Semantic Web 0 (0) 1
IOS Press

Ontologies' Mappings Validation and Annotation Enrichment Through Tagging

Peter Ochieng [a,*],
[a] *Computer Science, University of Makerere, , Uganda*
E-mail: onexpeters@gmail.com

Abstract. Pay as you go ontology matching, the technique of first executing an automatic matching tool and then engaging user(s) to improve the quality of an alignment produce by the tool is gaining popularity. Most of the existing techniques employ a single user to validate mappings by annotating them using terms from a controlled set such as "correct" or "incorrect". This single user based approach of validating mappings using a controlled set of vocabulary is restrictive. First, the use of controlled vocabulary does not maximize the effort of user since it restrains her from adding more meaning to the concepts participating in low-quality mappings using her own terms. Secondly, a single user approach of validating wide range of mappings is error prone since even the most experienced user may not be familiar with all subtopics contained in the input ontologies. We demonstrate in this research that through tagging of concepts participating in mappings flagged as low-quality, we can achieve both mappings' validation and ontology metadata enrichment by adding quality annotations to the ontology.

Keywords: Collaborative, Matching, ontology, Social, Repair, Alignment, Tagging

An ontology is a formal specification of shared conceptualization which provides a vocabulary describing the domain of interest [1]. It plays an important role in sharing and reusing knowledge among software agents. Due to their importance, multiple independent ontologies have been developed with some level of overlapping information among them. To enable integration of these independent ontologies, ontology matching, the task of finding correspondences between semantically related entities of different ontologies [2] is usually performed. Consequently, many tools have been developed with the aim of performing automatic matching of heterogeneous ontologies [3, 4]. However, today's ontologies' sizes have become large hence presenting scalability challenges to the tools matching them. Key among these challenges is the requirement that a tool has to reason over a large number of input ontologies' axioms in order to establish mappings that exist between their entities. This increased demand for reasoning on a tool degrades its efficiency making it to establish a significant number of wrong mappings between entities of input ontologies [5, 6]. To mitigate the effects of degraded efficiency, some tools such as LogMap [7], ALCOMO [8] and [9] have alignment repairing capabilities to automatically repair inconsistent mappings that exist in an alignment. However, as discussed in [10], the automatic repairing techniques suffer from a number of issues:

- They remove high number of correct mappings from the final alignment.
- When different repair techniques are applied to a given alignment, they produce different repaired alignments i.e. there is no consistency in the repaired alignments of the tools.

These problems of automatic alignment repair techniques have motivated different research works to explore how user(s) can be integrated into the ontology matching process with a goal to improve the quality of mappings produced between two ontologies. One key strategy that is gaining popularity is the pay as you go ontology matching technique. This is a strategy of performing initial ontology matching automatically and then the quality of the established alignment is refined over time [11]. For example, research in [11–13] have

*Corresponding author. E-mail: onexpeters@gmail.com.

incorporated users who improve the quality of mappings generated automatically. The existing research on pay as you go technique however still suffer from a number of drawbacks:

1. They restrict users to only evaluate the accuracy of the matched concepts through constrained terms such as "similar to", "corresponds to" etc. contradicting the work in [14] which established that users prefer tagging of concepts by assigning them their own meaning rather than being restricted to narrow range of mapping terminologies such as to answer questions like "corresponds to" and "similar to".
2. They employ a single user to evaluate mappings, a strategy which is error prone since it is very difficult to find an expert who is well versed with all subtopics in a large ontology.
3. They fail to maximize the effort of a user who apart from verifying mappings, can also enrich the contents of an ontology.

This research therefore seeks to fill these gaps by employing multiple users in a collaborative fashion to tag concepts that participate in mappings that are flagged as low quality. Through tagging these concepts, we demonstrate that both mapping validation and ontology's meta data (annotation) enrichment can be achieved. Users are selected based on their expertise in a given ontology domain hence they are considered domain experts who can assign alternative labels to an entity during the tagging process. Employing multiple users in mapping validation significantly reduces the degree of erroneous validations that is associated with single user. With tagging, users enter labels in a free form to tag concepts (entities) of an ontology. From users' tags we are able to validate if two concepts of a mapping are similar or not. Furthermore, based on work by [15] who showed that lightweight ontologies can be derived from the users' tags, we utilize quality tags that are associated with an entity to enrich its annotation in an ontology. The tags are embedded as an entity's label annotation. By doing this, we are able to add more meaning to an entity. Our work is restricted to validating equivalent mappings.

1. Definition of Terms

In this section, we define key terms used in this paper to make it self-sufficient.

Ontology matching is the process of finding semantic relationships that exist between entities of two ontologies.

A correspondence between two ontologies $\mathcal{O}_T$ and $\mathcal{O}_S$ is a triple $\langle e_1, e_2, r\rangle$ where the entity $e_1 \in \mathcal{O}_S$, $e_2 \in \mathcal{O}_T$ and r is the semantic relationship that exists between e_1 and e_2. It can also be referred to as a mapping. In some instances, a correspondence can be represented as a 4-uple, in such a case the degree of confidence of a correspondence is also included i.e $\langle e_1, e_2, r, v\rangle$ where $v \in [0, 1]$ is the confidence level of the semantic relationship r.

An alignment is a set of correspondences between two ontologies.

A matcher is an automatic matching algorithm employed by ontology matching tool to establish an alignment between two ontologies.

Similarity matrix is a matrix $M = (s_{ij})_{i=1,..n,j=1,..m}$ generated by a matcher L after matching the entities of the source ($\mathcal{O}_S$) and target ($\mathcal{O}_T$) ontologies. Each entry s_{ij} of the matrix M is the similarity value attached to a given relationship between entities $e_j \in \mathcal{O}_S$ and $e_j \in \mathcal{O}_T$. In this case, $|\mathcal{O}_S|$=n and $|\mathcal{O}_T|$=m.

In the remaining sections of the paper, section 3 discusses related work, section 4 provides a detailed discussion of the proposed framework of validating mappings through tagging and finally section 5 provides the results and discussion of the evaluation of the entire proposed technique of validating mappings through tagging.

2. Related Work

The pay as you go technique as implemented by the current literature can be summarized as described in Algorithm 1. From Algorithm 1, a number of issues arise which help us to focus our review in this section. In line 2 of the algorithm, once a tool has established an initial alignment A^I, it has to select low quality mappings from A^I that require user's validation. Section 3.1, provides a review of the different techniques that tools employ to select low quality mappings from the alignment A^I. Line 3 of the algorithm allows users to evaluate the low quality mappings selected in line 2. This step raises a number of issues:

1. How is a user selected to participate in the mapping validation?
2. What are the design features that should be included in the user interface of a tool with an interactive mapping evaluation module?

Algorithm 1 Summary of the pay as you go technique.

INPUT Source ontology ($\mathcal{O}_s$), target ontologies ($\mathcal{O}_t$) and automatic matching tool M.
OUTPUT Quality alignment A.

1: Generate initial alignment A^I between $\mathcal{O}_s$ and $\mathcal{O}_t$using automatic mapping tool M.
2: Select a set of mappings L from A^I that are considered low quality.
3: User or Users query the set of low quality mappings L for validation.
4: Improve the quality of A^I based on the usersâĂŹ feedback.
5: Iterate the process from step 2 until mapping quality of A^I attains the highest possible quality (i.e. highest possible precision and recall)
6: output the final alignment A.

3. How do tools accept feedback from the user(s)?
4. In case multiple users are involved in the validation process, how is consensus reached in cases where their feedback on a given mapping disagree?

The first two issues are conclusively addressed in [16, 17]. Our review here is focused on the last two questions i.e. we review different techniques of accepting feedback from the user(s) and the techniques tools are employing to reach consensus in case multiple users disagree in their feedback (section 3.2). Finally, in line 4 the quality of the alignment A^I is improved based on the user's feedback. Section 3.3 discusses the different techniques that different researches have proposed to utilize user's feedback to improve the quality of the alignment A^I.

