Methods for Aggregating Crowdsourced Ontology-based Item Annotations

Andrew Ponomarev
St. Petersburg Federal Research Center of the Russian Academy of Sciences
St. Petersburg, Russian Federation
ponomarev@iias.spb.su

***Abstract*—Crowdsourcing plays an important role in modern IT landscape, enabling the use of human information processing abilities to solve problems that are still hard for machines. One of the specific (and most demanded) applications of crowdsourcing is collecting item annotations, i.e., describing the contents of complex items with a help of labels (tags). Input received from crowdsourcing participants is typically unreliable, therefore, to increase the quality of annotations, each item is typically processed by several participants and the obtained annotations have to be aggregated. The paper considers a special case of annotating, where a set of possible labels, as well as the set of relationships between the labeled items and the labels are defined by an OWL 2 ontology (OWL QL). Such semantic item annotations turn out to be very useful in organizing large collections of items and enabling semantic search in them. In order to increase annotations quality, the paper proposes two aggregation methods – OntoVoting and OntoSB, differing in that the first one is agnostic with respect to participants' reliability and the second one accounts for variations in reliability. Simulation experiments with ontology-based annotations of varying quality show that the proposed aggregation methods increase the quality of collected ontology-based item annotations.**

I. Introduction

Crowdsourcing has become a widely used tool in a growing palette of IT tools, helping to address many problems that are hard for machines. An important kind of crowdsourcing applications is collecting annotations. In these applications, each participant is asked to associate some labels (tags) with a given item (usually, a complex one – text, video, or image). It involves understanding the contents of the item and finding the most appropriate labels for it. Such annotations are then may be used either to train AI models (for automating the process of item contents interpretation), or directly to enable tag-based search in data collections (it is typically much simpler to implement a tag-based search, than to automatically interpret complex items).

An important characteristic of crowd-based applications for collecting annotations is what is considered as an acceptable annotation. It is a spectrum, ranging from binary labels (e.g., a picture meets some criterion or not), through using a controlled vocabulary of tags (of varying size), to accepting any participant's input.

The paper deals with a special case of crowdsourced annotation problem, where each annotation is expressed as a set of formal statements in OWL 2 language [1]. In particular, it means that both a set of possible labels and a set of relationships between an item being labeled and the labels are defined by some OWL 2 ontology. Ontologies have received much attention in Semantic Web initiative and proved themselves to be an effective tool of reaching semantic interoperability, as they a) define the precise meaning of terms, b) describe relationships between terms, c) are based on formal models, enabling inference of information not provided explicitly (usually, on description logics). OWL 2, a W3C recommendation, is currently one of the most popular ontology languages that is used to represented variety of the ontologies in a range of application areas.

As input received from human participants is typically unreliable (due to lack of knowledge or effort), any crowdsourcing application has to implement some quality management strategy. In many cases, such strategy is based on redundancy – the same task is assigned to several participants and then results are aggregated in order to increase their reliability (the procedure is based on an assumption that errors introduced by the participants are independent). Obviously, the aggregation mechanism is the core of redundancy-based quality management. Variety of label aggregation techniques have been proposed [2]–[5], however, only few of them take into account relationships between labels, and if they do, they often consider pairwise relationships between labels, which turns out to be inapplicable when the number of labels is high. At the same time, ontology concepts are explicitly related and this can be used to improve annotations quality leveraging some redundancy of the input.

The problem of aggregating ontology-based descriptions was already considered in our previous papers [6], [7], however, in [7] we discussed only one construct of OWL – subclass assertion, and the hierarchical voting approach proposed in [6] doesn't take into account possible differences in participants' reliability.

In this paper, we propose two methods for aggregating annotations – OntoVoting and OntoSB. Both methods utilize the ontology structure to infer the most reasonable aggregated annotation from the set of unreliable ones and therefore to increase the annotation quality, correcting individual errors resulting from the lack of annotators' effort or understanding. The difference between the proposed methods is in the set of considered information. OntoVoting treats all participants as

ISSN 2305-7254

equally (un)reliable and can be used in situations where there is no prior information about participants (it is based on the method from [6]). Contrary, OntoSB leverages a reasoning under uncertainty technique and can take into account prior participant reliability. We perform a series of simulation experiments to understand the effect of the proposed aggregation methods on the quality of annotations.

