to this, for most monetary values the schema.org vocabulary allows text or number, e.g., price on Offer. It should be clear that choosing text here is the wrong approach. Other attributes like baseSalary of JobPosting allow number but also MonetaryAmount or PriceSpecification which allow for ranges and have special attributes for currency. It should be made more clear in the schema.org specifications that using typed literals or more fitting objects such as MonetaryAmount has advantages.

5.1 Future Work

As described in section 4.2 we could not import a lot of the data. The only dataset that we could import entirely was JSON-LD and even there we lost about 30 million triples due to errors with the Apache Jena bulkloader because of encoding errors in the source data. But the main problem there was the increasing time for importing the files. Further work could evaluate which triple store can be used with this data sizes. I unsuccessfully tried serialization into a berekely database and also on on top of a relational database (MySQL) both did slow down very significantly at a very early stage. Virtuoso would be the next in line that should be tried for this. The free version is unlimited, unlike StarDog or Allegro

which limit the number of triples. The W3C has an informal MediaWiki page for large triple stores here [1].

Another possible future work could be to use the Common Crawl data directly. We did not evaluate this avenue but it should be possible and not too much work. The hard work is the extraction of the semantic annotations from HTML. The Web Data Commons project used any23 [14] for extraction from the HTML documents. An evaluation on extraction tools would be interesting, for example, we used the RDFlib [22] in python previously to do the same kind of extraction (this involves a little bit more as it is only a library but it is doable in about 100 lines of python). A comparative evaluation of tools on a limited set of sites would be interesting. Moreover the problems that would be coming up in attempting this comparison, errors because of the source data, etc. are in themselves interesting (classify errors, for example by encoding, wrong type, etc. how many of each class appear).

Another bigger project could be to extract additional historical data (from previous crawls) from the Common Crawl. It would be interesting if and when errors in the encoding of the annotations get fixed. Or maybe when older vocabulary URIs vanish, e.g., rdf.data-vocabulary.org.

[1] https://www.w3.org/wiki/LargeTripleStores

Bibliography

[1] Microformats in rdf, 2012. URL http://semanticweb.org/wiki/Microformats_in_RDF.html.

[2] Owl 2 web ontology language document overview (second edition), 2012. URL https://www.w3.org/TR/owl2-overview/.

[3] Html microdata, 2013. URL https://www.w3.org/TR/microdata/.

[4] Sparql overview, 2013. URL https://www.w3.org/TR/sparql11-overview.

[5] Rdf 1.1 primer, 2014. URL https://www.w3.org/TR/2014/NOTE-rdf11-primer-20140624/.

[6] Json-ld 1.0, 2014. URL https://www.w3.org/TR/json-ld/.

[7] Rdf 1.1 n-quads, 2014. URL https://www.w3.org/TR/n-quads/.

[8] Microdata, 2014. URL https://en.wikipedia.org/wiki/Microdata_(HTML).

[9] Gmail event reservation, 2015. URL https://developers.google.com/gmail/markup/reference/event-reservation.

[10] Rdfa core 1.1 - third edition, 2015. URL https://www.w3.org/TR/rdfa-core/.

[11] Html+rdfa 1.1 - second edition, 2015. URL https://www.w3.org/TR/rdfa-in-html/.

[12] Rdfa lite 1.1 - second edition, 2015. URL https://www.w3.org/TR/rdfa-lite/.

[13] Web data commons november 2015 corpus, 2015. URL http://webdatacommons.org/structureddata/2015-11/stats/how_to_get_the_data.html.

[14] Apache any23, 2016. URL https://any23.apache.org/.

[15] Common crawl, 2016. URL http://commoncrawl.org/.

[16] Dbpedia, 2016. URL http://wiki.dbpedia.org/about/about-dbpedia/facts-figures.

[17] Search gallery, 2016. URL https://developers.google.com/search/docs/guides/search-gallery.

[18] Google introduction to structured data, 2016. URL https://developers.google.com/search/docs/guides/intro-structured-data.

[19] Microformats, 2016. URL http://microformats.org.

[20] Microformats search engines, 2016. URL http://microformats.org/wiki/search_engines.

[21] Open graph protocol, 2016. URL http://ogp.me.

[22] Python rdflib, 2016. URL https://github.com/RDFLib/rdflib.

[23] Schema.org, 2016. URL http://www.schema.org.

[24] D. Crockford. The application/json media type for javascript object notation (json). RFC 4627, RFC Editor, July 2006. URL https://www.ietf.org/rfc/rfc4627.txt.

[25] Jeni Tennison Ivan Herman Ian Hickson, Gregg Kellogg. Microdata to rdf, 2012. URL https://www.w3.org/TR/microdata-rdf/.

[26] M. Suignard M. Duerst. Internationalized resource identifiers (iris). RFC 3987, RFC Editor, January 2005. URL https://tools.ietf.org/html/rfc3987.

[27] Markus Lanthaler Richard Cyganiak, David Wood. Rdf 1.1 concepts and abstract syntax, 2014. URL https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/.

[28] Aaron Swartz. *The Programmable Web:An Unfinished Work*. Morgan and Claypool, 2013. ISBN 9781627051699. doi: 10.2200/S00481ED1V01Y201302WBE005.

[29] L. Masinter T. Berners-Lee, R. Fielding. Uniform resource identifier (uri): Generic syntax. RFC 3986, RFC Editor, January 2005. URL https://tools.ietf.org/html/rfc3986.

[30] Ora Lassila Tim Berners-Lee, James Hendler. Semantic web. *Scientific American*, 2001.