2.1. Selecting Low Quality Mappings for Validation

Manual curation of mappings such as pay as you go technique is time-consuming, especially if the input ontologies are large since they generate a large number of candidate mappings [18]. In order to make the process of user validation feasible i.e. appealing to the user(s) and scalable, only a small sample of initially generated mappings should be queried by the user for evaluation. Therefore, ontology matching tools performing interactive matching should implement techniques of selecting the most informative candidate mappings that after users' feedback, the feedback will maximize the improvement of the previous iteration's matching performance. In [13], the authors propose the use of an erroneous mapping as the most informative mapping since its correction and propagation of the correction in the similarity matrix guarantees further improvement of the previous mapping results. An erroneous mapping in this context is defined as a mapping between two entities from source and target ontology where automatic matcher establishes a certain relationship e.g. equivalence but according to some defined (possibly unknown) reference alignment, it is wrong. The problem of selecting low quality mappings for evaluation can be defined as:
Given an initial mapping set M, generated by automatic matcher(s), find a minimum set of mapping $M^u \subseteq M$ such that for each iteration $M \setminus M^u \rightarrow M^f$, where $M^f$is the final alignment. This definition assumes that all mappings in the set M^u are validated as wrong mappings. Three key techniques currently exist in literature for candidate mappings selection for user validation:

1. Probabilistic based techniques.
2. Similarity matrix based techniques.
3. Conservativity based techniques

2.1.1. Probabilistic Based Techniques

These techniques apply statistical methods to pick candidate mappings for evaluation. After establishing initial alignment using automatic matching tool, statistical methods are then applied to the alignment to pick sample mappings that user's feedback will be sought. This is applied in [11] where they apply simple random sampling as proposed in [19] to pick candidate mappings for user evaluation. A clear disadvantage of this technique is that it does not pay attention to selecting wrong mappings (informative mappings) therefore may result in users not providing significant improvement to the quality of mappings. However, the sample size of the selected mappings for user evaluation can be controlled to fit the user's needs.

2.1.2. Similarity Matrix Based Techniques

These techniques select low quality mappings from the final similarity matrix generated by a tool. Work in [12] presents a number of metrics that pick low quality mappings based on the conflicts that exists in the similarity matrix. They propose Cross Sum Quality (CSQ) and Similarity Score Definiteness (SSD) metrics which rank mappings in the increasing order quality. CSQ assigns higher quality score to a mapping which has less conflicts with other mappings in the similarity matrix while SSD measures how close the similarity value of mapping is to the similarity scoresâĂŹ upper and lower bounds [1,0]. Work in [13]

flags low quality mappings based on the confidence value assigned to a given mapping. The confidence value $Confidence(m(e_i, e_j))$ of a mapping $m(e_i, e_j)$ is computed by finding the difference between a given set threshold θ and the similarity value $S(e_i, e_j)$ of a mapping $m(e_i, e_j)$ as shown in equation 1. Mappings that have low confidence values are flagged as low quality mapping.
Generally, the similarity matrix based techniques of selecting low quality mappings ignore the underlying semantics that exists in the two input ontologies hence leave out a significant number of low quality mappings.

$$Confidence(m(e_i, e_j)) = |\theta - S(e_i, e_j)| \quad (1)$$

2.1.3. Conservativity Based Techniques

This technique flags low quality mappings when an alignment $\mathcal{A}$ generated by a tool is incoherent (see definition 3). The problem of incoherence in the alignment can arise either due to incompatibility between ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$ or as results of wrong mapping. Therefore, mappings that result in consistency violations become candidates for user evaluation. This technique is implemented by ALCOMO [8], AML [9] and LogMap [20]. The technique has the disadvantages that:

1. It selects a high number of correct mappings as wrong [10]. This is due to the fact that different ontologies can model a given concept differently hence a set of mappings that are correct can render an alignment $\mathcal{A}$ incoherent when compared to the input ontologies. Therefore, selecting mappings based on incoherence alone will flag a significant number of correct mappings as wrong [10].
2. It ignores a significant number of potential wrong mappings generated due to inconsistencies in the similarity matrix.
3. It ignores a significant number of potential wrong mappings contributed by instability in the final alignment.

2.2. User's Feedback

In order to get user's feedback, systems implement two key dimensions

1. Single user vs Multiple users
2. Controlled annotation vs Uncontrolled annotation.

2.2.1. Controlled Annotation vs Uncontrolled Annotation

In pay as you go, user(s) provide feedback in form of annotations. They annotate mappings based on the technique being applied by a system to verify correctness of a mapping. Formally, feedback can be viewed as a tuple $\langle M, a, u, t \rangle$ with M specifying the mapping, a the annotation provided by the user u, and t indicates the type of feedback [21]. Depending on the system, the set of annotations provided by the user can either be controlled or uncontrolled.
Controlled model is a setup where a user is allowed only to choose annotations from a fixed set of options such as evaluating if a mapping is correct or incorrect. Compared to uncontrolled scheme, this model is relatively fast since a user has no extra cognitive burden of coming up with his or her own tags, a process which may take some time. This scheme is implemented in [9, 12, 13, 20, 22, 23].
Uncontrolled model is where users provide the annotations in free form nature. Despite placing an extra cognitive burden on users, that they should come up with their own tags, it is supported by work in [14] who found out that users prefer tagging of concepts to assign them their own meaning rather than being restricted to narrow range of mapping terminology such as to answer questions like "corresponds to" and "similar to". Currently there is no tool that implements uncontrolled annotation model. To the best of our knowledge we believe the work presented in this paper is the first to exploit this technique.

2.2.2. Single User vs Multiple Users

Single user evaluation is where only a single user is allowed to validate a mapping. This has the advantage that it avoids dealing with diverse and sometimes disagreeing answers that are common when multiple users are engaged. However, it faces a number of criticisms. First, it is hard to find a single expert who is well versed will all subtopics contained in the two input ontologies. Consequently, restricting the ability of a single user to evaluate wide range of mappings from different subtopics that exist in the input ontologies. Secondly, in the single user mapping evaluation scheme, in case a user makes an error in evaluation, his or her evaluation is adopted since there are no other user to dispute. Some research such as [11] evaluate the reliability of a single user using Intra Observer Reliability (IaOR) as proposed in [24]. The single user technique is applied in [9, 13, 20, 22, 23, 25].

Multiple user evaluation is where multiple users provide feedback on a given mapping. With multiple users, conflicts in annotations of a given mapping are likely to arise. A conflicting feedback is a feedback where a mapping M is assigned two or more annotations with conflicting meaning [21]. Tools that employ this scheme have an extra task of establishing consensus among the conflicting users in a given mapping. Its reliability is however more assured as compared to single user model since a feedback is adopted only after majority of users agree on a given feedback. This scheme is implemented in [12] where they use simple majority vote to come to the final evaluation of a mapping from different answers provided by different users. [26] uses an alignment API to assess mappings provided by the crowd.

2.3. Improving Mappings Using User's Feedback

In pay as you go technique user's feedback is used to improve the mappings generated by the automatic mapping tool. The existing techniques in literature utilize user's feedback by either updating the similarity matrix or directly correcting the mappings. In [13], once users confirm a mapping to be either correct or wrong they use similarity flooding proposed in [27] to correct other related mapping therefore minimizing users effort. In [12] they update similarity matrix based on the users' response then execute the automatic selector algorithm to select new improved alignment from the adjusted similarity matrix. In [20], they directly remove mapping rejected by users from the final set of mapping.

3. TagMatch Interactive Framework

In this section, we present our proposed framework of evaluating mappings through tagging. Our discussion mainly follows the steps of the generic algorithm presented in Algorithm 1.

3.1. Step1: Executing Automatic Matcher

Given two input ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$, an automatic matching tool is first executed to establish an alignment $\mathcal{A}$ that contains mappings between entities of ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$.

3.2. Step2: Selecting Low Quality Mappings

After an automatic matching tool has generated an alignment $\mathcal{A}$ consisting of mappings between the entities of the source and target ontologies, the challenge becomes how to identify potential wrong mappings contained in the alignment $\mathcal{A}$ that will require users' validation. We advance the state of the art by introducing the concept of flagging low quality mappings based on unstable mappings that exists in $\mathcal{A}$. We also present different techniques of picking low quality mappings based on similarity matrix. We finally intersect the results of similarity matrix based technique and the ontology structure based techniques to establish mappings that have the highest likelihood of being wrong. In order to pick potential wrong mappings, we employ two key techniques:

1. Ontology structure dependent techniques.
2. Ontology structure independent techniques.

3.2.1. Ontology Structure Dependent Techniques (OSDT)

To flag low quality mapping using this technique, we rely on the structural relationships that are modeled in the two input ontologies to be matched and those generated based on merging the two input ontologies via equivalence correspondences established between their respective entities. Mappings that make the final alignment unstable or incoherent are flagged as low quality and therefore need usersâĂŹ validation.

