Any Suggestions? Active Schema Support for Structuring Web Information

Silviu Homoceanu, Felix Geilert, Christian Pek, and Wolf-Tilo Balke

IFIS TU Braunschweig, Mühlenpfordstraße 23, 38106 Braunschweig, Germany
{silviu,balke}@ifis.cs.tu-bs.de,
{f.geilert,c.pek}@tu-bs.de

Abstract. Backed up by major Web players schema.org is the latest broad initiative for structuring Web information. Unfortunately, a representative analysis on a corpus of 733 million Web documents shows that, a year after its introduction, only 1.56% of documents featured any schema.org annotations. A probable reason is that providing annotations is quite tiresome, hindering wide-spread adoption. Here even state-of-the-art tools like Google's Structured Data Markup Helper offer only limited support. In this paper we propose SASS, a system for automatically finding high quality schema suggestions for page content, to ease the annotation process. SASS intelligently blends supervised machine learning techniques with simple user feedback. Moreover, additional support features for binding attributes to values even further reduces the necessary effort. We show that SASS is superior to current tools for schema.org annotations.

Keywords: Schema.org, semantic annotation, metadata, structuring unstructured data.

1 Introduction

The Web is a vast source of information and continues to grow at a very fast pace. Unfortunately most of the information is in the form of unstructured text, making it hard to query. Recognizing the importance of structured data for enabling complex queries, the problem of algorithmically structuring information on the Web has been extensively researched, see e.g., [3, 4, 15]. However, current *automatic approaches* still face quality problems and require significant effort for extracting, transforming and loading data. Thus, from a practical perspective they are not yet mature enough to keep up with the volume and velocity, at which new data is published on the Web.

In contrast, the Linked Open Data (LOD) [1, 2] initiative tried a *manual approach*. It offers technology for information providers to directly publish data online in structured form and interlinked with other data. LOD is very flexible since it allows for each data publisher to define its own structure. But this flexibility comes at a price [7]: Although data stores may overlap in terms of the data stored, the vocabulary used for structuring (and thus querying) may seriously differ. Ontology alignment has been proposed as a remedy, but the quality of results is still not convincing [11, 12].

S.S. Bhowmick et al. (Eds.): DASFAA 2014, Part II, LNCS 8422, pp. 251–265, 2014.

To avoid all these problems while improving their query capabilities, major Web search engine providers went a slightly different way. Their *managed approach* builds on a collection of ready-made schemas accessible on schema.org, which are centrally managed by Bing, Google, Yahoo! and Yandex. These schemas are used as a vocabulary to be embedded in the HTML source code of a page using microdata. The main incentive for page owners to use schema.org is that once a Web page features content annotated with schema.org's vocabulary, any search engine can present it as a rich snippet. Furthermore, the Web page has a higher chance of being found by users interested in that very specific content, too. Indeed, motivating page owners to annotate their data with schema.org vocabulary has multiple advantages:

- The effort is spread over many shoulders reducing the effects of volume and velocity at which new data comes to the Web;
- annotations are of high quality – the one creating the data should understand its semantic meaning best;
- the structure is centrally managed and data can be queried globally without complicated alignment operations like in the case of LOD;
- complex queries with Web data are enables, ultimately fostering semantic search for the next generation Web.

But is schema.org being adopted by page owners? An in depth analysis on the acceptance of schema.org reveals, that the number of annotations is in fact very small. The main reason is that the annotation process is quite demanding. Annotators have to repeatedly switch between the page to annotate and schema.org, while browsing through more than 500 schemas with numerous attributes each to find the best matches. Furthermore, adding the actual markup can be tiresome, especially for publishers using "What You See Is What You Get" content management systems.

Sharing the confidence that schema.org will empower complex queries with Web data, we propose *SASS* (Schema.org Annotation Support System), a two stage approach offering support for annotating with schema.org. Analyzing any web page, in the first stage the system finds suggestions for schemas matching page content. For this purpose, simple models are trained with common *machine learning techniques*. But to gain high precision SASS then relies on *user feedback* to validate and fine tune proper schema annotations. The second stage specifically focuses on aiding users in associating schema attributes to values from the page content: *typical attributes* for items of a certain schema have a higher chance of being mentioned in the item data. SASS thus encourages users to consider attributes in order of their typicality. Finally, the system directly generates the HTML code enriched with schema.org annotations. Through the semi-automatic schema matching process and the benefits of considering typical attributes first, our system is superior to solutions like Google's Structured Data Markup Helper (www.google.com/webmasters/markup-helper) or the method recently presented in [13] offering only graphical interface support.

The contribution of this paper can be summarized as follows: we first perform an extensive analysis on the acceptance of semantic annotation technologies with an in-depth focus on schema.org. We then present the design of our annotation support system SASS that relies on lessons learned from the analysis; and finally we present and evaluate machine learning methods for supporting semi-automatic annotations.

2 Web-Scale Analysis on Data Annotation Technologies

Semantic annotation technologies for Web content started to gain importance about a decade ago. Introduced in 2005, microformats are the first important semantic annotation technology. But microformats cover only few entity types annotated through generic HTML "class" attribute. This complicates both the process of annotation and of finding annotations. Attempting to tackle the problems that microformats have, the W3C proposed RDFa as a standard in 2008. It introduces special data annotation attributes for HTML. However it has no centralized vocabulary, leading to heavy fragmentation: Different sources may use different vocabulary to describe the same data. This hinders the process of automatically interpreting or querying data as a whole.

