Semi-automatic Metadata Annotation of Web of Things with Knowledge Base

Yunong Yang[1], Zhenyu Wu[2], Xinning Zhu[3]

Beijing University of Posts and Telecommunications, Beijing 100876, China
yangyunong, shower0512, zhuxn@bupt.edu.cn

Abstract: Integrating semantic technology to the Web of Things (WoT) facilitates the creation of a networked knowledge infrastructure with more interoperable data from both physical and cyber world. However, most of current solutions relies on manual methods based on domain-specific ontology, which is only suitable for domain expert and developers but not compatible with large-scale knowledge construction. This paper proposes a semi-automatic annotation framework for the metadata representation of WoT resource. This framework is based on a probabilistic graphical model to collectively infer entities, classes and relations from schematic WoT resources mapping to domain independent knowledge base. A proof-of-concept implementation and performance evaluation are illustrated to show a feasibility of our method.

Keywords: Web of Things; Structural data; Knowledge base; Factor graph; Iterative inference

1 Introduction

Web of Things [1] aims at reusing Web patterns and protocols to make networked physical objects first-class citizens of the World Wide Web. Application developers and open platform vendors could define their own data model, resource representations and web APIs to mashup with other services. Obviously, the openness lowers the barriers of developing IoT application, however, it also results in heterogeneity and isolation of data from different domain-specific data sources. Accordingly, bridging semantic technology to the Web of Things facilitates the creation of a networked knowledge infrastructure [2] with more interoperable data from both physical and cyber world. Hence, the semantic WoT applications allow user to get the high-level details of the sensory observation and infer and query additional knowledge to gain more contextual and intuitive NL-based (Natural Language) interactions, based on semantic search and AI-based QA technologies. According to the vision of the semantic Web of Things, annotation, processing and reasoning thing data on a large-scale will be a challenging task for applications that publish and utilize these data from various sources.

There are several efforts to integrate semantic web with web-based Internet of Things (IoT) platform including SENSEI, Semantic Sensor Web [3], SPITFIRE [4]. By annotating sensor-related features such as the network, deployment, data formats, etc., with machine-understandable vocabulary, i.e., Semantic Sensor Network (SSN) ontology [5], it becomes possible to automate further tasks, e.g., deployment, maintenance, and integration. However, these efforts still have limitations on supporting large-scale applications: things are modeled using domain-specific vocabularies without considering general approach compatible with growing body of semantic global Linked Open Data (LOD) such as DBpedia, Yago, Freebase, GeoNames, FOAF and etc.; the current frameworks rely on domain expert to manually add semantic ontology to the sensor metadata, which is not supporting automatically annotation on raw metadata of things.

Since current thing representation could be freely defined by AVP (Attribute-Value-Pairs) structural schema, such as XML, JSON and HTML, and the content of attribute and value are filled with free-text, information retrieval (IR) framework [6] [7] [8] for structured data based on NLP techniques could be used to automatically interpret the WoT metadata and map the AVP data to entities, classes and relations in given knowledge base (KB). Thus, we propose a practical and effective semi-automatic annotation framework, which modeling the WoT metadata as a graphic model and disambiguating with joint inference method, to achieve the entity assignments task. The main contributions of this framework are summarized as follows:

• We propose a probabilistic graphic model (PGM) to link heterogeneous domain-specific WoT metadata of different schema to the domain independent KB, such as DBpedia, including entities, types and classes;

• We propose an Iterative Message Passing algorithm (IMP) to jointly optimize the identification of the mapping entities for the WoT metadata.

• We implement a tools to semi-automatically transfer WoT metadata to linked data and publish as linked WoT pages on the Web;

• We setup experiment and evaluate our framework to prove the feasibility and performance of our method with quantitative results.

The remainder of this paper is structured as follows. In section 2, we analyze the problem and define research goal. Section 3 mainly presents the system model, along with the semi-automatic annotation method based on PGM inference model. In section 4, we propose a reference implementation and setup experiment to testify

the feasibility and performance of our framework. Finally, we conclude this paper in section 5.