The rest of the paper is structured as follows. Section II discusses related work on label aggregation using external knowledge. Section III describes formal model of ontology-based item annotation. Section IV presents the aggregation methods, and Section V – experimental setup and results.

II. Related Work

Existing label aggregation techniques are typically based on strict comparison of labels obtained from individual participants, ignoring possible relationship between them. These methods are suited for situations where there are few possible labels and these labels are disjoint (e.g., binary labels reflecting presence or absence of some feature). Obviously, these techniques do not work well in situations where there are many possible (interrelated) labels (in particular, ontology statements). There are several papers on adapting consensus methods to situations where there are relations between labels. In [8], the authors propose an extension of the Dawid-Skene algorithm [9] and explore different models for representing relations between labels (including a Bayesian network as a compact representation of a joint distribution). However, Dawid-Skene algorithm has high computational complexity, therefore, its applicability for labeling using large ontologies is limited. In [10], a probabilistic labeling model for hierarchical classification is proposed, that is, for the situation when the labels that participants assign to objects are classes organized in a hierarchy (classification of books, goods). This is especially close to the considered problem, because typically a core of an ontology is a class taxonomy, defined with a help of OWL 2 SubClassOf construct. However, the method from [10] is intended for sequential tag refinement by a participant in a series of tasks, which is not always reasonable.

The closest problem definition is considered in [6], [7]; these papers also consider aggregation of ontology-based annotations. However, [7] considers only SubClassOf ontology construct, while algorithms proposed in this paper account for nine OWL constructs (in class, property, and individual levels). The method proposed in [6] is based on the assumption that all participants are equally reliable. This assumption is relaxed in the OntoSB algorithm, proposed in this paper.

III. Formal Model

This section describes the annotation structure, considered in the paper, and lists OWL 2 language constructs that can affect the aggregation procedure.

A. Item Annotation Structure and Problem Definition

Ontology-based item annotation includes two kinds of information: ontology specification and item annotation. In this paper, we consider specifically ontologies expressed in OWL 2. An example of such ontology is shown in Fig. 1. The ontology specifies a set of classes to be used for thematic classification (organized in a hierarchy), class Item, and three properties (hasTopic, hasPrimaryTopic, and hasLength) that can be used to describe items.

Item annotation is a set of OWL 2 statements (assertions) linking the item with some ontology entities (classes or individuals) and/or using properties defined in the ontology. This paper considers annotations consisting of only two kinds of statements:

- ObjectPropertyAssertion(*OP*, *item*, *v*), where *OP* is an object property defined by the ontology used for annotation, *item* is the item being annotated, *v* is some entity, introduced by the ontology.
- DataPropertyAssertion(*DPE*, *item*, *lt*), where *DPE* is a data property defined by the ontology, *item* is the item being annotated, and *lt* is some literal.

```
Prefix(:=<http://purl.org/ontologies/fruct/s#>)
Ontology( <http://purl.org/ontologies/fruct/s>
  Declaration(Class(:Item))
  Declaration(ObjectProperty(:hasPrimaryTopic))
  Declaration(ObjectProperty(:hasTopic))
  Declaration(DataProperty(:hasLength))

  SubClassOf( :InformationTechnology owl:Thing )
  SubClassOf( :AI :InformationTechnology )
  SubClassOf( :AIOntology :AI )
  SubClassOf( :Crowdsourcing :InformationTechnology )

  SubObjectPropertyOf( :hasPrimaryTopic :hasTopic )
  ObjectPropertyDomain( :hasTopic :Item)

  DataPropertyDomain( :hasLength :Item)
  DataPropertyRange( :hasLength xsd:int)
)
```

Fig 1. Example ontology

Example of an item annotation using the ontology shown in Fig. 1 could be the following:

```
<https://doi.org/10.1000/xyz123>
    o:hasPrimaryTopic o:Crowdsourcing .
<https://doi.org/10.1000/xyz123>
    o:hasTopic o:AIOntologies .
<https://doi.org/10.1000/xyz123>
    o:hasLength "5"^^xsd:int .
```

This describes an item with the unique identifier <https://doi.org/10.1000/xyz123> (using URI or, in general, IRI is part of semantic web standards stack), using two object properties introduced by the ontology and one data property. To save space, we use the namespace ‘o’, implying that it is defined to match the ontology URI. Note, that for simplicity we express object property assertions and data property assertions as RDF triples, according to mapping specified in [11].