To tackle this problem, search engine providers proposed microdata, a semantic annotation technique relying on few global attributes and a standard vocabulary. The first such vocabulary was "data-vocabulary" proposed by Google in 2009. In mid-2011 Bing, Yahoo and Yandex joined Google's initiative. The "data-vocabulary" was extended to a collection of schemas. Made available on *schema.org*, this collection is evolving continuously to reflect data being published on the Web. To provide for a high level of quality, new schema proposals are reviewed for approval by a standardization committee. An example of a Web page for the movie "Iron Man 3" annotated with schema.org is presented in Figure 1.

a)

```
<div>
    <h1>Iron Man 3</h1>

    <div>
    When Tony Stark's world is torn apart by a formidable terrorist called the
    Mandarin, Stark starts an odyssey of rebuilding and retribution.
    </div>
    <div>
        Director: <span>Shane Black</span>
    </div>
    <div>
        <span>10</span> stars from <span>1337</span> users.
        Reviews: <span>42</span>.
    </div>
</div>
```

b)

```
<div itemscope itemtype="http://schema.org/Movie">
    <h1 itemprop="name">Iron Man 3</h1>

    <div itemprop="description">
        When Tony Stark's world is torn apart by a formidable terrorist called the
        Mandarin, Stark starts an odyssey of rebuilding and retribution.
    </div>

    <div itemtype="http://schema.org/Person" itemscope itemprop="director">
        Director: <span itemprop="name">Shane Black</span>
    </div>

    <div itemtype="http://schema.org/AggregateRating" itemscope itemprop="aggregateRating">
        <span itemprop="ratingValue">10</span> stars from <span itemprop="ratingCount">1337</span> users.
        Reviews: <span itemprop="reviewCount">42</span>.
    </div>
</div>
```

Fig. 1. Web Page section presenting information about movie Iron Man 3. (a) before and b) after annotating it with schema.org.

Currently, there are 529 schemas on schema.org, organized in a 5 level hierarchical structure. Each level introduces a higher degree of specificity. At the root of the hierarchy, there is an all-encompassing schema called "Thing" with 6 general attributes suitable for describing all kinds of entities. Attributes are inherited from parent to child schemas. They may be of basic types e.g., text, boolean, number, etc. or they may in turn represent schemas. For example, for schema Movie, the attribute actor is of type Person which is itself a schema on schema.org. From a structural stance schema.org is similar to DBpedia, the central data repository in LOD. Overall DBpedia comprises 458 schemas. DBpedia maps its structural information to schema.org through the "owl:equivalentClass" attribute. However, the mapping is relatively small: Only 45 links are provided in the current version of DBpedia despite far more semantic similarities easy to spot on manual samples.

To assess the acceptance of schema.org we analyzed ClueWeb12, a publicly available corpus comprising English sites only. The corpus has about 733 million pages, crawled between February and May 2012. It comprises pages of broad interest: The initial seeds for the crawl consisted of 3 million websites with the highest PageRank from a previous Web scale crawl. Our analysis shows that about a year after its introduction, only 1.56% of the websites from ClueWeb12 used schema.org to annotate data. The numbers of pages annotated with mainstream data annotation techniques found in the ClueWeb12 documents are presented in Table 1. The use of different standards reflects the chronology of their adoption: Microformats are the most spread followed by RDFa, microdata and schema.org. It's interesting to notice that while microdata was introduced just a year after RDFa, there is a noticeable difference between their usage rates. The reason for this behavior is that when it was introduced, RDFa was presented as the prime technology for semantic annotation. Many content providers adopted it. Further developments brought by Google in 2009 have been regarded as yet another annotation method. It was only in mid-2011 when microdata became the main annotation technology for the newly proposed schema.org that microdata started gaining momentum. In fact, out of 15 million documents annotated with microdata, 12 million (80%) represent schema.org annotations.

Table 1. Distribution of Annotations in ClueWeb12

Data	Found URLs
Microformats	97,240,541 (12.44%)
RDFa	59,234,836 (7.58%)
Microdata	15,210,614 (1.95)
schema.org	12,166,333 (1.56%)

Out of the 296 schemas available in mid-2012 when ClueWeb12 was crawled, only 244 schemas have been used. To retrieve the state of schema.org at that time we used the Internet Archive (web.archive.org/web/20120519231229/www.schema.org/docs/full.html). The number of annotations per schemas (Table 2) follows a power law distribution with just 10 highest ranking schemas being used for 80% of the annotations and 17 schemas making for already 90% of all annotations. From the low

occurring schemas in the long tail, 127 schemas occur less than 1000 times and 96 schemas occur even less than 100 times.