2 Problem Statement and Research Goal

Our goal is to encode WoT metadata including schematic keys and corresponding values as RDF linked data, specifically, mapping the keys of schema to appropriate classes or types, while mapping corresponding values to entities and identify relation between keys, according to given KB. The specific goals of constructing such system are listed below:

1) Associate one or more type with each key of each schema. If a key is deemed not to have any type in target KB, determine that as well

2) Annotate pairs of keys with a binary relation in the given KB. If two keys are not involved in any binary relation in our KB, determine that as well.

3) Annotate all the values corresponding to the key with the entities

4) Generate linked WoT data, store in KB and publish each of the annotated WoT resources with key-value pair representations under different schema as searchable linked WoT pages.

Since the content of both key and value of a schema with instances are all text mentions generated by natural language, these annotation tasks are challenging with

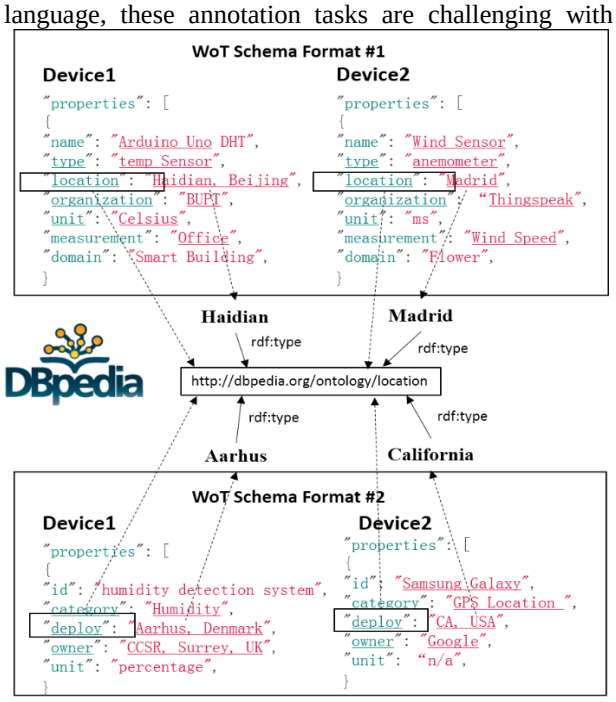

Figure 1 Linked WoT data with semantic annotation of schematic keys, values and relation between keys between different schema

potential ambiguity problem. In particular, there is no direct way to know that two keys or values in two different schema format refer to the same type or entity. When annotating entity mentions in free-form text, the textual context provides clues for disambiguation. In contrast, WoT representation following specific schema structure have negligible amounts of additional contextual. Finally, there may be no intrinsic clue in a WoT representation as to how types and entities therein are related.

Figure 1 shows a typical scenario. Note that the schematic key "location" of WoT schema #1 can easily refer to dbpedia-owl:location or dbpedia-owl:place by using keyword-based search tools since the mention has high similarities of spelling and meaning with the class in DBpedia. While it is difficult to infer the schematic key "deploy" of WoT schema #2 as concept of location, despite the corresponding value "Aarhus, Denmark" and "CA, USA" imply the type of the key means location. According to the hint, it is possible to explain the type of the key and also disambiguate the value's corresponding entities in KB based on collective contextual signals. Thus based on the collective inference, the schematic keys and corresponding values with relations could be linked together via linking to the concept, type and entities in the given KB.