Let an item x can be completely described by a set of statements $A^*(x)$, such that each statement actually corresponds

to the contents of x, none of the statements in $A^*(x)$ follow from other statements, and no statements could be added (correspond to the contents of x but are not included in $A^*(x)$).

The goal of the problem-setter is to obtain annotation $A^*(x)$, but it is generally unknown, and what the problem-setter can get are annotations, that are close to the real one, but with possible deviations (too general statements, missing statements, not relevant statements).

The ontology-based annotation aggregation problem can be specified by a tuple $P = <x, U, A, M, O>$, where x – is an item, U is a set of users (annotation participants), A is a set of annotations of the item, M is a mapping between annotations and their authors ($A \rightarrow U$), and O is the ontology used in the annotations A. The aggregation method $\mathcal{M}(P)$ should result in some annotation for item x that is as close as possible to $A^*(x)$. This specification can be extended by user attributes (e.g., reliability), and one of the proposed methods, namely OntoSB, actually is based on an extended formulation. The notion of annotation "closeness" (or, similarity) is formalized in Section V. The proposed methods are not driven by loss minimization, and the formal definition of similarity is used only for the evaluation.

B. OWL Constructs

As it was noted earlier, two statements shouldn't be simply compared for equality, because difference in the used property or property value doesn't necessarily mean that the meaning of the statements is entirely different. As properties and property values are described in the ontology, there might be relationships stating that two different values (with different identifiers) are essentially the same. We analyzed a set of constructs of OWL QL profile, a subset of OWL 2, providing efficient query processing and suitable for crowdsourcing, where a large number of objects is annotated with a relatively simple ontology. In particular, we identified three groups of such statements in OWL QL. The first group is concept and property hierarchy assertions:

- SubClassOf – Defines one class as a subclass (or, more specialized class of another class). Instances of the subclass are also instances of the more general class.
- SubObjectPropertyOf/SubDataPropertyOf – States that the extension of one object property expression is included in the extension of another object property expression.

The second group is equivalence (or, synonym) assertions:

- EquivalentClass – Asserts that two classes are equivalent, i.e. contain exactly the same set of individuals.
- EquivalentObjectProperties – States that the extensions of several object property expressions are the same.

Finally, the assertions of the third group are used to explicitly state difference and inconsistency:

- Disjoint – Asserts that no individual can be at the same time an instance of two classes of the specified ones.
- DisjointObjectProperties/DisjointDataProperties – States that the extensions of several object property expressions are pairwise disjoint — that is, that they do not share pairs of connected individuals.
- DifferentIndividuals – States that several individuals are all different from each other.

IV. Annotation Aggregation Methods

This section introduces two methods for aggregating ontology-based annotations: OntoVoting and OntoSB. They share similar principle, but OntoSB takes into account possible difference in expected quality between annotators.

A. OntoVoting

The proposed aggregation methods are based on an observation that an annotation statement can be generalized (following the ontology specification) without losing its validity. For example, let one of the participants stated that some article is primarily dedicated to crowdsourcing:

```
<https://doi.org/10.1000/xyz123>
  o:hasPrimaryTopic o:Crowdsourcing .
```

According to the formal definition of the OWL 2 SubClassOf construct, which is used in the ontology (Fig. 1) to describe class Crowdsoucing, any individual belonging to this class also belongs to class InformationTechnology. Therefore, the statement:

```
<https://doi.org/10.1000/xyz123>
  o:hasPrimaryTopic o:InformationTechnology .
```

is also valid. It is less specific, however, valid. This generalization can be continued, using other SubClassOf definitions to the statement that the primary topic of the paper is owl:Thing.