Schemas on schema.org are quite extensive. They include on average 34 attributes. "Thing" is with 6 attributes the smallest schema while "ExercisePlan" with 71 attributes is the most extensive schema on schema.org. The annotations however are by far not as extensive as the structure allows. On average over all annotations, only 4.7 attributes were used. This accounts for about 10% of the attributes available in the corresponding schemas despite remaining data and existing matching attributes. It seems users are satisfied with just annotating some of the attributes. Most probably, this behavior is driven by the fact that rich snippets can only present a few attributes. In consequence users annotate only those few attributes that they consider should be included in the rich snippet. This way, from a user perspective, both the effort of annotating additional information and the risk that the rich snippet would present a random selection out of a broader number of annotated attributes are minimized.

Table 2. Top-20 Schema.org Annotations on the ClueWeb12 Corpus

Schemas	Occurrences	Average Nr. of Attributes	Percentage (Schema.org)
http://schema.org/Blog	5,536,592	5.56	19.57%
http://schema.org/PostalAddress	3,486,397	3.62	12.32%
http://schema.org/Product	2,983,587	2.28	10.54%
http://schema.org/LocalBusiness	2,720,790	3.29	9.62%
http://schema.org/Person	2,246,303	4.97	7.94%
http://schema.org/MusicRecording	1,580,764	2.77	5.59%
http://schema.org/Offer	1,564,257	1.32	5.53%
http://schema.org/Article	1,127,413	1.04	3.99%
http://schema.org/NewsArticle	823,572	3.81	2.91%
http://schema.org/BlogPosting	767,382	3.32	2.71%
http://schema.org/WebPage	659,964	4.11	2.33%
http://schema.org/Review	470,343	3.20	1.66%
http://schema.org/Organization	407,557	1.35	1.44%
http://schema.org/Event	400,721	2.69	1.42%
http://schema.org/VideoObject	396,993	0.47	1.40%
http://schema.org/Place	380,055	2.50	1.34%
http://schema.org/AggregateRating	342,864	1.66	1.21%
http://schema.org/CreativeWork	232,585	2.30	0.82%
http://schema.org/MusicGroup	223,363	1.15	0.78%
http://schema.org/JobPosting	168,542	4.38	0.60%

3 Learning to Annotate Unstructured Data with Schema.org

The benefits of building a structured Web are obvious and the approach followed by schema.org seems promising. Most of the data annotated with schema.org vocabulary represents e-shopping entities like products, restaurants or hotels. Indeed economic factors may have driven the adoption of schema.org for e-shopping relevant data. For

the rest, the benefit of having pages presented as rich snippets seems rather small when compared to the effort of annotating data.

Unfortunately, as we have seen, schema.org annotations are not yet used broadly. The main reason invoked insistently on technology blogs on the Web is that the actual process of annotating Web data with schema.org is quite demanding, see e.g., http://readwrite.com/2011/06/07/is_schemaorg_really_a_google_land_grab. In particular, the structure is centrally managed by schema.org and not at the liberty of annotators like in the case of RDFa. This means that when annotating pages one has to:

a) repeatedly switch between the Web page to annotate and schema.org,
b) to browse through hundreds of schemas with tens of attributes each trying to find those schemas and attributes that best match the data on the Web page,
c) and finally to write the microdata annotation with corresponding schema.org URL resources into the HTML code of the page.

With such a complicated process it's no wonder that 1.1% of all found annotations are erroneous. Most frequent errors were bad resource identifiers caused by misspelled schemas or attributes or by schemas and attribute names incorrectly referred through synonyms.

We believe that providing support for the annotation process will make using schema.org much more attractive for all kinds of data. For this purpose, we propose SASS, a system to assist page owners in:

1. matching schemas from schema.org to the content of a web page,
2. linking the attributes of the matched schemas to the corresponding values from the page,
3. and automatically generating the updated HTML page to include the schema.org annotations.

In contrast to simple tool s like Google's Structured Data Markup Helper or the system presented in [13], SASS goes beyond a mere user interface and actively analyzes page content, to find and propose the best matching schemas, to the user.

Let us take a closer look at SASS's basic interactive annotation workflow (cf. also Algorithm 1): Once a page has been created and before publishing it on the Web, the owner loads the HTML source file into SASS's Web-based annotation support system. First the system finds matches between schemas and pieces of page content, using models that have been trained with machine learning techniques on data annotated in ClueWeb12. Theoretically, any selection comprising consecutive words from the page content is a possible candidate for the matching. But considering all possible selections of page content is not feasible. Fortunately, the layout expressed through HTML elements says much about how information is semantically connected. With the help of the Document Object Model (DOM) API the HTML page is represented as a logical structure that connects HTML elements to page content in a hierarchical DOM tree node structure. These nodes envelop the pieces of content that are matched to the schemas. Starting from the most fine-granular nodes (nodes are processed in the reversed order of the depth-first search) the content of each node is checked for possible match with all schemas from schema.org. Once a match is found, the user is requested to provide feedback. If the match is accepted by the user, the system goes in the second stage of linking schema attributes to values.