3 System Model Description

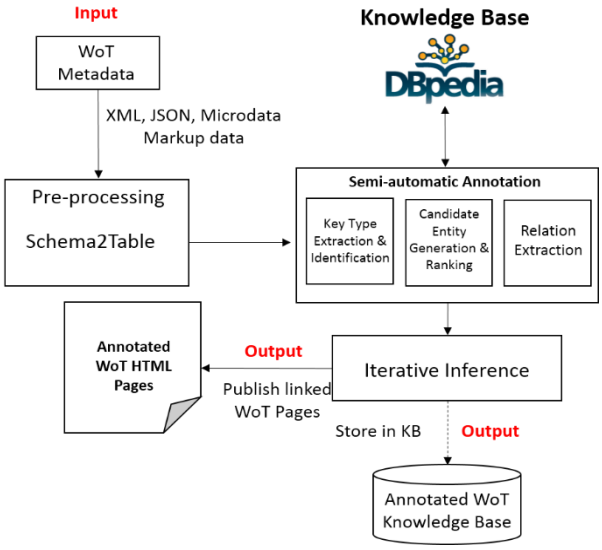

Figure 2 Annotation Framework on WoT metadata based on entity ranking and IMP inference model

Figure 2 shows the annotation framework on WoT metadata based on entity ranking and IMP inference model. It translates and represent WoT metadata with specific schema as linked data according to the domain independent KB. In our current realized system, DBpedia is utilized as the background KB entries and other KBs, such as Yago, Freebase and etc., are also supported.

The input is a group of WoT metadata instances with the same schema template with sterilization formats, i.e. XML, JSON and Microdata with HTML, and the input raw data will be translated as table format with rows and columns for better representing the structure of the data and processing of the algorithm. The keys of the schema are mapped to the columns and the corresponding values of each WoT metadata instance are mapped to the row at correspondent key's columns. Furthermore, the

numerical values are cleaned out from the raw data, since it is not our concerns to extracted semantics from the raw numerical data in our framework.

A formatted tabular data is then processed by the ***Candidate Generation*** module which quires the DBpedia to generate candidate ranked assignments for cell rows, column types and relations between columns. Once candidates are generated, the ***Iterative Inference*** module uses a PGM to infer and disambiguate candidate assignment for entities, types and relations for column, cell rows and column relations. After the linking and annotation process, the linked WoT data are produced, stored in the backend KB and published as annotated linked WoT pages for each WoT resources.

3.1 Candidate Generation

The candidate generation contains Query and Re-rank module which generates a set of candidate assignments for column types, cell texts and relations between columns by using given KB.

3.1.1 Query Module

The query module generates an initial ranked list of candidate assignments for the cell values from DBpedia. DBpedia could be accessed via its open endpoint through its URL [9]. The sparql query of generating candidate entities for cell text is formulated as Figure 3.

The cell text is used as query string, and the return is limited with predefined conditions. The candidate entity should be a "resource" category and the query string should fuzzily match the content of rdfs:label. According to the rule, an initial ranked list of entities for each sparql query statement are generated, along with the entity popularity score.

```
Input: sparql statement
select DISTINCT ?s1
    ( sql:rnk_scale ( <LONG::IRI_RANK> ( ?s1 ) ) ) as ?rank
        where
          {
              ?s1 rdfs:label ?o1 .
              ?o1 bif:contains "( QueryString)" .( e.g.    Humidity )
              Filter regex (str(?s1),"resource").
        }
Output: All matching instance from KB
(e.g. Humidity ,Absolute_Humidity,Relative_Humidity...)
```

Figure 3 sparql query statement to DBpedia endpoint for entity generation

3.1.2 Re-rank Module

We expect the ground truth ranking as high as possible, but what we actually get from DBpedia endpoint is a disordered candidates list which does not fully meet our requirement. For instance, when we input cell text "USA" into query module as query string, what we actually get from the top of the return list are "Democratic_Party_(United_states)","Republican_Party _(United_States)", etc. The target entity "United_States" ranks out of the top 50 of initial ranked list.

To raise the target entity's ranking, we train a SVM ranking classifier [10] that scores how likely a candidate entity is to be a correct assignment for a given query string. And we use this prepared model to re-rank initial candidate entities list. The SVM ranking classifier is trained on a set of string similarity and popularity metrics as its features, which we present as follows.