Moreover, the ontology uses OWL 2 SubObjectPropertyOf to establish relation between hasPrimaryTopic and hasTopic, which means that any pair of entities connected by hasPrimaryTopic property are also connected by a more general property hasTopic. Therefore, the original statement can also be generalized to:

```
<https://doi.org/10.1000/xyz123>
  o:hasTopic o:Crowdsourcing .
```

And further to:

```
<https://doi.org/10.1000/xyz123>
  owl:ObjectProperty o:Crowdsourcing .
```

where owl:ObjectProperty is a top object property. Both paths of generalization can be applied independently, so there are six statements following from the original statement.

On the other hand, there are disjoint classes, implying that an individual may belong only to one of these classes, and disjoint properties, implying that no two entities can be connected by both properties. It means that along with possible generalizations of the original statement, there are also some

negative statements that follow from the original statement. We propose Algorithm 1 (StatementConsequences) for identifying both positive and negative consequences of a given statement. For a given statement (in a form of a triple – object, property, value), the algorithm builds two sets of triples – positive and negative consequences, the statements that should also be true, and the statements that has to be false, respectively. The algorithm relies on ValueGeneralizations, PropertyGeneralizations, and Disjoints functions. These functions are typically provided by a programming library for processing ontologies. ValueGeneralizations function builds a list of all concepts that are generalizations of a given concept (or value), using SubClassOf, EquivalentClass, and ClassAssertion statements. For example, in the ontology, shown in Fig. 1, ValueGeneralization function for the concept "Crowdsourcing" would return a set, consisting of "Crowdsourcing", "InformationTechnology", and "owl:Thing". PropertyGeneralizations function does similar thing for properties, based on inference using SubObjectPropertyOf, SubDataPropertyOf, and EquivalentObjectProperties assertions. Disjoints function returns a set of all concepts that are explicitly stated as different or disjoint with the specified one, using Disjoint and DifferentIndividuals ontology constructs. Finally, DisjointProperties returns properties disjoint with the given one (interpreting DisjointObjectProperties/DisjointDataProperties constructs).

Algorithm 1 StatementConsequences

Input: statement $< item, p, v >$
1: $S^+, S^-, V^+, V^- := \varnothing$
2: **for** $v' \in$ ValueGeneralizations(v)
3: $V^+ := V^+ \cup \{v'\}$
4: $V^- = V^- \cup$ Disjoints(v')
5:**for** $p' \in$ PropertyGeneralizations(p)
6: $S^+ = S^+ \cup \{<item, p', pv> \mid pv \in V^+\}$
7: $S^- = S^- \cup \{<item, p', nv> \mid nv \in V^-\}$
8: **for** $np \in$ DisjointProperties(p')
9: $S^- = S^- \cup \{<item, p', pv> \mid pv \in V^+\}$
10:**return** $< S^+, S^- >$

Algorithm 2 AnnotationConsequences

Input: annotation A (set of statements)
1: $M :=$ **dictionary**()
2: **for** $s \in A$
3: $< S^+, S^- >$:= StatementConsequences(s)
4: **for** $s' \in S^+$
5: **if** $s' \in M$.**keys**
6: $M[s'] =$ **max**($M[s']$, 1)
7: **else**
8: $M[s'] = 1$
9: **for** $s' \in S^-$
10: **if** $s' \in M$.**keys**
11: M[s'] = **max**($M[s']$, -1)
12: **else**
13: $M[s'] = -1$
14: **return** M

Participant's annotation usually contains several statements, and usually there are generalized statements that follow from more than one original statement (indeed, a statement that the value of an owl:ObjectProperty of the item is owl:Thing generalizes vast majority of statements). Algorithm 2 (AnnotationConsequences) shows the proposed algorithm for finding all consequences of an annotation, consisting of several statements. In OntoVoting algorithm all positive consequences are assigned 'vote' of 1, and all the negative consequences are assigned 'vote' of -1. If a statement is receives a 'vote' of 1 from one statement, and a 'vote' of -1 from another, resulting 'vote' is assumed to be 1.

Finally, OntoVoting aggregation algorithm is organized in the following way: it builds consequences of each annotation (provided by different participants), sums votes for respective statements, filters only those statements that have the specified number of votes, and removes all the statements that follow from some other statements in the resulting set (Algorithm 3).