If the proposed match is not correct the user may browse through the set of next best matching schemas. If there is no suitable match, the user can always refuse the match recommendation. In this case, the system proceeds to the next node. The process continues until all nodes have been considered. In Figure 2.a. we present a snapshot of SASS proposing "Movie" as best matching schema for the Web page content framed in red. Other schemas showing weaker but still relevant match to the same content area are also presented on the left hand side (listed in the descending order of matching strength). In some cases, fine-tuning may be necessary to adjust the size of the selection marked by the red square. If the user considers that a proposed schema matches the marked content, but the selection square is either too broad or too small, the size control elements enable moving up and down the DOM tree node structure to adjust selection size. Of course, allowing users to fine-tune the selection region may also lead to conflicting assignments: for instance if the new selection corresponds to a node that has already been matched to another schema. Since the content in each node may be associated to just one schema, in such cases the user is asked to confirm which assignment is valid.

Once a match has been confirmed, the system proceeds to the second phase of linking the attributes of the chosen schema values from the selected page content. For schemas, the annotations found in ClueWeb12 are enough for learning schema models to support the schema-to-page-content matching process. However, our analysis presented in Section 2 shows that just about 10% of the attributes have actually been used for annotations. Data for attribute annotations is thus very sparse and learning

Algorithm 1. Content to schemas matching workflow.

Input: *HTML* – the HTML page content, *SORG* – the set of schemas from schema.org

Output: *R* – *result* set comprising matching relations between nodes and schemas, and corresponding list of attributes and values

```
1: doc ← DOMDocument(HTML)
2: N ← DFS(doc).reverse
3: R ← ∅;
4: foreach n in N do
5:     S ← ∅; s ← null; A ← ∅
6:     if n in R then
7:         continue        // skip n as it was already added probably by the
           user's adjusting the selection for some other node
8:     end if
9:     S ← match(n, SORG)
       // returns all schemas from schema.org matching the content from n
10:    if S ≠ ∅ then
11:      s = USER_FEEDBACK(S)
12:      if s ≠ null then
13:          A ← bind(n, s) // A contains attribute value pairs obtained
             from the attributes to page content associations made by the
             user with drag&drop functionality
14:          R.add(n, s, A)  // if n already exists in R the tuple will be
             updated with the new matching
15:      end if
16:    end if
17: end for
```

algorithms didn't prove as successful as for the schemas matching. The lack of proper attribute annotations shows that this phase would benefit the most from any level of support the system can offer. We assume that attributes showing higher degree of typicality for some item-type have a higher chance of appearing in the respective item-data than other attributes. For the confirmed schema, our system displays the attributes in the descending order of their typicality value and encourages users to consider at least those attributes showing highly typicality for the chosen schema. Those attributes are made available on the right hand side of the screen and can be dragged-and-dropped on the corresponding values if provided in the page content (Figure 2.b.). Start-end selection sliders allow for fine content selection.

With the obtained schema matching and attribute value pairs, the system then enriches the original HTML with the corresponding microdata. The generated HTML source is finally validated with MicrodataJS (github.com/foolip/microdatajs).

Let us now provide a detailed description of all core functionalities of the SASS approach within the workflow described above. In Section 3.1 we present the process of matching schemas to unstructured data - page content enclosed in DOM nodes. In Section 3.2 the process of computing attribute typicality is discussed.

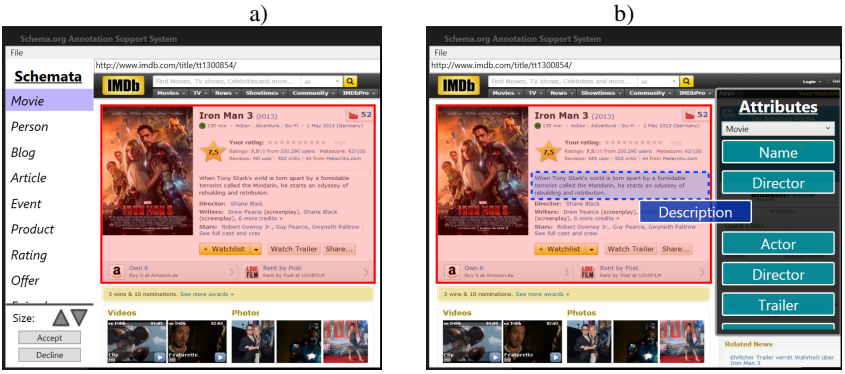

Fig. 2. Schema.org Annotation Support System: a) Matching schema to unstructured data; b) Mapping schema attributes to HTML elements with corresponding values

3.1 Matching Schemas to Unstructured Data

From a sequence of words (the content of a DOM node) and the list of schemas from schema.org, the matching process finds those schemas that "best" match the content. More formally, given W_n={w_1, w_2, …, w_k} the sequence of words representing the content of node n, and S, the set of URIs for schemas from schema.org, the schemas that best match W_n are:

$$S_{W_n} = \{s_i \mid s_i \in S \wedge match(W_n, s_i) \geq \theta\} \quad (1)$$

where θ is a quality regulating parameter (for our experiments θ was set to 0.5), and *match*:{Words × URIs} → [-1,1] is the function for computing the confidence that a certain schema matches the given set of words. The expression of this function

depends on the method that is chosen to perform the matching. There are various such methods. For instance, given that schemas published on schema.org describe various types of entities, one of the first approaches that come to one's mind for binding these schemas to unstructured data is entity recognition and named entity recognition. This has proven to work well for some entity types like products, persons, organizations or diseases [18]. However, considering the popular entities annotated on the ClueWeb12 corpus (Table 2), most of them describe more abstract entities e.g. "Blog", "Review", "Offer", "Article", "BlogPosting", etc. In fact, out of the top-20 entity types, entity recognitions systems like OpenNLP (opennlp.apache.org) or StandfordNER [5] recognize less than half of them. Given an observation W_n, and the annotations extracted from ClueWeb12 as a training set comprising a large number of observations whose category of membership is known (the annotated schema) this becomes a problem of identifying the class for observation W_n. Machine learning methods like Naïve Bayes classification or Support Vector Machines have proven successful for text classification tasks even for more abstract entity types ([8, 9]).