Definition 1: Merged ontology. Given an alignment

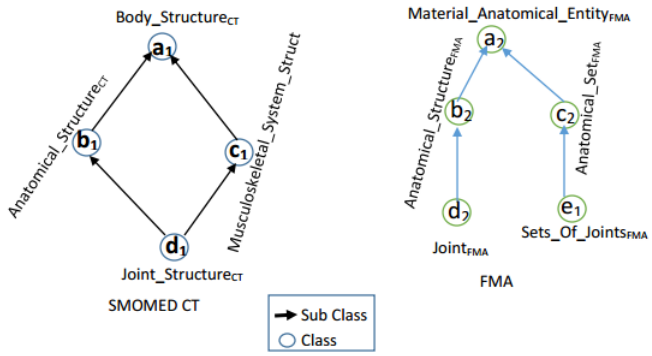

Fig. 1. Fragments of SNOMED CT and FMA ontologies

$\mathcal{A}$ between two ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$, the merged ontology $\mathcal{O}_1 \cup_{\mathcal{A}} \mathcal{O}_2$ of ontology $\mathcal{O}_1$ and $\mathcal{O}_2$ connected via $\mathcal{A}$ is defined as $\mathcal{O}_1 \cup \mathcal{O}_2 \cup EQ(c)$ where the function EQ converts equivalent correspondences between

$\mathcal{O}_1$ and $\mathcal{O}_2$ into equivalence axioms in $\mathcal{O}_1 \cup_{\mathcal{A}} \mathcal{O}_2$. Converting the equivalent correspondences between the two ontologies into equivalence axiom in merged ontology helps in achieving subsumption relationship between concepts of the independently modeled ontologies.

Definition 2: Unstable alignment. An alignment $\mathcal{A}$ between ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$ is regarded as unstable if there exists a pair of concepts $A, B \in \mathcal{O}_i$ with $i \in \{1, 2\}$ such that:

1. $\mathcal{O}_i \not\models A \sqsubseteq B$ and $\mathcal{O}_1 \cup_{\mathcal{A}} \mathcal{O}_2 \models A \sqsubseteq B$.
2. $\mathcal{O}_i \not\models A \equiv B$ and $\mathcal{O}_1 \cup_{\mathcal{A}} \mathcal{O}_2 \models A \equiv B$.

If a correspondence introduces new subsumption or equivalence relationship between the entities of a given ontology in the merged ontology that did not exist in the original ontology, the correspondence is flagged as unstable (unreliable) mapping. The mappings that are flagged as unstable include:

1. If $\mathcal{O}_1 \models X \sqsubseteq X'$ and $\mathcal{O}_2 \models \neg(Y \sqsubseteq Y')$ and any of the following mappings combination $m_1 = \langle X, Y, \equiv \rangle$ and $m_2 = \langle X', Y', \equiv \rangle$ or $m_1 = \langle X, Y', \equiv \rangle$ and $m_2 = \langle X', Y, \equiv \rangle$ exists in the final alignment, then flag the mappings m_1 and m_2 as unstable since they introduce new subsumption relationship $Y \sqsubseteq Y'$ or $Y' \sqsubseteq Y$ respectively in the merged ontology while the relationship did not exist in the ontology $\mathcal{O}_2$. The mappings m_1 and m_2 are added into a set T i.e set comprising of low quality mappings.
   Example 1: Consider the two ontologies in figure 1, $FMA \models \neg(d_2 \sqsubseteq c_2)$ while $CT \models d_1 \sqsubseteq b_1$. If the mappings $(d_1, d_2, \equiv)$ and $(b_1, c_2, \equiv)$ exist in the final alignment generated by a given tool, then a new subsumption relationship $d_2 \sqsubseteq c_2$ between concepts of FMA ontology is introduced in the merged ontology while it did not exist in the FMA ontology. The mappings which introduce this new relationship are flagged as unstable and hence are candidates for user validation.

2. If $X_1 \in \mathcal{O}_1$ and $\mathcal{O}_2 \models \neg(Y_1 \equiv Y_2)$ and the mappings $m_1 = (X_1, Y_1, \equiv)$ and $m_2 = (X_1, Y_2, \equiv)$ exist in the final alignment, then flag the mappings as unstable and update the set T with the mappings m_1 and m_2 if they are not already in T. The mappings m_1 and m_2 are flagged as unstable since they introduce new relationship $(Y_1, Y_2, \equiv)$ in the merged ontology that did not exist in the ontology $\mathcal{O}_2$.
   Example 2: In figure 1, $b_1 \in CT$ while $FMA \models \neg(b_2 \equiv c_2)$. If the mappings$(b1, b_2, \equiv)$ and $(b_1, c_2, \equiv)$ exist in the final alignment, the mappings should be flagged as unstable since they introduce new relationship $(b_2 \equiv c_2)$ in the merged ontology while the relationship is not entailed in FMA ontology.
3. If $\mathcal{O}_1 \models \exists R_1.\top \sqsubseteq X$ and $\mathcal{O}_2 \models \neg(\exists R_2.\top \sqsubseteq Y)$ and the mappings $m_1 = \langle R_1, R_2, \equiv \rangle$ and $m_2 = \langle X, Y, \equiv \rangle$ exist in the final alignment, then flag the mappings m_1 and m_2 as unstable and update the set T with the mappings m_1 and m_2 if they are not already in T. The mappings m_1 and m_2 are flagged as unstable since they introduce the relationship $\mathcal{O}_2 \models \exists R_2.\top \sqsubseteq Y$ that was not entailed in ontology $\mathcal{O}_2$.

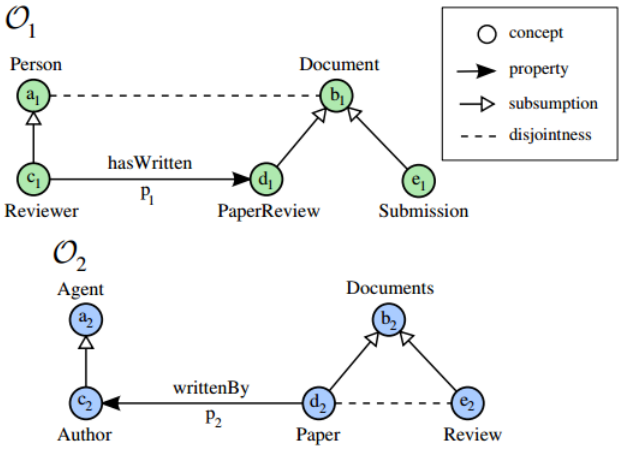

Fig. 2. Ontology fragment

Example 3: In figure 2, $\mathcal{O}_1 \models \exists$hasWritten.$\top \sqsubseteq$ Reviewer and $\mathcal{O}_2 \models \neg(\exists writtenBy.\top \sqsubseteq$ Author). If both the mappings (hasWritten, writtenBy, $\equiv$) and (Reviewer, Author, $\equiv$) exists in the final alignment, the mappings should be flagged as unstable since they introduce new relationship $\mathcal{O}_2 \models \exists$writtenBy.$\top \sqsubseteq$ Author in the merged ontology while the relationship is not entailed in the ontology $\mathcal{O}_2$ which is the source of the two entities.

Definition 3: Incoherent alignment. An alignment $\mathcal{A}$ between ontology $\mathcal{O}_1$ and $\mathcal{O}_2$ is regarded as incoherent if there exists a concept $C \in \mathcal{O}_i$ with i$\in \{1, 2\}$ such

that $\mathcal{O}_i \not\models C \sqsubseteq \bot$ and $\mathcal{O}_1 \cup_A \mathcal{O}_2 \models C \sqsubseteq \bot$.
Incoherence arises if the merged ontology has contradicting axioms. The mappings that are flagged as incoherent are:

1. If $\mathcal{O}_1 \models X \sqsubseteq \neg X'$ and $\mathcal{O}_2 \models Y \sqsubseteq Y'$ and any of the mapping combinations $m_1 = \langle X, Y, \equiv \rangle$ and $m_2 = \langle X', Y', \equiv \rangle$ or $m_1 = \langle X, Y', \equiv \rangle$ and $m_2 = \langle X', Y, \equiv \rangle$ exists in the final alignment, flag the mappings m_1 and m_2 as incoherent and update the set T with m_1 and m_2 if they are not already in T. The mappings m_1 and m_2 are flagged as incoherent since through them both $X \sqsubseteq \neg X'$ and $X \sqsubseteq X'$ will be entailed in the merged ontology hence the merged ontology will entail $X \sqsubseteq \bot$.
   Example 4: Consider the ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$ in figure 2. $\mathcal{O}_1 \models Person \sqsubseteq \neg Document$ and $\mathcal{O}_2 \models Review \sqsubseteq Documents$. If the mappings $(Document, Documents, \equiv)$ and $(Review, Person, \equiv)$ both exist in the final alignment, then they are flagged as incoherent since this implies both $Person \sqsubseteq Document$ and $Person \sqsubseteq \neg Document$ are entailed in the merged ontology hence $Person \sqsubseteq \bot$.
2. If $\mathcal{O}_1 \models \exists R_1.\top \sqsubseteq X$, $\mathcal{O}_2 \models \exists R_2.\top \sqsubseteq \neg Y$ and the mapping combination $m_1 = (R_1, R_2, \equiv)$ and $m_2 = (X, Y, \equiv)$ exist in the final alignment, flag the mappings m_1 and m_2 as incoherent and the set T with the mappings m_1 and m_2 if they are not already in T.