Algorithm 3 OntoVoting

Input: Q - set of annotations from different participants,
τ - votes threshold.
1: $M :=$ **dictionary**()
2: # Find logical consequences of each annotation
3: # (summing votes)
4: **for** $q \in Q$
5: $V :=$ AnnotationConsequences(q)
6: **for** $s \in V$.**keys**
7: **if** $s \in M$.**keys**
8: $M[s] =$ Combine(sum, $M[s]$, $V[s]$)
9: **else**
10: $M[s] = V[s]$
11: # Filter out statements that don't have enough support
12: $S := \{s \mid s \in M$.**keys**, $M[s] >= \tau\}$
13: # Filter out non-specific statements
14: **for** $s \in S^-$
15: $G^+, G^- :=$ StatementConsequences(s)
16: $S := S \setminus G^+$
17: **return** S

B. OntoSB

In OntoVoting algorithm, evidence from each participant is equally important, as every participant has exactly one 'vote' that is propagated to all statements that follow from the participant's annotation. However, in many situations, participants differ by knowledge and/or effort and in these situations, it may be reasonable to give different value for their input. OntoSB allows for taking this into account by plugging a model for reasoning under uncertainty to the propagation scheme of the OntoVoting algorithm. In particular, OntoSB uses Shortliffe-Buchanan scheme for reasoning under uncertainty [12] (hence, the name), however, other options are also possible (e.g., Dempster-Shaffer theory [13]).

Shortliffe-Buchanan scheme was originally used in expert system (MYCIN) and is based on degree of belief, associated to facts. Degree of belief is a real number in the range [-1; 1], where -1 corresponds to "highly unlikely" and 1 corresponds to

"highly likely". In the proposed OntoSB method we associate reliability (in the range [0; 1]) to every participant. The participant's reliability is associated to all positive consequences of the participant's annotation, and negative reliability is associated to all negative consequences. Therefore, the algorithm for finding consequences basically stays the same as for OntoVoting (Algorithm 2), however, fixed votes (1 and -1) are changed to the participant's reliability value.

Shortliffe-Buchanan scheme defines the following rule to calculate the degree of belief for the fact that has two evidences with degrees of belief b_1 and b_2 [12]:

$$b_1 \circ b_2 = \begin{cases} b_1 + b_2 (1 - b_1), b_1 >= 0, b_2 >= 0 \\ b_1 + b_2 (1 + b_1), b_1 < 0, b_2 < 0 \\ b_1 + b_2 / [1 - \min(|b_1|, |b_2|)] \end{cases}$$

This degree of belief combination formula is used in line 8 of the Algorithm 3 instead of summation. These two changes (initial values for statement weight and another combination operator) transform OntoVoting into OntoSB algorithm.

V. Evaluation

The evaluation of the proposed methods is based on simulation of the noisy annotations. This section explains the evaluation methodology (the process of data generation and ontology generation and measuring the annotation quality) as well as the comparison of the algorithms.

A. Ontologies

We generated several ontologies for the simulation study. All these ontologies share the same organization principles, but differ in size. The skeleton of each of the generated ontologies is several concept hierarchies, defined by a SubClassOf construct, which is typical for most of the real-life ontologies. The generated ontologies also include a number of "synonym" classes (1/3 of all the classes forming the hierarchies). Besides, each ontology defines five object properties organized into two object property hierarchies. The ontologies do not include data properties, as the evaluation mostly aims at exploring conceptual aggregation, to deal with data properties any data aggregation approach could be plugged.

In the simulation study, we used three generated ontologies of different sizes: small, medium, and large:

- Small – 2 hierarchies, each with 4 levels, 3 subclasses per non-leaf class. In one hierarchy the subclasses are disjoint. There are 313 classes in total.
- Medium – 4 hierarchies, each with 4 levels. Two of them has 3 subclasses per non-leaf class, the other two – 4 subclasses. In half of the hierarchies, the subclasses are disjoint. There are 1197 classes in total.
- Large – simply twice as big as the medium one, 2393 classes in total.

B. Ground Truth Annotations

For each ontology, a ground truth dataset was generated, including annotations of 500 items. Each item annotation consists of one hasPrimaryTopic property and two other properties selected at random. Values of each property are selected randomly from all the classes of the respective ontology.