Naïve Bayes classifiers rely on probabilities to estimate the class for a given observation. It compares the "positive" probability that some word sequence is the observation for some schema to the "negative" probability that the same word sequence is an observation for other schemas. In this case the matching function is:

$$match_{Bayes}(W_n, s) = \mathrm{P}(s|W_n) - \mathrm{P}(\bar{s}|W_n) \quad (2)$$

But neither of the two probabilities can be computed directly from the training set. With the help of Bayes's Theorem $\mathrm{P}(s|W_n)$ can be rewritten in computable form as $\mathrm{P}(s|W_n) = \frac{\mathrm{P}(W_n|s)*\mathrm{P}(s)}{\mathrm{P}(W_n)}$. Since W_n is a sequence of words that may get pretty long (W_n=$\{w_1, w_2, \ldots, w_n\}$), and this exact same sequence may occur rarely in the training corpus, to achieve statistically significant data samples "naive" statistical independence between the words of W_n is assumed. The probability of W_n being an observation for schema s becomes: $\mathrm{P}(s|W_n) = \frac{\prod_{j=1} \mathrm{P}(w_j|s)*\mathrm{P}(s)}{\prod_{j=1} \mathrm{P}(w_j)}$, and all elements of this formula can be computed based on the training set: $\mathrm{P}(s)$ can be computed as the relative number of annotations for schema s, $\mathrm{P}(w_j|s)$ the number of annotations for schema s that include w_j relative to the total number of annotations for s, and $P(w_j)$ as the relative number of annotations including w_j. The negative probability $P(\bar{s}|W_n)$ is computed analogously and the matching function on the Bayes classifier can be rewritten as:

$$match_{Bayes}(W_n, s) = \prod_{j=1} \mathrm{P}(w_j|s) * \mathrm{P}(s) - \prod_{j=1} \mathrm{P}(w_j|\bar{s}) * \mathrm{P}(\bar{s}) \quad (3)$$

Being common to all matching involving W_n, $\prod_{j=1} \mathrm{P}(w_j)$ can safely be reduced without negative influence on the result. Probabilities for all words from the training set comprising annotations from ClueWeb12 (excluding stop words) build the statistical language models for all schemas, which are of course efficiently precomputed before performing the actual Web site annotations.

Support Vector Machines use a different approach for classification. For each schema, a training set is built. It comprises annotations of the schema ("positive annotations") and annotations of other schemas ("negative annotations") in equal

proportions. Each training set is represented in a multidimensional space (the Vector Space Model) with terms from all annotations as the space axes and annotations as points in space. In this representation, SVM finds the hyperplane that best separates the positive from the negative annotations for each schema. In the classification process, given observation W_n, and a schema s, SVM represents W_n in the multidimensional term space and determines the side W_n is positioned in with respect to the hyperplane of s. If it's the positive side then there is a match. The normalized distance from W_n to the hyperplane reveals the confidence of the assignment. The closer W_n is to the hyperplane of s, the less reliable the assignment. In this case, the *match* function is:

$$match_{SVM}(W_n, s) = distance(W_n, H_s) \tag{4}$$

3.2 The Typicality of Attributes for a Chosen Schema

Schemas on schema.org on average have 34 attributes. But our analysis on annotated content shows that on average only 4 attributes are actually being annotated. On manual inspection over annotations for multiple schemas we observed that some of the prominent attributes were left un-annotated although there was matching content available. For instance, for movie data, the 'title' was always annotated along with maybe the 'description' or 'director' attributes. But the 'genre' or 'actors' were often left un-annotated. In fact only 38% of the movie annotations also include 'genre' although simply by performing keyword search with a list of genres we found that the information was available in more than 60% of the cases. Clearly, attributes that are typically associated with the concept of movie will most probably also appear in content about movies. Unfortunately many of those attributes had less than a hundred annotations, not enough for building reliable classification models. To support users in providing more extensive annotations we make it easy for them to find those attributes having a high chance to appear in the content. For this purpose we ask users to consider the attributes in the order of their *typicality* w.r.t. to the chosen schema.

Following on the concept of *typicality* from the field of cognitive psychology in [10] we define *attribute typicality* and present a novel and practical rule for actually calculating it. This method doesn't require that attributes themselves be annotated, schema annotations are enough. It's built on top of open information extraction tools and works directly with unstructured data. Starting from content that has been annotated with a certain schema the method is able to compute typicality values for the schema attributes that it finds, even if no annotation is provided for the attributes. To be self-contained, in the following we briefly describe the core of attribute typicality.