Entity Popularity:

Entity popularity is a kind of prior probability feature. It helps in cases where disambiguation is hard due to existence of entities having the similar name. It has already been integrated into the DBpedia so that we can access it from query module directly. The popularity score of an entity is based on how frequently they are referenced by other entities. Entity popularity score is increased as a function of the score of the referencing entity, that is, the higher popularity score an entity get, the more reference to this entity.

String Similarity:

In contrast to popularity, string similarity provides a syntactic comparison between cell text and candidate entities. Many candidates do not fully match cell text, so we select several common features such as Levenshtein Distance, Dice Score and string length to measure string similarity between this pair. Several 0/1 check metric are developed to measure whether entity fully contains the words in the cell text, or whether entity equals to cell text. We also check whether all words in the cell text are found in the same order in the candidate entity.

3.1.3 Type and Relation Generation

Entities' type and relation are used to collectively constrain entity assignment in a cell text. We set another kind of query to DBpedia endpoint for seeking all types that a candidate belongs to. The query is formed by using the candidate as subject and "rdf:type" as predicate. The object is what we want. The last kind of query that we send to DBpedia endpoint aims at finding out relations between two keys in metadata. The query uses one key as subject and the other key as object to search predicate that represent relation.

3.2 Iterative Inference

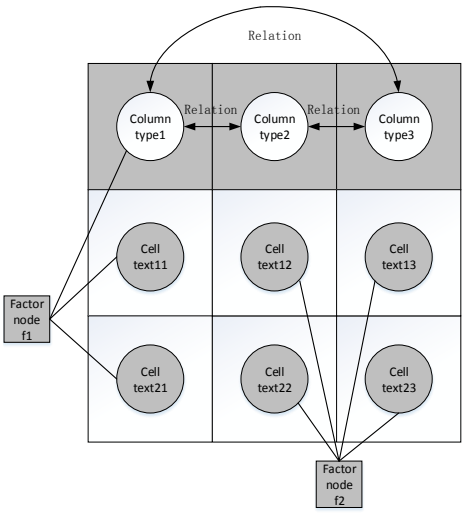

Figure 4 factor graph of a given table

We propose the Iterative Message Passing algorithm to

collectively disambiguate all assignments in a given WoT metadata template. To clearly describe IMP, we first represent the given table as a factor graph which is kind of PGM that shown in Figure 4. In this factor graph, each solid circle is called cell node and each hollow circle is called column node. Both of them belong to what are known as 'variable nodes'. And each square node belongs to what is known as 'factor node'. All variable nodes in one column can be linked to factor node f1 and all cell nodes in different two columns can be linked to factor node f2. Factor nodes could provide agreement functions on selecting a best assignment to each variable node according to the joint constraints.

IMP is an iterative inference method which reassigns each currently assigned candidate of a variable node to its maximum-likelihood value. IMP evaluate the likelihood according to the constraint of column type and relation which is relevant to a variable node. In each iteration, we compute the maximum-likelihood value for each variable node by using the factor nodes they are linked to. Algorithm 1 shows how IMP performs iterative inference over the factor graphical model to find a high-likelihood set of referent entities for all assignments. The method initializes each currently assigned candidate with an entity ranking at the top of the re-rank list derive from re-rank module (line 2-4), and then iteratively re-computes constraints between column type, relation and entity (line 5-12), reassign candidate entity until there is no change in assignment to any variable node or the maximum iteration limit is reached (line 13-15).