C. Participant Model

To build a participant behavior model, we analyzed possible types of error [14]–[17]. In particular, each participant submits a set of statements in the «object – property – value». Therefore, errors can be associated with the number of statements, properties used and the specified property value in each statement. In particular, when choosing a property of a value, the following types of errors are possible: a) using too general property or value (insufficient specification), b) using too specific or unrelated property or value. Besides, a participant may skip some of the statements of the true annotation or add some noise statements.

In the simulation model, each participant is characterized by the following characteristics:

- Observancy, describing how many of the ground truth statements the participant can detect. It is defined as a probability that the participant considers a particular true statement.
- Diligence, describing the propensity of using exact values, not generalizing them. It is defined as a probability that some considered true statement will be reported "as is". With probability $1 - d$ the property name or object value will be changed to a more general (chosen randomly).
- Noise, controlling how many statements unrelated to the true statements a participant will generate. It is defined as a probability to generate a noise statement (where property and value are chosen at random). Noise statements are generated iteratively while the value returned by a uniform random generator in range [0; 1) is less than the value of noise parameter.

We used three types of participants in the simulation:

- high quality (observancy 0.9, diligence 0.9, noise 0.1);
- medium quality (observancy 0.75, diligence 0.75, noise 0.2);
- low quality (observancy 0.6, diligence 0.6, noise 0.4).

D. Annotation Penalty Score

To measure the similarity of descriptions (e.g., ground truth descriptions and aggregated) we use the following score.

Let $D^{(1)} = \{s_i^{(1)}\}$ and $D^{(2)} = \{s_i^{(2)}\}$ be two annotations defined as sets of statements. Further, let $G(s)$ be a set of statements that generalize statement s, and $L(s, s')$ be a minimal number of generalization steps to transform s to s'. Then, statement penalty score is calculated as:

$$p^{(1)}(s) = \min_{s_i^{(2)} \in D^{(2)}, s'} \left(L(s, s') + L\left(s_i^{(2)}, s'\right) \right), s \in D^{(1)},$$

$$p^{(2)}(s) = \min_{s_i^{(1)} \in D^{(1)}, s'} \left(L(s, s') + L\left(s_i^{(1)}, s'\right) \right), s \in D^{(2)}.$$

In other words, penalty for a statement (with respect to some other annotation), is the minimal distance (in generalization operations) to any of the statements of the other annotation.

Annotation penalty score is calculated as:

$$P\left(D^{(1)}, D^{(2)}\right) = \sum_{s_i \in D^{(1)}} p^{(1)}(s_i) + \sum_{s_j \in D^{(2)}} p^{(2)}(s_i).$$

In the presented results, the value of this metric is the annotation penalty score averaged per items.

E. Results and Discussion

The first experiment is to verify that the proposed aggregation method (OntoVoting) is able to improve annotations with respect to the annotations provided by one participant and to understand the effect of algorithm parameters (number of voters, voting threshold) on the resulting annotation quality. To do so, we used small ontology and medium quality participants. Table I shows the average annotation penalty score after 10 simulations (for 500 annotated items), standard deviation is shown in parenthesis. The average quality of annotations of one medium quality participant is 8.38 (0.39).

It can be seen, that with three or more participants working on an annotation, it is possible to achieve significantly better annotation quality, than with one annotator. When the number of annotators is greater than five, the gain of the annotation quality is diminishing. Most effective threshold is 2. It can be explained by the fact, that with more strict thresholds, the consensus is reached only in higher levels of aggregation, increasing the annotation penalty.

TABLE I. THE EFFECT OF ONTOVOTING PARAMETERS

	Redundancy				
Threshold	2	3	4	5	6
2	10.0 (0.15)	**5.43** (0.15)	**2.83** (0.16)	**1.88** (0.17)	**1.7** (0.12)
3	-	11.93 (0.16)	7.68 (0.2)	4.36 (0.11)	2.4 (0.14)
4	-	-	13.0 (0.11)	9.59 (0.18)	6.09 (0.21)

TABLE II. ONTOLOGY SIZE AND ANNOTATION QUALITY

Ontology Size	Single	Redundancy			
		3	4	5	6
Small	8.25 (0.28)	5.31 (0.16)	2.9 (0.18)	1.94 (0.1)	1.67 (0.12)
Medium	8.99 (0.34)	5.78 (0.19)	3.03 (0.17)	1.94 (0.09)	1.71 (0.08)
Large	8.92 (0.26)	5.64 (0.16)	3.01 (0.19)	1.86 (0.1)	