The Concept of Typicality. It has been often shown that some instances of a semantic domain are more suitable than others to represent that domain: For instance Jimmy Carter is a better example of an American president than William Henry Harrison. In her quest for defining the psychological concept of typicality, Eleanor Rosch showed empirically that the more similar an item was to all other items in a domain, the more typical the item was for that domain. In fact, the experiments show that typicality strongly correlates (Spearman rhos from 0.84 to 0.95 for six domains) with *family*

resemblance a philosophical idea made popular by Ludwig Wittgenstein in [19]. For family resemblance Wittgenstein postulates that the way in which family members resemble each other is not defined by a (finite set of) specific property(-ies), but through a variety of properties that are shared by some, but not necessarily all members of a family. Based on this insight, Wittgenstein defines a simple family-member similarity measure based on property sharing:

$$S(X_1, X_2) = |X_1 \cap X_2| \tag{5}$$

where X_1 and X_2 are the property sets of two members of the same family. But this simple measure of family resemblance assumes a larger number of common properties to increase the perceived typicality, while larger numbers of distinct properties do not decrease it. In [16] Tversky suggests that typicality increases with the number of shared properties, but to some degree is negatively affected by distinctive properties:

$$S(X_1, X_2) = \frac{|X_1 \cap X_2|}{|X_1 \cap X_2| + \alpha|X_1 - X_2| + \beta|X_2 - X_1|} \tag{6}$$

where α and $\beta \geq 0$ are parameters regulating the negative influence of distinctive properties. For $\alpha = \beta = 1$ this measure becomes the well-known Jaccard coefficient.

Following on the theory introduced by Wittgenstein and extended by Tversky, properties that an entity shares with its family are more typical for the entity than properties that are shared with other entities. Also in the context of Web data, similar entities, in our case items that share the same schema, can be considered to form families. When talking about factual information extracted from the Web we have to restrict the notion of family resemblance based on generic properties (like characteristics, capabilities, etc.) to clear cut attributes. Attributes are in this case given by predicates extracted from (subject, predicate, object) triple-relations extracted from text. Given some family F consisting of n entities $E_1, \ldots, E_n$ all being annotated with the same schema, let's further assume a total of k distinct attributes given by predicates $p_1, \ldots, p_k$ are observed for family F in the corresponding entities' textual annotations. Let X_i and X_j represent the attribute sets for two members E_i and E_j, then:

$$|X_i \cap X_j| = 1_{X_i \cap X_j}(p_1) + 1_{X_i \cap X_j}(p_2) + \cdots + 1_{X_i \cap X_j}(p_k) \tag{7}$$

where $1_X(p) = \begin{cases} 1 \; if \; p \in X \\ 0 \; if \; p \notin X \end{cases}$ is a simple indicator function.

Now we can rewrite Tversky's similarity measure to make all attributes explicit:

$$S(X_i, X_j) = \frac{\sum_{l=1}^{k} 1_{X_i \cap X_j}(p_l)}{|X_i \cap X_j| + \alpha|X_i - X_j| + \beta|X_j - X_i|} \tag{8}$$

According to Tversky, each attribute shared by X_i and X_j contributes evenly to the similarity score between X_i and X_j. This allows us to calculate the *contribution score* of each attribute of any member of the family (in our case schema) to the similarity of each pair of members: Let p be an attribute of a member from F. The *contribution score* of p to the similarity of any two attribute sets X_i and X_j, denoted $C_{X_i,X_j}(p)$, is:

$$C_{X_i,X_j}(p) = \frac{1_{X_i \cap X_j}(p)}{|X_i \cap X_j| + \alpha|X_i - X_j| + \beta|X_j - X_i|} \tag{9}$$

Attribute Typicality. Let F be a set of n entities $E_1, \ldots, E_n$ having the same schema type, represented by their respective attribute sets $X_1, \ldots, X_n$. Let U be the set of all

distinct attributes of all entities from F. The typicality $T_F(p)$ of an attribute $p \in U$ w.r.t. F is the average contribution of p to the pairwise similarity of all entities in F:

$$T_F(p) = \frac{1}{C_2^n} \cdot \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} C_{X_i X_j}(p) \tag{10}$$

where C_2^n represents the number of possible combinations of entities from F.

Typicality values change slowly over time and are influenced only by significant data evolution. For the purpose of this application, the typicality values for each attribute of each schema can be computed as an offline process, on a monthly basis.

4 Evaluation

The approach introduced in this paper requires two stages for performing a full annotation: the first is for matching a piece of content to a schema and the second stage is for associating attributes proposed by the system in the descending order of their typicality, to page content. The main merits of the system are to suggest best matching schemas for the first, and to compute attribute typicality for the second stage.

For evaluating the schema matching functionality we prepared two data sets. Each has about 60,000 annotated web pages randomly harvested from ClueWeb12 comprising annotations with about 110 different schemas each. One of them is used as a training corpus for the classification methods. The other one is used as a test set. The test set is stripped of all annotations and provided to our system. We disabled user feedback at this stage for evaluation purposes. Instead, each match proposed by the system is accepted as correct. We compare the pages annotated by the system for both Naïve Bayes and SVM, to the pages from the original test set and measure the schema matching effectiveness in terms of precision and recall.