Algorithm 1 : Iterative Message Passing

```
1: Vars ← all variable nodes in graphical model
Factors ← all factor nodes in graphical model
2: for all v in Vars do
3:    e ← currently assigned candidate of v
4: end for
5: for all f in Factors do
6:     v' ← all v in Vars which has a link with f .
7:     if   f==f1   then
8:          columnConstraint(v');
9:     else if   f==f2   then
10:         relationConstraint(v');
11:    end if
12: end for
13: for all v in Vars do
14:     reassignCandidate();
15: end for
16: Repeat till all e of Vars no longer change.
```

The constraints derive from both column and relation. Both constraint functions are defined as factor nodes in our factor model. As the principle of both constraints is iterative feedback and re-assign, we only present Algorithm2 which describe Column constraint factor as an example. Algorithm2 shows how column constraint factor works, the factor first elect a most common type as the column type annotation through majority voting process (line 2-13). The query function in line 4 is available in Section 3.1.3. If more than one types get the same vote score, the function will get their granularity which is computed by counting the number of instances belong to the type. The more specific a type is, the fewer instance it gets. IMP select the type with maximum vote score and minimum granularity as the column annotation, which means all assignment in this column should have this type. Then, the column constraint is used as feedback. The factor node will send change message along with target type to those variable nodes, whose currently assigned candidate's type is not comply with column annotation.

Algorithm 2 : Column Constraint function

```
function columnConstraint(VariableNodeSet   v')
1: E ← all e of v'  ※  e is generated in Algorithm1
2: Map ← Create a key-value mapping, key represent
   type and value represent vote
3: for all e in E do
4:    T ← queryColumnTypes(e);
5:    for all t in T do
6:        if   Map.contains(t)   then
7:            t.value += 1
8:        else
9:            Map.put(t , 1)
10:       end if
11:   end for
12: end for
13: topVote ← max votes in all t.value
14: T_high ← all t where t.value == topVote
15: for all t in T_high do
16:    t.granularity ← searchGranularity(t);
17: end for
18: top_col ← t with maximum t.value and minimum
    t.granularity
19: topScore ← top_col.value / number of rows
20: if   topScore <= threshold   then
21:   send low-confidence and unchanged messages
      to all variable nodes in this column.
22:    column annotation ← "no-annotation"
23: else
24:    column annotation ← top_col
25: end if
26: for all e in E do
27:    if   e.T.contains(top_col)   then
28:       send change message along with top_col as
          the type for entity that e should update to
29:    else
30:       send the unchanged message.
31:    end if
32: end for
end function
```

Relation constraint process is similar to the Column constraint. Comparing to Column constraint, relation constraint generates candidate relation with the current assignment of cell nodes in the same row of two columns at first, and send message to both cell nodes at last. The query is available in Section 3.1.3. Relation can be established in both directions. When a relation annotation between two columns is elected, IMP should take both directions into account.

4 Implementation and Evaluation

4.1 Proof of Concept Applications

Prototype Setup

We use DBpedia as our background KB and get candidate entities by querying DBpedia open endpoint [9]. We then choose an open source SVM ranking classifier to re-rank the initial rank list from DBpeida and develop an online semi-automatic WoT device description tools by using Ruby on Rails framework. Finally, the annotation result is persisted as RDF format in openlink Virtuoso.

Typical Use Case

User can use a predefined metadata template or create a new user-defined metadata template to describe their device. For instance, a user defines a set of keys in template, such as "sensor_type", "unit", "organization", "location" and "observation". Then user describe device under this metadata template. The process is shown in Figure 5.

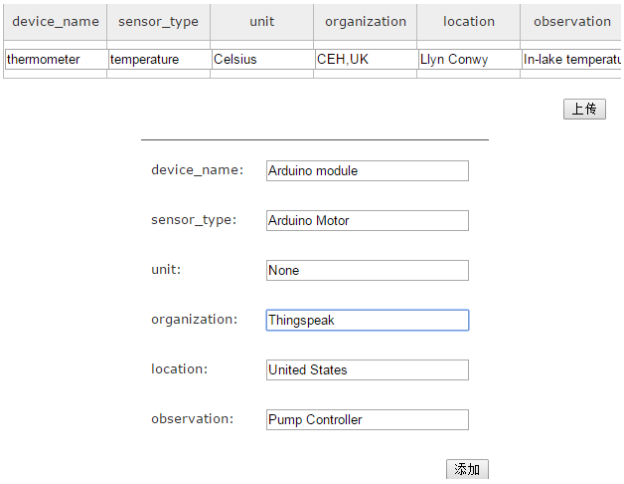

Figure 5 Device description generation page

When a user finished describing all devices and submitted his operation. The description metadata will be delivered to entity annotation module. The result of this module can be seen in Figure 6. The metadata keys and text values have been linked to a DBpedia page.