TABLE III. ONTOLOGY SIZE AND ANNOTATION QUALITY

Annotator Quality	Single	Redundancy			
		3	4	5	6
Low	14.47 (0.49)	10.3 (0.17)	8.15 (0.31)	7.5 (0.29)	7.25 (0.21)
Medium	8.19 (0.19)	5.29 (0.16)	2.85 (0.11)	1.89 (0.12)	1.79 (0.14)
High	3.52 (0.14)	1.28 (0.1)	0.41 (0.06)	0.3 (0.04)	0.37 (0.07)

Table II shows how the ontology size influences the annotation quality. All the annotations were done by a medium quality participant, the threshold was set to 2 (the results shown are averages of 10 runs).

The quality of annotation doesn't depend on the size of the ontology (all variations are within confidence intervals). OntoVoting algorithm with redundancy 3 to 5 provides significant gains in annotation quality (relative gains are roughly the same for all the examined ontology sizes).

Table III shows the effect of the quality of the original annotations on the aggregated ones (on the small ontology dataset). Aggregation turns out to be effective for each of the original annotation qualities, however, it has certain limits – no matter how many low quality participant annotations are aggregated, the result is still significantly worse, than could be produced by one high quality participant.

TABLE IV. ONTOVOTING AND ONTOSB COMPARISON

Dataset (h-m-l)	Single	OntoVoting (thr = 2)	Best OntoSB
Redundancy 4			
1-1-3	11.75 (0.33)	4.83 (0.14)	**3.16** (0.11)
1-3-1	8.9 (0.19)	**2.89** (0.15)	2.9 (0.16)
3-1-1	6.9 (0.27)	1.6 (0.1)	**1.52** (0.12)
Redundancy 5			
1-1-3	11.75 (0.33)	**3.87** (0.14)	3.97 (0.08)
1-3-1	8.9 (0.19)	2.07 (0.14)	**1.97** (0.17)
3-1-1	6.9 (0.27)	1.15 (0.11)	**0.93** (0.08)

Please, note, that for the experiments above, OntoSB can obtain similar results. If the reliability of all the participants is the same, then combination of 1, 2, 3, etc. belief values would yield only the number of distinct values equal to the number of participants, annotating each item (redundancy). So, picking a degree of belief threshold in this case is equivalent to picking a redundancy threshold. Essentially, OntoSB is designed for the situation, where the participants are different. To model this situation a dataset with varying annotators was generated. In particular, participants, annotating items were selected randomly with different probability (e.g., high quality – 0.2, medium quality – 0.2, low quality – 0.6). Table IV summarizes results on some distributions, showing the best aggregated annotation quality, obtained by each algorithm. Dataset column describes proportion of the workers in the order high-medium-low. It is important to note, that in this experiment OntoSB was given participant confidence values tuned for each particular situation. In general, experiments with OntoSB show that the aggregation quality of OntoSB is usually either comparable with OntoVoting, or slightly beats it (in some cases – significantly, e.g., on 1-1-3 dataset OntoSB with 4 participants gives better quality than OntoVoting with 5). However, OntoSB turns out to be very sensitive to the participant

reliability values. Without a justified scheme for selecting these values, OntoVoting might be preferable due to its robustness.

VI. Conclusion

The paper considers the problem of aggregating semantic annotations, expressed as represented as sets of statements, describing items in terms of some OWL QL ontology. We examined OWL QL constructs to identify those, which can be utilized in the process of aggregation. Based on this analysis we proposed two algorithms – OntoVoting and OntoSB. The experimental evaluation on a simulated dataset shows that the proposed algorithms effectively utilize the redundancy, significantly improving the quality of semantic annotations, obtained from unreliable participants.

Potential applications of the semantic description aggregation are various crowd-based annotation systems, especially in the fields where many high-quality ontologies exist (e.g., scientific research, and, in particular, biomedical sciences). Besides, as ontologies are a universal tool, based on strong logical foundations, some problems related to collecting descriptions may be reduced to the ontology-based semantic annotation, which widens the scope of possible applications even further.