3.2.2. Ontology Structure Independent Techniques

Here we flag low quality mappings using similarity matrix. The similarity matrix used to flag low quality mapping is determined by how a tool executes its matchers. If a tool executes its matchers in sequential fashion such as in [28, 29], we use the final similarity matrix generated by the tool. Otherwise, if a tool executes its matchers in parallel such as in [30, 31], we use multiple similarity matrices generated by each matcher.

Final Similarity Matrix Based Technique (FSMBT)
If a tool executes its matchers in sequential fashion, we generate low quality mappings by identifying disagreeing mappings in the final similarity matrix.

Definition 4: Conflicting mappings from the final similarity matrix (D_1). Given the source ($\mathcal{O}_1$) and target ($\mathcal{O}_2$) ontologies and entities $e_i \in \mathcal{O}_1$ and $e_j, e_l \in \mathcal{O}_2$ where $e_j \neq e_l$, If a tool T which establishes 1:1 cardinality equivalence mappings generates a final similarity matrix S such that $s_{ij} = k$ and $s_{il} = w$ then the mappings $m_1(e_i, e_j)$ and $m_2(e_i, e_l)$ are conflicting if:

1. k and w ranging [0, 1] are greater than set selection threshold T and
2. $|k - w| \leqslant \epsilon$ i.e the difference between the similarity values of the two mappings is very small. The mappings participating in a disagreement are all flagged as low quality and placed in a set D_1.

Multiple Similarity Matrices based Technique (MSMBT)
When a tool executes its matchers in parallel, each matcher generates a similarity matrix. The multiple similarity matrices can then be aggregated into single matrix using any state of the art techniques such as weighted sum aggregation, average weighted aggregation, MAX aggregation, MIN aggregation etc. [32]. We flag low quality from the multiple similarity matrices generated by the matchers before they can be aggregated to a single final similarity matrix.

Definition 5: Conflicting mappings from multiple matchers (D_2). If the matchers M_1 and M_2 which establish 1:1 cardinality equivalence mappings are executed in parallel to match entities of source ($\mathcal{O}_1$) and target ($\mathcal{O}_2$) ontologies, then conflicting mappings can arise due to:

1. Inter-matcher conflict with regard to the mappings m_1 and m_2 i.e. the matcher M_1 establishes the mapping $m(e_i, e_j, \equiv)$ and matcher M_2 establishes the mapping $m(e_i, e''_j, \equiv)$ where the entities $e_i \in \mathcal{O}_1$ and $e_j, e''_j \in \mathcal{O}_2$ and $e_j \neq e''_j$ then the two marchers are conflicting with regard to the mappings.
2. Intra-matcher conflict. Intra-matcher conflicting mappings are flagged based on definition 4.
   All the unique conflicting mappings flagged by the the inter-matcher and intra-matcher techniques are stored in a set D_2.

3.3. Step 3: Ranking of Low Quality Mappings

Since users are involved in the mapping validation, there is need to anticipate user fatigue, therefore the most informative mappings should be displayed to the users first such that in case users abandon the validation process at any given stage, we are able to capture the best feedback from already validated mappings. To achieve this, we categorize the low quality mappings into three key groups:

1. **Most informative mappings.** These are a set of mappings which have the highest likelihood that they are wrong hence most informative when val-

idated. To establish the most informative set of mappings, we find the intersection of mappings in the sets T and D_i i.e if the tool uses sequential matcher aggregation, the most informative mapping set $\mathcal{M}_{\mathcal{I}}$ is:

$$\mathcal{M}_{\mathcal{I}} = D_1 \cap T \tag{2}$$

For a case where a tool uses parallel matcher aggregation, the most informative mapping set $\mathcal{M}^*_{\mathcal{I}}$ is:

$$\mathcal{M}^*_{\mathcal{I}} = D_2 \cap T \tag{3}$$

This category is basically composed of mappings where the ontology structure independent and ontology structure dependent techniques of picking low quality mappings agree that a given set of mappings are of low quality.

2. **Averagely informative mappings.** From the experiments described in section 5.2, we observed that the ontology structure dependent techniques of picking low quality mapping was more reliable than the ontology structure independent techniques. Hence to compute averagely informative mapping $\mathcal{M}_{\mathcal{A}}$ we compute the set of differential mappings in the set T with respect to D_i.

$$\mathcal{M}_{\mathcal{A}} = T \setminus D_1 \text{ or } \mathcal{M}^*_{\mathcal{A}} = T \setminus D_2 \tag{4}$$

3. **Informative mappings.** This is the lowest ranked informative mappings. The mappings are computed as set of differential mappings in the set D_i with respect to T.

$$\mathcal{M} = D_1 \setminus T \text{ or } \mathcal{M}^* = D_2 \setminus T \tag{5}$$

3.4. Step 4: Users' Feedback

After identifying low quality mappings, we engage the users to perform mappings validation. Previous works in [11–13] employ a single user to validate mappings using terms from a controlled vocabulary set such as "Correct" and "Incorrect". This validation scheme has the advantages that:

1. A user has no cognitive burden of coming up with his or her own tags.
2. Employing a single user avoids the challenge of dealing with conflicting feedback on a given mapping which is likely to arise when multiple users are involved in the validation process.

However, the scheme does not solve the underlying issue of why an entity's meaning was not correctly disambiguated. Hence when the same ontology is matched with a new ontology, the same entities are likely to be involved in low quality mapping. Furthermore, the use of single user is error prone since it is unlikely that even an experienced user will be familiar with all subtopics contained in an ontology. In our case, we take the view that a concept is involved in unreliable mapping due to lack of enough information modeled in the ontology to explicitly disambiguate its meaning. We therefore employ multiple users to validate the mappings by tagging the concepts that are involved in the low quality mappings. Quality tags associated with a given concept are embedded in the ontology as concept's label annotations. Through this we are able to enrich the annotations of a concept. This will increase the ability of a tool to disambiguate the meaning of an entity in future matching process of the entity since most ontology matching tools rely on entities' annotations to establish their meaning. Users' tags are also used to validate the correctness of a mapping. When a pair of concepts of a mapping share tags which have the same meaning, the mapping is judged to have been validated by users as correct. Hence we are able to achieve both ontology meta data enrichment as well as mapping validation.

Tagging Scheme Challenges

When multiple users take part in the interactive process such as tagging, a number of challenges arise which have to be addressed in order to get maximum benefits of the tagging scheme. The challenges are:

1. **Conflicting tags.** Tags conflict when a given concept C is assigned two or more tags with different meaning. This arises due to different interpretation of a concept's meaning by different users. This problem is exacerbated in tagging systems, since there is no fixed or controlled vocabularies that guide tag selection.
2. **Synonym divergence.** This problem arises when users use different syntactic terms which have the same meaning e.g. cell-phone and mobile-phone.
3. **Syntactic divergence.** This problem arises when different users use different forms of a word to represent the same tag e.g. blog, blogging.
4. **Low tag frequency problem.** This problem arises when users fail to tag a concept, therefore, making the concept to lack enough tags to generate meaningful information.

Addressing Challenges of The Tagging Scheme
We use two key strategies to minimize the challenges associated with the tagging scheme:
Tag Suggestion
Tag suggestion, the automated process of suggesting useful and informative tags to an emerging object based on historical information [33] is one of the ways that has been suggested to deal with synonymy and syntactic divergence problem [33]. By providing consistent tag suggestion to the users, we provide a way of converging a tag corpus of a concept to popular tags, therefore, helping to reduce the synonym and syntactic divergence problem. Tag suggestion also provides the much needed cognitive support to the users. Most tagging systems use weights of already used tags (historical tags) to suggest potential tags of a given concept to the user. This method has the disadvantage that it works best when a given concept has already been tagged many times and by this time a significant amount of syntactic and synonym problem would have been introduced into the tag corpus hence reducing the benefits of tag suggestion. For an ontology, a unique scenario emerges where the entities are sometimes already annotated with labels and comments and entity-relationship structure is well known. Label annotation of an entity mostly has an alternative name of the entity while comment annotation gives a brief descriptive sentence which is helpful in getting more meaning of the entity in the ontology. We use the annotations of an entity as an initial tag suggestion corpus of the entity in the absence of user-supplied tags. This helps us to tackle the syntactic and synonym problem from the beginning of the tagging process. It is important to note that annotations of an entity are used to suggests tags to the users before the entity is tagged multiple times by the users. As users begin tagging the entity, their tags are also included as candidates for tag suggestion by the tag suggestion algorithm.
Tag Suggestion Algorithm
Processing annotation corpuses. Given an entity e_i in the most informative mapping set ($\mathcal{M}_{\mathcal{I}}$ or $\mathcal{M}^*_{\mathcal{I}}$) or in the averagely informative mapping set ($\mathcal{M}_{\mathcal{A}}$ or $\mathcal{M}^*_{\mathcal{A}}$) or in the informative mapping set ($\mathcal{M}$ or $\mathcal{M}^*$), we extract its annotation (Comments and Labels). We then perform pre-processing of its annotations by removing all the special symbols such as "@","#" "*","!" and stop words such as "and", "of", etc. We then create a vector $v(e_i) = \{w_1, w_2, \cdots, w_n\}$ in an n-dimensional space, where each w_i represents the weight of i^{th} distinct word that appear in annotation corpus of the entity e_i. To compute the weight w_i of the i^{th} distinct word in the annotation corpus of an entity e_i, we make the entity e_i to mimic a document and compute word's weight w_i as its TF-IDF in the annotation corpus of an entity as shown in equation 6.