On inspection over the results, about 5% of the schemas were not detected at all. The reason for this behavior is the fact that these schemas are present in the test set but they have no or almost no occurrences (up to 10) in the training set. Increasing the size of the training set helps reducing the number of undetected schemas. In fact, initial experiments with 10,000 and 30,000 web pages as training sets, with smaller schema annotation coverage, showed higher numbers of undetected schemas.

Overall, on the 110 schema annotations the system achieves on average 0.59 precision and 0.51 recall for Naïve Bayes and 0.74 precision and 0.76 recall for SVM, Simply matching schemas at random, as a comparison method, results in precision and recall lower than 0.01. The result values vary strongly from schema to schema. For brevity reasons, in Table 3 we show the results for 15 schemas. The system is counting on user feedback to even out the so called "false alarms" emphasized by precision. But the "false dismissals" emphasized by recall are much harder to even out by the user. The system proposes a list of alternatives (formula 1), but if the matching schema is missing from this list, the corresponding annotation will most probably fail. For this reason, the 15 schemas presented in Table 3 are chosen to cover the whole spectrum of F_2-measure values, given that the F_2-measure weights recall twice as much as precision.

No correlation between the number of occurrences in the training set and results could be observed. Having hundreds of schema annotations seems to lead to results similar to having tens of thousands of annotations. A few schemas, especially in the case of Naïve Bayes, have catastrophic precision and recall values (less than 0.01), despite occurring more than 4,000 times in the training set. These are schemas with very broad meaning e.g. "WebPage" or "Thing". Overall, SVM does better than Naïve Bayes. But it is interesting to notice that for many schemas the two approaches seem to complement each other: schemas where the Bayes achieves bad results are handled much better by SVM and vice versa. This finding encourages us to believe that approaches relying boosting meta-algorithms like the well know AdaBoost [6] will provide even better results.

For the second stage, correctness of the typicality value of an attribute can only be assessed through broad user studies. Our experiments, presented in detail in [10] show that with the attribute typicality method, the top-10 most typical attributes are selected with average precision and recall values of 0.78 and 0.6 respectively. The lower recall value is explained by the fact that attributes are extracted from text, and don't always exist in the corresponding schema.org schemas.

Table 3. Precision and Recall values for the matching of schemas with Bayes and SVM

	Naïve Bayes				SVM					
Nr.	Schemata	Prec.	Rec.	#	F_2	F_2	Prec.	Rec.	#	Schemata	Nr.
1	JobPosting	0.80	0.91	4,129	0.89	0.95	0.94	0.95	10,622	Blog	10
2	Event	0.63	0.78	15,856	0.74	0.85	0.79	0.87	4,129	JobPosting	1
3	LocalBusiness	0.86	0.71	65,691	0.74	0.82	0.83	0.82	126,220	Product	9
4	Offer	0.49	0.82	73,907	0.72	0.81	0.72	0.84	65,691	LocalBusiness	3
5	MusicRecording	0.42	0.77	12,848	0.66	0.77	0.69	0.79	34	EducationEvent	14
6	Movie	0.38	0.38	3,407	0.38	0.68	0.63	0.69	10,395	Place	8
7	Review	0.53	0.32	4,551	0.35	0.68	0.59	0.70	4,551	Review	7
8	Place	0.47	0.30	10,395	0.32	0.58	0.54	0.60	1,153	MobileApplication	13
9	Product	0.92	0.26	126,220	0.30	0.57	0.52	0.59	14,398	Article	12
10	Blog	0.11	0.16	10,622	0.15	0.46	0.44	0.46	3,407	Movie	6
11	Organization	0.52	0.10	10,558	0.12	0.46	0.41	0.47	12,848	MusicRecording	5
12	Article	0.06	0.05	14,398	0.05	0.45	0.43	0.46	4,725	WebPage	15
13	MobileApplicatior	0.01	0.04	1,153	0.02	0.41	0.41	0.41	15,856	Event	2
14	EducationEvent	0.00	0.25	34	0.01	0.39	0.38	0.40	10,558	Organization	11
15	WebPage	0.00	0.00	4,725	0.00	0.16	0.17	0.16	73,907	Offer	4

On average the system matches schemas correctly even without user feedback in 2 out of 3 cases. But the overall quality of the results doesn't encourage us to believe in the feasibility of a fully automatic annotation system. User input is especially important for the attribute annotations. The attributes being presented first, show high typicality values and have a higher chance of appearing in the content to be annotated. This reduces the effort needed for broader annotations and has the benefit of controlling that at least the important attributes are included in the annotation.

5 Related Work

Acknowledging the difficulties users have when annotating data with schema.org, a variety of tools have recently been proposed (schema.rdfs.org/tools.html). Schema-Creator (schema-creator.org) and microDATAGenerator (microdatagenerator.com) are two important solutions for form based interface-focused tools for schema.org annotations. Google's Structured Data Markup Helper offers more elaborate GUI support for more comfortable manual content annotation. An in-depth analysis regarding the visualization of un-structured semantic content along with a formal description of visualization concepts is presented in [13]. Building on these concepts the authors propose and evaluate a graphical user interface on two application scenarios for annotating data with schema.org. In contrast, our system goes beyond a simple GUI and makes high quality suggestions for the annotations.