Figure 6 a device description page with the links to DBpedia

In our application, two keys in different template may describe same property of a device, but they may have different key name. if two key in different metadata template have this relation, a relationship is established between them that both of them can be linked to the same entity in DBpedia. And users can figure out what their key actually refers to.

4.2 Evaluation

4.2.1 Experiment Setup

We use fixedWebTable from Bhagavatula et al. [8] and five tables generated from our application as experiment dataset. The fixedWebTable contains 9177 original texts with their annotations from 428 tables extracted from the web. And the five tables are generated by using different metadata template in our application, which contain 597 cell texts, and it is denoted as Application generated tables.

To train candidates re-ranking module, we select 7500 origin text-annotation pairs from fixedWebTable and use each text as query to search DBpedia endpoint. Then, the result which fully matches the annotation is labeled with a high ranking score, and other results are labeled with low score. Finally, the labeled dataset is used to train SVM ranking model. Test dataset is consisted of two parts. One is the left 1677 texts in fixedWebTable, the other is Application generated tables which we manually annotate each cell text with ground truth in DBpedia. Then we drop the texts which has no ground truth in DBpedia from our dataset and denote this processed dataset as D.

Accuracy is chosen as evaluation metric to score the annotation results, which is standard in information retrieval.

4.2.2 Re-rank Module Evaluation

Re-rank is an important step in our entity linking system. It is known that only one assignment matches the ground truth in DBpeida. For each text in D, we find its re-ranked candidate list, and evaluate re-rank accuracy by judging whether the top N re-ranked entities contain the ground truth. Table I shows the accuracy comparison between two test dataset's parts in re-rank module.

Table I accuracy for the SVM ranking model

Dataset	condition	Ground truth rank in top 10	Ground truth rank in top 1
Fixed WebTable	initial rank	90.3%	57.9%
	re-ranked	94.7%	71.8%
Application generated	initial rank	84.3%	58.4%
	re-ranked	94.4%	83.1%

To compare with the accuracy that a text's ground truth ranks in top N after re-ranking model, we use the origin rank list which is got from querying DBpedia sparql endpoint. The judgment is the same as what we do to re-ranked candidate list. It can be seen in Table I, without a candidate entities re-ranking, the accuracy of "ground truth rank in top 1" has a significant reduction, which means the iterative inference model will take more iterations to converge. To each cell text, we only set top 10 entities in re-ranked list as the input candidate entities to iterative inference module. The decreasing accuracy of "ground truth rank in top 10" will directly result in annotation error due to the absence of the target assignment.

4.2.3 Entity Annotation

After convergence iterations, every cell text in an input table is assigned to an entity in KB or to a string "no-annotation". IMP algorithm can not annotate a nonexistence entity to a text. Thus, we have dropped the entities that missing ground truth in DBpedia. Then we compare the entity links generated by our system to the ground truth manually annotated before. If the assignment generated from the iterative inference model comply with the ground truth, we consider it as a correct prediction. Otherwise, it is wrong. We present the result about accuracy of our algorithm in Table II, and also present accuracy without re-rank module in contrast to show how important re-rank is.

Table II The accuracy of entity annotations

Dataset	IMP only	IMP + re-rank
Application generated	57.2%	74.2%

As input table has several columns, we evaluate entity linking accuracy for each column and find that entity annotations under the column which represent unit has an extraordinary low accuracy, only 26.1%, far below the other columns whose mean accuracy is 88%. Lower accuracy is likely due to the lack of relevant data in DBpedia. Although our system might have discovered the correct assignments for column type and relation, if this entity does not have the same type and relation information in DBpedia, system will miss this correct assignment.