In this paper we assume, that the participant's reliability (used by OntoSB) is known, but it is not always the case. Further research is dedicated to methods of evaluating the reliability based on participant's behavior, similarly to many other aggregation algorithms, which simultaneously estimate item labels and user features.

Acknowledgement

The reported study was funded by Russian Foundation for Basic Research, project number 19-07-01120.

References

[1] W3C, "OWL 2 Web Ontology Language Document Overview," 2012. [Online]. Available: https://www.w3.org/TR/owl2-overview/. [Accessed: 10-Oct-2020].

[2] N. Quoc Viet Hung, N. T. Tam, L. N. Tran, and K. Aberer, "An Evaluation of Aggregation Techniques in Crowdsourcing," in *Web Information Systems Engineering – WISE 2013, Lecture Notes in Computer Science, vol 8181*, H. G. Lin X., Manolopoulos Y., Srivastava D., Ed. Berlin: Springer, Berlin, Heidelberg,2013,pp.1–15.

[3] A. Sheshadri and M. Lease, "Square: A benchmark for research on computing crowd consensus," in *1st AAAI Conf. Human Comput. Crowdsourcing*, 2013, pp. 156–164.

[4] A. I. Chittilappilly, L. Chen, and S. Amer-Yahia, "A Survey of General-Purpose Crowdsourcing Techniques," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 9, pp. 2246–2266, 2016.

[5] J. Zhang, V. S. Sheng, and J. Wu, "Crowdsourced Label Aggregation Using Bilayer Collaborative Clustering," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 10, pp. 3172–3185, Oct. 2019.

[6] A. Ponomarev, "Aggregation of Crowdsourced Ontology-Based Item Descriptions by Hierarchical Voting," in *Proceedings of the 2nd Crowd Science Workshop: Trust, Ethics, and Excellence in Crowdsourced Data Management at Scale co-located with 47th International Conference on Very Large Data Bases (VLDB 2021)*, 2021, pp. 60–71.

[7] A. Ponomarev, "An Iterative Approach for Crowdsourced Semantic Labels Aggregation," in *Advances in Intelligent Systems and Computing, vol 1295*, 2020, pp. 887–894.

[8] L. Duan, S. Oyama, H. Sato, and M. Kurihara, "Separate or joint? Estimation of multiple labels from crowdsourced annotations," *Expert Systems with Applications*, vol. 41, no. 13, pp. 5723–5732, 2014.

[9] A. P. Dawid and A. M. Skene, "Maximum Likelihood Estimation of Observer Error-Rates Using the EM Algorithm," *Applied Statistics*, vol. 28, no. 1, pp. 20–28, 1979.

[10] N. Otani, Y. Baba, and H. Kashima, "Quality control of crowdsourced classification using hierarchical class structures," *Expert Systems With Applications*, vol. 58, pp. 155–163, 2016.

[11] W3C, "OWL 2 Web Ontology Language Mapping to RDF Graphs," 2012. [Online]. Available: http://www.w3.org/TR/owl-mapping-to-rdf.

[12] E. H. Shortliffe and B. G. Buchanan, "A model of inexact reasoning in medicine," *Mathematical Biosciences*, vol. 23, no. 3–4, pp. 351–379, Apr. 1975.

[13] J. Y. Halpern, *Reasoning under uncertainty*. MIT Press, 2017.

[14] B. Frenay and M. Verleysen, "Classification in the Presence of Label Noise: A Survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 5, pp. 845–869, May 2014.

[15] C. Eickhoff and A. P. de Vries, "Increasing cheat robustness of crowdsourcing tasks," *Information Retrieval*, vol. 16, no. 2, pp. 121–137, 2013.

[16] G. Kazai, J. Kamps, and N. Milic-Frayling, "Worker types and personality traits in crowdsourcing relevance labels," *Proceedings of the 20th ACM international conference on Information and knowledge management - CIKM '11*, pp. 1941–1944, 2011.

[17] U. Gadiraju, R. Kawase, S. Dietze, and G. Demartini, "Understanding Malicious Behavior in Crowdsourcing Platforms," *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems - CHI '15*, pp. 1631–1640, 2015.