$$w_i = n_{ij} \times log\frac{N}{N_i} \tag{6}$$

Where n_{ij} is the count of number of times a given word (term) t_i appears in the annotation corpus of an entity e_i, N_i is the total number of occurrences of word (term) t_i in all entity corpuses of the ontology $\mathcal{O}$ and N is total the number of entity corpuses of $\mathcal{O}$.
Missing annotation problem. Within an OWL ontology, entities can be modeled without annotations hence limiting our tag suggestion algorithm that uses entity's annotation as its initial tag suggestion corpus. To solve this, we exploit the structural relationships that exist within an ontology to ensure that entities which are closely related contribute their annotation to their close relatives if their relatives have no annotations either because of lack of tagging or missing annotation during ontology modeling. The key relationships that we exploit are:

1. **rdfs:subClassOf or rdfs:subPropertysOf** i.e. $X_2 \sqsubseteq X_1$ (an entity X_1 subsumes an entity X_2). In the OWL ontology, the child inherits the attributes of the parent in the superEntity-subEntity relationship. Therefore, if an entity X_1 has annotations and its child X_2 has none, the annotations of X_1 is also assigned to the entity X_2. However, every distinct word w_i in the inherited annotation is penalized by assigning it a weight of 0.75 of the original weight (computed as in equation 8). We only assign annotations of a parent to its direct children i.e no inferencing is performed.
2. **owl:equivalentClass or owl:equivalentProperty** i.e. $X_2 \equiv X_1$ (entities X_1 and X_2 are equivalent). If an entity X_1 has no annotations and X_2 has annotations, the annotations of X_1 is also assigned to X_2 with each distinct word having equal weight as the original weight.
3. **hasExactSynonym** and **exact_synonym.** If X_2 is an exact synonym of X_1 and X_2 has no annotations, then annotations of X_1 is assigned to X_2 with each distinct word having equal weight as the original weight.
4. **other synonyms** such as hasRelatedSynonym, broad_synonym etc. are treated as owl:hasExactSynonym however the weight of each distinct word from

the acquired annotations is assigned a weight of 0.8 of its original weight.

To suggest potential tags of an entity, we adopt the information gain which is always used to classify documents. It measures how good a word is in discriminating between classes. We use it to rank the words that exist in the annotation corpus of an entity e_i in the decreasing order of how important a given word is in classifying (discriminating) the entity in an ontology $\mathcal{O}$. We therefore compute the information gain of all unique words that exist in the annotation corpus of the entity e_i and select those with top k information gain values as its potential tags. To compute information gain of a word w in an entity e_i in an ontology $\mathcal{O}$ we use equation 7.

$$IG(w) = -\sum_{e_i \in \mathcal{O}} P(e_i)\log P(e_i) + P(w)\sum_{e_i \in \mathcal{O}} P(e_i|w)\log P(e_i|w) + P(\bar{w})\sum_{e_i \in \mathcal{O}} P(e_i|\bar{w})\log P(e_i|\bar{w}) \tag{7}$$

Where $P(e_i)$ is the probability of an entity (e_i) in the ontology $\mathcal{O}$. $P(e_i|w)$is the probability of an entity in (e_i) from all the entities that contain the word w, $P(e_i|\bar{w})$ is the probability of an entity (e_i) from all entities that do not contain word w. The probability $P(e_i)$ is computed according to equation 8.

$$P(e_i) = \frac{C(Sy(e_i)) + C(Eq(e_i)) + 1}{|\mathcal{O}|} \tag{8}$$

where $C(Sy(e_i))$ is the count of all synonyms of entity e_i, $C(Eq(e_i))$ is the count of all equivalences of the entity e_i and $|\mathcal{O}|$ is the count of all entities in the ontology $\mathcal{O}$. The value 1 is added in the numerator to account for the entity e_i.
Note that at the beginning of tagging process the only annotations that exist are those which were modeled during ontology creation. However, as users start tagging the entities, selected quality tags are embedded on the entities as label annotation therefore adding to the size of the tag suggestion corpus.

Auto-Completing of Tags
In order to further reduce syntactic divergence and improve the usability of the tagging system, the tagging system is able to auto complete tags as they are being typed by the user.

3.4.1. Tagging Process

Order of mapping validation. Users validate mappings that are contained in the most informative mapping set ($\mathcal{M}_{\mathcal{I}}$ or $\mathcal{M}^*_{\mathcal{I}}$), the averagely informative mapping set ($\mathcal{M}_{\mathcal{A}}$ or $\mathcal{M}^*_{\mathcal{A}}$) and the informative mapping set ($\mathcal{M}$ or $\mathcal{M}^*$). The mappings are validated in the decreasing order of informativeness i.e mappings in the most informative set are validated first, followed by averagely informative set and informative set respectively. The mappings of a given set are validated based on the decreasing order of their similarity values.
Users' mapping validation process. A user of the system is allowed to create a profile where after successful login, he or she is able to initiate the process of mappings validation. A pair of concepts of a mapping to be validated is then displayed to the user side by side so that he or she can assign alternative labels (tag) to them. Before a user can tag a concept, the top k most popular tags of the concept computed according to equation 7 are suggested to the user for possible adoption. A user can either adopt any of the suggested tags or assign a new tag to the concept. If a user feels that he or she cannot provide a tag to any concept, he or she is allowed to skip the concept. The skipped concepts can be requested at any stage of the tagging process. Once a user has tagged a concept, he or she can save it. Future editing of a tag assigned to the saved concept is allowed.

3.5. step 5: Processing Users' Tags

Once the tagging process is complete, we need to evaluate if users have accepted or rejected a mapping based on their supplied tags. A mapping $m_i(e_i, e_i)$ between the entities e_i and e_j is accepted and rejected based on the condition below.

$$\begin{cases} m_i(a,b) \text{ is Correct}, & \text{if } |\text{tags}(e_i) \cap \text{tags}(e_i)| \geqslant \theta \\ m_i(a,b) \text{ is Wrong}, & \text{otherwise} \end{cases}$$

where $tags(e_i)$ is a set containing all tags used to tag an entity $tag(e_i)$ and θ is a set threshold.
The idea here is that if two concepts of a mapping share many similar tags, then users have approved the mapping. The major task therefore is to establish similarities between users supplied tags. To establish similar tags, we use spectral clustering [34] technique. Spectral clustering is one of the most successful graph and matrix partitioning heuristics [35]. It has been applied in various domains such as solving sparse linear systems [36], image segmentation [37] etc. It ex-

ploits the eigenvectors of the Laplacian matrix to partition a graph. We therefore exploit a spectral clustering technique to place the tags into different clusters such that similar tags are clustered together. Once the tags have been clustered based on their similarities, the entities of the source and target ontologies that have been tagged are also clustered by being assigned to a tag cluster where most of its tags appear, this becomes its resident cluster. If two concepts originally evaluated by an automatic matching tool to be mapped are allocated to two separate tag clusters, it automatically means that users have rejected the mapping. We therefore adjust the similarity value between these entities in the similarity matrix to zero. If the two concepts of a mapping are assigned to the same cluster, we further compute their similarity value from the tags perspective. To cluster the tags and concepts into k clusters, we execute three key steps:

1. Construct a graph G with its nodes being the tags.
2. Partition graph G into k clusters.
3. Assign entities to the k clusters.