For instance, the column of unit is annotated as DBpedia class "UnitOfMeasurement113583724". Some of input text are "ms" which means millisecond, "kpH" which means kilometers per Hour. Both of them has ground truth in DBpedia, but none of them has "rdf:type" label. It means the lack of type information result in a correct assignment being filtered by our column constraint. Lower accuracy also stems from the size of the candidate entity set. We restricted the size of the candidate entities to 10 in iterative inference. We have presented re-rank accuracy in Table I, which a correct assignment ranking out of top 10 can be still found in.

4.2.4 Performance Evaluation

Five Application generated tables mentioned in 4.2.1 are selected to test how quickly the IMP converges. The number of variable nodes need to be reassigned decrease to zero after the first two iterations. Two reasons can be used to explain this phenomenon. One is re-rank model has a highly accuracy that it can precisely re-rank the target assignment at the top of the re-rank list, the other is that few relations exist between the column in our test dataset. Number of relations is positively related to messages that a variable node received. A variable node with several relations may result in iteration increasing.

The average rows and columns in Application generated Tables are 24 and 5. The average time consuming in each annotation WoT table is 4.1s, and there is considerable variation depending on the number of rows and columns.

5 Conclusions

In this paper, we have studied the problem of metadata annotation of the Web of Things. We proposed a semi-automatic annotation framework that can effectively annotate WoT metadata representations with given KB to the best of our knowledge. We described an Iterative Message Passing algorithm that use constraint from keys and relation between keys in metadata to collective disambiguate an entity annotation. To evaluate the effectiveness of our framework, we conducted an experimental study by providing a typical system use case and an analysis in entity annotation accuracy. The results show that the proposed framework could facilitate linking WoT metadata semi-automatically to the entities, classes and relations in the given KB with few human annotations. This method will improve the effectiveness and efficiency to publish domains-specific WoT resource as Linked Open Data.

Acknowledgements

This work was supported by the Chinese Megaproject under grant No.2015ZX03003012.

References

[1] Guinard, D., Trifa, V., Mattern, F., and Wilde, E. From the Internet of Things to the Web of Things: Resource Oriented Architecture and Best Practices. In Architecting the Internet of Things, D. Uckelmann, M. Harrison, and F. Michahelles, Eds. Springer, Berlin, Germany, 2011, pp. 97-129.

[2] M. Hauswirth and S. Decker, "Semantic reality - connecting the real and the virtual world," Microsoft SemGrail Workshop, pp. 21–22, June 2007.

[3] A. Sheth, C. Henson, and S. Sahoo, "Semantic Sensor Web," IEEE Internet Computing, July/August 2008, 2008, pp. 78–83.

[4] D. Pfisterer et al., "SPITFIRE: toward a semantic web of things," in IEEE Communications Magazine, vol. 49, no. 11, pp. 40-48, November 2011.

[5] Lefort, L., Henson, C., Taylor, K., Barnaghi, P., Compton, M., Corcho, O., Castro, R., Graybeal, J., Herzog, A., Janowicz, K., Neuhaus, H., Nikolov, A., Page, K.: Semantic Sensor Network XG Final Report. W3C Incubator Group Report 28 June 2011.

[6] Varish Mulwad, Tim Finin, and Anupam Joshi. Semantic message passing for generating linked data from tables. In The Semantic Web–ISWC 2013, pages 363–378. Springer, 2013.

[7] Girija Limaye, S Sarawagi, and S Chakrabarti. Annotating and searching web tables using entities, types and relationships. Proceedings of the VLDB, pages 1338–1347, 2010.

[8] Bhagavatula C S, Noraset T, Downey D. TabEL: Entity Linking in Web Tables [M]. The Semantic Web – ISWC 2015. Springer International Publishing, 2015.

[9] http://dbpedia.org/sparql

[10] T. Joachims. Training Linear SVMs in Linear Time, Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD), 2006.