In [17] the authors present MaDaME, a system that infers mappings between content highlighted by the user and schemas from schema.org. For this purpose the system relies on WordNet. But the system is only appropriate for annotating named entities and nouns known to WordNet. Everything else is not supported. Atomic content like nouns or names also have to be highlighted by the user for the annotation process as the system doesn't process pages as a whole. In [14], a tool for adding schema.org types automatically is presented. It relies on domain knowledge and NER to extract key terms and to generate structured microdata markup. It requires a high quality knowledge base with metadata represented in RDF for each schema to be annotated and has been show to work only on patent data. This will not scale for all schemas. Furthermore, NER alone cannot deal with all types of schemas from schema.org.

6 Conclusions and Future Work

The Web is an abundant source of information – however, mostly in unstructured form. Querying such data is difficult and the quality of results obtained by keyword-based approaches is far behind the quality offered by querying structured data. Sustained by major Web players and relying on a controlled set of schemas, schema.org has the potential to change this situation once and for all. Unfortunately, annotating data with schema.org still is a tiresome process, hindering its wide-spread adoption. Since state-of-the-art tools like Google's Markup Helper offer only limited benefits for annotators, we inspected the feasibility of providing better annotation support.

Driven by insights into schema.org annotations obtained by analyzing a large corpus of 733 million web documents, we derived a set of desirable design goals. A successful support has to always maintain high annotation suggestion quality, while using supervised machine learning techniques to cater for the highest possible amount of automation. Our innovative SASS approach shows that given the right blend of techniques integrated in an intelligent workflow, existing annotations can indeed be effectively used for training high quality models for schema matching. Relying on the concept of attribute typicality SASS first offers those attributes having higher chance of appearing in the page content. As our evaluations show this indeed essentially

reduces the effort of searching through attribute lists. Currently, these features make our system superior to any other tool offering support for schema.org annotations.

Schema matching approaches have different strengths. In future work, we plan to evaluate the performance of boosting meta-algorithms employing multiple matching techniques. User feedback generated through the system usage can also be used to improve the quality of the suggestions, a subject we leave to future work.

References

1. Berners-Lee, T.: Linked Data. Design issues for the World Wide Web Consortium (2006), http://www.w3.org/DesignIssues/LinkedData.html
2. Bizer, C., et al.: Linked Data - The Story So Far. Int. J. Semant. Web Inf. Syst. (2009)
3. Cafarella, M.J., et al.: WebTables: Exploring the Power of Tables on the Web. PVLDB (2008)
4. Cafarella, M.J., Etzioni, O.: Navigating Extracted Data with Schema Discovery. Proc. of the 10th Int. Workshop on Web and Databases, WebDB (2007)
5. Finkel, J.R., et al.: Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In: Proc. of Annual Meeting of the Assoc. for Comp. Linguistics, ACL (2005)
6. Freund, Y., Schapire, R.E.: A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. J. Comput. Syst. Sci. 55, 1 (1997)
7. Homoceanu, S., Wille, P., Balke, W.-T.: ProSWIP: Property-based Data Access for Semantic Web Interactive Programming. In: Alani, H., et al. (eds.) ISWC 2013, Part I. LNCS, vol. 8218, pp. 184–199. Springer, Heidelberg (2013)
8. Homoceanu, S., et al.: Review Driven Customer Segmentation for Improved E-Shopping Experience. In: Int. Conf. on Web Science, WebSci (2011)
9. Homoceanu, S., et al.: Will I Like It? Providing Product Overviews Based on Opinion Excerpts. IEEE (2011)
10. Homoceanu, S., Balke, W.-T.: A Chip Off the Old Block – Extracting Typical Attributes for Entities based on Family Resemblance (2013) (Under submission), http://www.ifis.cs.tu-bs.de/node/2859
11. Jain, P., et al.: Contextual ontology alignment of LOD with an upper ontology: A case study with proton. The Semantic Web: Research and Applications (2011)
12. Jain, P., et al.: Ontology Alignment for Linked Open Data. Information. Retrieval. Boston (2010)
13. Khalili, A., Auer, S.: WYSIWYM – Integrated Visualization, Exploration and Authoring of Un-structured and Semantic Content. In: WISE (2013)
14. Norbaitiah, A., Lukose, D.: Enriching Webpages with Semantic Information. In: Proc. Dublin Core and Metadata Applications (2012)
15. Suchanek, F.M., Weikum, G.: YAGO: A Core of Semantic Knowledge Unifying WordNet and Wikipedia. In: WWW (2007)
16. Tversky, A.: Features of similarity. Psychol. Rev. 84, 4 (1977)
17. Veres, C., Elseth, E.: Schema. org for the Semantic Web with MaDaME. In: Proc. of I-SEMANTICS (2013)
18. Whitelaw, C., Kehlenbeck, A., Petrovic, N., Ungar, L.: Web-scale named entity recognition. In: CIKM (2008)
19. Wittgenstein, L.: Philosophical investigations. The MacMillan Company, New York (1953)