3.5.1. Graph Construction Process

From the set of all tags $T = \{t_1, t_2, \cdots, t_n\}$ supplied by the users, we construct an undirected graph G with V=V(G) and E=E(G) being the vertex and edge set respectively. The tags are modeled as the vertices of the graph G and an edge is created between a pair of tags (t_i, t_j). The graph G is weighted by the weight function

$$w : E(G) \mapsto \mathbb{R}^+$$

The weight function assigns a non-negative real weight w_{ij} to an edge between a pair (t_i,t_j) of vertices. The weight w_{ij} satisfies the following properties:

1. $w_{ij} > 0$ if $(t_i, t_j) \in V(G)$.
2. $w_{ij} = 0$ if $(t_i, t_j) \notin V(G)$.
3. $w_{ij} = w_{ji}$

Property 3 is due to undirected nature of graph G. The weight w_{ij} represents tag relatedness between tag t_i and t_j. If $w_{ij} = 0$, it means the tags t_i and t_j are not related at all. To compute the weight w_{ij} of the edge (t_i, t_j) we rely on the work proposed by [15, 38]. To establish the how strong a pair of tags are related, we begin by defining a folksonomy as $F \subseteq U \times C \times T$, where U represents a set of all users who participated in tagging process, C is a set of all concepts that have been tagged and T is a set of all tags used to tag concepts. The folksonomy F is naturally represented as tripartite hypergraph $H(G) = \langle V, E \rangle$ where $V = A \cup C \cup T$ and $E = \{(u, c, t) | (u, c, t) \in G\}$ [15, 38]. The tripartite graph $H(G)$ can then be folded into three bipartite graphs i.e. graphs representing relationships between Concepts and Tags $G(TC)$, Concepts and Users $G(CU)$ and Users and Tags $G(TU)$ [15]. In our case, we are interested in the graphs $G(TC)$ and $G(TU)$ since they help us in establishing the closeness that exist between a pair of tags. In the graph $G(CT)$, a concept $c \in C$ is related to tag $t \in T$ if there is at least a user $u \in U$ who has used a tag t to tag the concept c. From the bipartite graph $G(CT)$, we form a matrix $A = a_{ij}$ where $a_{ij} = 1$ if a concept c_i is related to a tag t_j, otherwise $a_{ij} = 0$. From matrix A, we again create a one mode matrix $S = \{s_{ij}\}$ where $s_{ij} = \sum_{x=1}^{|C|} a_{xi} a_{xj}$ [15, 38]. This matrix represents the relationship between the tags based on shared concepts. Each s_{ij} represents the co-affiliation between the tags t_i and t_j based on common concepts between them. In the second bipartite graph $G(TU)$, a tag $t \in T$ is related to a user $u \in U$ if there exist at least one concept $c \in C$ which has been tagged by the user u using the tag t. From the graph $G(TU)$, we create a bipartite matrix $B = b_{ij}$ where $b_{ij} = 1$ if a user u_i is related to a concept t_j, otherwise $b_{ij} = 0$. From the matrix B, we create a one mode matrix $K = \{l_{ij}\}$ where $l_{ij} = \sum_{i=1}^{|U|} b_{xi} b_{xj}$. This matrix represents the relationship between the tags t_i and t_j based on shared users. Each l_{ij} represents the co-affiliation between the tags t_i and t_j based on shared users. The total co-affiliation C_{ij} between tag t_i and t_j is computed as

$$C_{ij} = l_{ij} + s_{ij}$$

i.e the total co-affiliation C_{ij} is the summation of the co-affiliation due to common users and that based on common concepts between the tag t_i and tags t_j. After computing co-affiliation values between pairs of tags, we form a vector $v_i \in R^T$ for each tag t_i. Each co-affiliation value C_{ij} between the tags t_i, t_j defines a dimension in Euclidean space, where $1 < j \leqslant |T|$ i.e. $v_i = \{C_{i1}, C_{i2} \cdots, C_{i|T|}\}$. The weight w_{ij} between the tags t_i and t_j is computed using cosine similarity [39] which has been demonstrated in [40] as an effective way of establishing tag relatedness.

$$w_{ij} = \frac{v_i \cdot v_j}{(||v_i|| \cdot ||v_j||)} \tag{9}$$

In a case where users have provided a total of n tags, for each tag we will need to perform $(n-1)$ pairwise

computations of equation 9. This results into a total of $(n^2 - n)$ comparisons. This is not effective in terms of space and time complexity. To reduce the number of pairwise comparisons, we set a threshold ϕ such that if the relatedness between the tags t_i and t_j is below ϕ, their relatedness is set to 0 and we apply the following rule: If the tag relatedness between t_i and t_j computed according to equation 9 is above a set threshold ϕ, and that between t_j and t_k is below the threshold ϕ, we skip the pairwise comparison of t_i and t_k since this computation will also result in low relatedness value. By this rule we are able to significantly reduce the number of pairwise computations hence speeding up the graph creation process.

3.5.2. K-Way Graph Partitioning of Tags

A k-way graph partitioning entails partitioning of graph G into a set of clusters (subsets of V) $P^k = \{C_1, C_2, \cdots, C_k\}$ such that each $t_i \in V$ is a member of exactly one C_i, with $1 \leqslant i \leqslant k$. The goal of partitioning is such that for each partition(cluster) $C_i = \{t_1, t_2, \cdots, t_t\}$ where $t < n$ and $n = |T|$, the weighted sum of squared distances between the points of a partition is minimized i.e. the relatedness between tags of a given partition is maximized. Therefore, our task becomes minimizing equation 10 of a given partition.

$$z = \frac{1}{2}\sum_{i=1}^{t}\sum_{j=1}^{t}(t_i - t_j)^2 w_{ij} \tag{10}$$

The optimal solution of equation 10 is NP-hard, therefore, to solve it, we apply spectral partitioning as proposed in [41]. To setup spectral partitioning framework, we create a weighted adjacency matrix $W = (w_{ij})_{i,j=1,2...,n}$ from the similarity graph G. The degree of a vertex $t_i \in V$ is defined as

$$d_i = \sum_{j=1}^{n} w_{ij} \tag{11}$$

The degree of vertex t_i is basically the sum that runs over all vertices adjacent to it. We define degree matrix D as the diagonal matrix with the degrees $\{d_1, d_2, ..., d_n\}$ on the diagonal. In order to perform spectral clustering of the points(tags) of the graph G, we employ the graph Laplacian matrix which is the main tool used for clustering [42]. The unnormalized graph Laplacian is defined as $L = D - W$. L has a number of desirable properties [43] which include:

1. For any vector $v = \{v_1, v_2, \cdots, v_n\} \in R^n$

$$v^T L v = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(v_i - v_j)^2 w_{ij} \text{ [42, 44]}$$

   where v^T is the transpose of vector v.
2. L has n non-negative real valued eigenvalues $0 = \lambda_1 \leqslant \lambda_2 \cdots \leqslant \lambda_n$.
3. The eigenvectors of L form a basis in n-dimensional space since they are all mutually orthogonal.

We exploit these properties of L to partition the graph G. Using property 1, we need to establish a vector v such that equation 12 is minimized.

$$v^T L v = \frac{1}{2}\sum_{i=1}^{t}\sum_{j=1}^{t}(t_i - t_j)^2 w_{ij} \tag{12}$$

Hall [44] demonstrated that the eigenvectors of the matrix L will minimize $v^T Lv$, with the eigenvector associated with second lowest eigenvalue being the optimum minimum for $v^T Lv$. Therefore, the eigenvectors of matrix L of graph G can be used to solve equation 12. In this case, the eigenvectors of the graph Laplacian matrix L are viewed as a solving relaxation of an NP-hard discrete graph partitioning problem. Different researches on spectral partitioning have explored the question on which eigenvectors can be used to achieve the best partitioning results. Some work such as [36, 45] proposed partitioning of the graph based on eigenvector associated with the second eigenvalue of L. This technique recursively partitions a graph into two sections until the desired number of partitions is achieved. However, work in [34, 46] experimentally demonstrated that the more eigenvectors used and directly partitioning the graph G into k clusters the better. This work exploits the use of k eigenvectors associated with k largest eigenvalues to directly partition the graph G into k partitions (clusters). Algorithm 2 summarizes our graph partitioning framework. Algorithm 2 has some desirable properties:

1. It focuses on producing partitions where similar tags are clustered together.
2. It has low computation cost.
3. The number of similar tags in a partition can be varied by varying the value of k.

Algorithm 2 Graph partitioning algorithm

Input Similarity Graph G= (V, E).
Output partitions P_i where $i = 1, 2, \cdots, k$.

1: Given a graph $G = (\{t_1, t_2, ..., t_n\}, E)$
2: Form an adjacency matrix $W = (w_{ij}) i \neq j$ where w_{ij} is computed according to equation 9 and $w_{ii} = 0$.
3: Define D to be a diagonal matrix whose (i,i) element is the sum of the i^{th} row of W i.e degree $d_i(equation11)$.
4: Construct the unnormalized Laplacian matrix L=D-W
5: Find $\{\vec{v_1}, \vec{v_2}, ..., \vec{v_k}\}$ the k largest eigenvectors of L (chosen to be orthogonal to each other in case of repeated eigenvalues).
6: Form a matrix $X = \{\vec{v_1}, \vec{v_2}, ..., \vec{v_k}\} \in R^{n \times k}$ by arranging the vectors into columns.
7: Form a matrix Y from X by normalizing each row of X according to $y_{ij} = \frac{x_{ij}}{\sum_j (x_{ij}^2)^{\frac{1}{2}}}$
8: Treating each row of Y as a point in R^k, cluster them into clusters using K-means algorithm.
9: Assign a tag t_i to a cluster j if and only if row i of the matrix Y was assigned to cluster j
10: Each cluster i is a partition P_i where $i = 1, 2, \cdots, k$.

3.6. Assigning Concepts to Partitions

Once the tags have been clustered into different partitions based on their similarities, we also cluster the tagged concepts(entities). An entity e_i of a mapping $m(e_i, e_j)$ which was under validation is placed in a tag cluster C_i if there is no other tag cluster C_j such that $|tags(e_i) \cap C_j| > |tags(e_i) \cap C_i|$, where $tags(e_i)$ is a set of all tags used to tag the entity e_j.

3.7. Mapping Validation

If a pair of entities e_i and e_j of a mapping $m(e_i, e_j)$ are placed in different clusters, the mapping is rejected. If the entities are placed in the same tag cluster, we further compute their similarities using tags. We compute the similarity of entities e_i and e_j of the mapping $m(e_i, e_j)$ using the similarity computation proposed in [47]. Dissimilarity between entities e_i and e_j is computed as the cardinality of the set of differential tags of e_i with respect to e_j and the set of of differential tags of e_j with respect to e_i as shown in equation 13.

$$|\phi(e_i) \setminus \phi(e_j)| + |\phi(e_j) \setminus \phi(e_i)| \quad (13)$$

In order to enable accurate comparison between the dissimilarity computed for entities e_i and e_i, we normalize equation 13 by dividing it by total number of tags between the two entities. To further improve the accuracy, we introduce the logarithm informed by the discussion in [47] who established that semantic features are better evaluated in non-linear fashion rather than linear. The logarithm is computed by adding 1 to the ratio to avoid infinite values for equivalent concepts. The final equation is shown in 14.

$$DIS(e_i, e_j) = \log_2(1 + \frac{|\phi(e_i) \setminus \phi(e_j)| + |\phi(e_j) \setminus \phi(e_i)|}{|\phi(e_i) \setminus \phi(e_j)| + |\phi(e_j) \setminus \phi(e_i)| + |\phi(e_j) \cap \phi(e_i)|}) \quad (14)$$

Similarity between the two concepts e_i and e_j is computed as shown in equation 15.

$$SIM(e_i, e_j) = 1 - DIS(e_i, e_j) \quad (15)$$

If two entities e_i and e_i are found to be similar based on a set threshold, their similarity value in the similarity matrix is adjusted to 1 and all other values in the similarity matrix where e_i and e_i participate is adjusted to zero. If the similarity value is below the set threshold the similarity value in the similarity matrix is adjusted to this value. After processing all validated mappings, the selector algorithm of the automatic tool is applied to pick the the final alignment from the adjusted similarity matrix. The process can then be iterated from step 2 this time only involving the adjusted similarity matrix and the ontology independent technique for picking low quality mappings. This is repeated until a given set number of iteration t is achieved or the number of low quality mappings fall below a set number n and the final alignment is selected using the selector algorithm of AML.

4. Evaluation and Discussion

We conducted four key evaluations with a goal to answer the following questions:

1. How helpful is the tag suggestion algorithm to the users during tagging?
2. How accurate are the techniques described in section 4.2 in picking wrong mappings?
3. Does mapping validation through tagging lead to a significant improvement in quality of the final alignment?

4. Does enrichment of an entity's annotations in an ontology lead to an improvement in disambiguating its meaning?

4.1. Tag Suggestion Algorithm Evaluation

To answer the first question, we used a small sized ontology, ekaw.owl downloaded from Ontology Alignment Evaluation Initiative (OAEI) 2015 conference track[1]. The ontology contains terms that deal with conference organization. It has 73 classes and 33 object properties. To evaluate the "goodness" of the tag suggestion algorithm in generating helpful tags to users, we employed services of 14 masters' students who had at least submitted a research paper to a journal, therefore, were familiar with the paper submission terms as used in journals. The students were required to perform two key tasks, first was to generate handwritten tags for the classes contained in ekaw.owl. The second task to the users is where they were presented with system generated tags and were required to answer two key questions:

1. If they would prefer the tag generated by the system to their own tags.
2. They were also tasked to rate the system generated tags on their relevance based on our developed subjective scale of [0-10], with 0 being not relevant at all and 10 being completely relevant.

To rate the tag suggestion algorithm performance, we used three metrics i.e. precision, coverage and novelty. We defined precision as number of tags with an average score of 5 and above according to developed scale divided by total number of tags generated by the system as shown in equation 16. Tag coverage is defined as the number of common tags between user generated tags and system generated tags divided by the number of user generated tags, this is shown in equation 17. Finally, we also evaluated how good the system was in generating tags the user has not thought of but were very accurate representation of an entity i.e. the novelty of the tag generating system as shown in equation 18. The results of evaluation are shown in table 1.

$$Precision = \frac{C(T_{average} \geqslant 5)}{T_{totals}} \tag{16}$$

where $C(T_{average} \geqslant 5)$ is the total count of all tags generated by the algorithm that were judged by students as having relevance of 5 and above based on our developed scale and T_{totals} is the total number of tags generated by the system.

$$Coverage = \frac{UserTag \cap SystemTag}{UserTag} \tag{17}$$

Where UserTag is the tags suggested by the users and SystemTag is the tag suggested by the system.

$$novelty = \frac{TagAdopted}{SystemTag} \tag{18}$$

where TagAdopted is the number tags users have adopted over their own tags and SystemTag is the total number of tags generated by the system.

Table 1
Performance of tag suggestion algorithm.

Method	Precision	Coverage	Novelty
Tag suggestion Algorithm	59%	56%	30%

The tagging algorithm attains a precision of 59% which can be considered a good cognitive support to the user during the tagging process. The average precision is augmented by the 30% novelty of the tag suggestion algorithm. The algorithm also attains a coverage of 56% which depicts that on average tags suggested by the algorithm have 56% acceptance rate by the users.

4.2. Evaluating the Performance of Low Quality Mapping Picking Techniques

This section answers the second question: How accurate are the proposed techniques described in section 4.2 in picking wrong mappings? To perform this part of evaluation, we used data from OAEI 2016 anatomy track. The anatomy track is based on matching human.owl which has 3304 classes with mouse.owl which has 2743 classes. To establish initial mappings of human.owl and mouse.owl, we used SBOMT [48] executed on an Intel Core i5-5200 CPU @ 2.20GHz with 4 GB RAM laptop. SBOMT was selected because it had the following desirable features that were relevant to our evaluation:

1. It does not perform automatic repair of wrong mappings hence we were able to use our proposed techniques to pick potential wrong mappings from alignment generated by SBOMT.

2. It implements multiple independent matchers i.e. string based matcher, semantic based matcher, taxonomy based matcher hence we were able to exploit similarity matrices generated by each matcher to evaluate our Multiple Similarity Matrices Based Technique (MSMBT) as described in section 4.2.2.
3. We were also able to use its final similarity matrix to evaluate the Final Similarity Matrix Based Technique (FSMBT) described in section 4.2.2

A set $\mathcal{W}_{SBOMT}$ which is composed of wrong mappings was created by comparing the alignment $\mathcal{A}_{SBOMT}$ generated by SBOMT and $\mathcal{A}_{REF}$ the OAEI reference alignment provided for Anatomy track. Our task therefore was to evaluate how accurate the proposed techniques of selecting low quality mappings are in picking the wrong mappings in the set $\mathcal{W}_{SBOMT}$. The results of Ontology Structure Dependent Technique (OSDT), Multiple Similarity Matrices Based Technique (MSMBT), Final similarity Matrix Based Technique (FSMBT) are shown in table 2. We also compared the performances of these low quality mapping selection techniques against those of Cross Sum Quality (CSQ) and Similarity Score Definiteness (SSD)metrics as proposed in [12]. SBOMT generated a total of 258 mappings that were wrong i.e. the set $\mathcal{W}_{SBOMT}$ contained 258 mappings. These were the mappings we expected a given low quality mapping selecting technique to pick.

Table 2

Performance of different techniques of selecting low quality mapping.

Technique	Total Mappings Selected	True Positives	True Negatives	Precision	Recall
OSDT	253	165	162	0.6522	0.6395
FSMBT	156	70	86	0.4487	0.2713
MSMBT	262	147	115	0.5611	0.5698
(CSQ)	142	62	80	0.5864	0.2403
(SSD)	122	66	56	0.5409	0.2558

From the results in table 2, Ontology Structure Dependent Technique(OSDT) leads in precision followed by Cross Sum Quality (CSQ) and Multiple Similarity Matrices Based Technique (MSMBT) respectively. OSDT also leads in terms of recall followed by MSMBT. CSQ, SSD and FSMBT pick a total of 62, 66 and 70 wrong mappings respectively out of the expected value of 258. Since they leave out a large number of wrong mappings, the three methods perform poorly in recall. On the overall, the results in table 2 show that ontology structure based technique is able to identify more wrong mappings as compared to techniques that rely on the similarity matrix. Based on the performance of MSMBT as compared to those of CSQ, FSMBT and SSD, using multiple similarity matrices in flagging low quality mappings results in better precision and recall of the flagged mappings as compared to only using a single similarity matrix.

4.2.1. Ranking Low Quality Mappings

In an interactive framework such as user validation, user fatigue should be anticipated, therefore we should increase the probability of showing wrong mappings to the user such that in case he or she abandons the validation process halfway, we are to get informative feedback from already provided answers. We therefore use techniques discussed in section 5 to generate mappings set that favor precision over recall.

Table 3 shows the performance of the most informative mapping ($\mathcal{M}_{\mathcal{I}}$,$\mathcal{M}^*_{\mathcal{I}}$), averagely informative mappings ($\mathcal{M}_{\mathcal{A}}, \mathcal{M}^*_{\mathcal{A}}$) and Informative mappings ($\mathcal{M}, \mathcal{M}^*$). The most informative mapping ($\mathcal{M}_{\mathcal{I}}$ and $\mathcal{M}^*_{\mathcal{I}}$) posts the leading precision values followed by averagely informative mappings($\mathcal{M}_{\mathcal{A}}$ and $\mathcal{M}^*_{\mathcal{A}}$). If $\mathcal{M}^*_{\mathcal{I}}$ is displayed to the user first, there is a 84.47% likelihood that the user will validate a wrong mapping. If a user validates all mappings in $\mathcal{M}^*_{\mathcal{I}}$ he or she will have validated 82.47% and 92.51% of the wrong mappings generated by OSDT and MSMBT respectively. At this point, even if the users stop the validation process, he or she has provided significant validations for alignment quality improvement. The informative mapping $\mathcal{M}^*$ which is displayed to the users last only contain 11 wrong mappings representing 6% and 7% of wrong mappings flagged by OSDT and MSMBT respectively. This shows that even if the user had abandoned the validation process before validating the mappings in $\mathcal{M}^*$, most of the required validation have already been captured. These techniques are geared towards prioritizing wrong mappings over the correct ones hence utilizing users' effort.

Table 3

Performance of different rankings of low quality mapping set

Technique	Total Mappings Selected	True Positives	True Negatives	Precision	Recall
$\mathcal{M}_{\mathcal{I}}$	81	67	14	0.8271	0.2597
$\mathcal{M}^*_{\mathcal{I}}$	161	136	25	0.8447	0.5271
$\mathcal{M}_{\mathcal{A}}$	172	98	74	0.5698	0.3798
$\mathcal{M}^*_{\mathcal{A}}$	92	29	63	0.3152	0.1124
$\mathcal{M}$	75	3	72	0.04	0.0116
$\mathcal{M}^*$	101	11	90	0.1089	0.04264

4.3. Evaluating The Performance of Tagging Process in Improving Alignment Quality

This section answers the question: Does mapping validation through tagging lead to a significant improvement in quality of the final alignment? To answer the question, we used SBOMT to match OAEI anatomy track ontoloies i.e. human.owl and mouse.owl. We then used both MSMBT and OSDT techniques to flag low quality mappings. From the low quality mappings generated by MSMBT and OSDT, we created the most informative set $\mathcal{M}^*_{\mathcal{I}}$, the averagely informative set $\mathcal{M}^*_{\mathcal{A}}$ and the informative set $\mathcal{M}^*$. The mappings in $\mathcal{M}^*_{\mathcal{I}}$ were displayed to the users first followed by those in $\mathcal{M}^*_{\mathcal{A}}$ and $\mathcal{M}^*$ respectively. In each set, mappings were displayed to the users in the decreasing order of similarity values. To tag the concepts from the low quality mapping sets, we employed the services of 14 masters' student from the zoology department who were familiar with the concepts in the two ontologies hence considered domain experts. The results are displayed in table 4. We compared our results to those from interactive track (Anatomy) in OAEI 2016 conference. We specifically compared our results with those posted by AML and LogMap in the interactive track (Anatomy) with 0.1 error rate.

Table 4
Performance of tagging framework

Tool	Precision	Recall	F-measure
BEFORE USER INVOLVEMENT			
AML	0.95	0.936	0.943
LogMap	0.911	0.846	0.877
$SBOMT^{tagging}$	0.90	0.87	0.8847
AFTER USER INVOLVEMENT			
AML	0.953	0.945	0.956
LogMap	0.961	0.832	0.909
$SBOMT^{tagging}$	0.954	0.926	0.942

The tagging framework $SBOMT^{tagging}$ results in the highest increase in precision value i.e. 0.054 which represents a 6% increase in precision. Compared to AML and LogMap, AML registers a 0.003 increase in precision which represents a 0.32 % increase in precision. LogMap increases the precision by 0.050 which represents 5.4 % increase in precision value. When it comes to recall, AML records an increase of 0.009 which represents 0.96% increase while LogMap is affected negatively by user's feedback since its recall drops by 0.014 representing 1.654% decrease in recall value. $SBOMT^{tagging}$ records an increase of 0.056 on recall which represents 6.436% increase in recall. On the overall, AML benefits the least from the user's feedback since it records the lowest increase in f-measure i.e. 0.013 while LogMap records an increase of 0.032. $SBOMT^{tagging}$ records the highest increase in f-measure i.e. 0.0571. These results show that alignment quality improvement based on multi-user tagging of concepts that are part of low quality mapping is superior compared to alignment quality improvement based on a single user and controlled feedback. It should also be noted that the results of AML and LogMap are based on a simulated single user with at error rate of 0.1 while in our case, $SBOMT^{tagging}$ is based on real users' feedback hence the error rate in our case may be higher.

4.4. Evaluation of Ontologies' Annotation Enrichment

This section answers the question: Does enrichment of an entity's annotations in an ontology help in disambiguating its meaning? To answer this question, we first employed SBOMT to match mouse.owl and human.owl ontologies that had not been modified (see lower part of figure 3). We then matched the same ontologies after the first iteration of the tagging process had been completed and the annotations of the entities which participated in low quality mappings enriched (see the upper part of figure 3). The results are shown in table 5

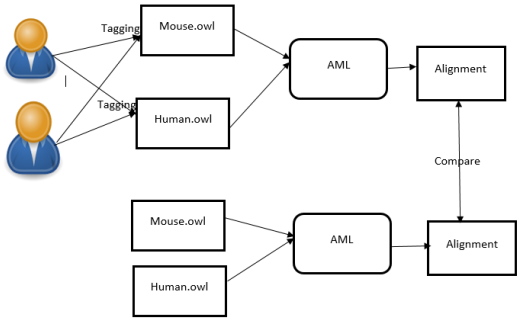

Fig. 3. Comparing alignment quality before and after tagging

According to the results in table 5, after the enrichment of annotations of entities which were flagged as low quality, the f-measure records an increase of 0.0371. This may be attributed to the fact that increased annotations help in the disambiguation of the meanings of some entities which were previously not

Table 5
Matching results before and after annotation enrichment

Tool	Precision	Recall	F-measure
BEFORE ANNOTATION ENRICHMENT			
SBOMT	0.90	0.87	0.8847
AFTER ANNOTATION ENRICHMENT			
SBOMT	0.937	0.907	0.9218

clear hence AML could not properly match them. This is a justification of the tagging effort since in case of future matching of these ontologies with a new one, most of their entities' meanings will be clear leading to them being correctly mapped with the entities of the new ontology.

5. Conclusion

The research has been able to demonstrate that:

1. Alignment quality validation through uncontrolled vocabulary leads to higher improvements in both the precision and recall values of the alignment as compared to the improvements based on controlled vocabulary.
2. Quality tags generated by users can be used to significantly improve entities' meanings.

Despite these achievements, the challenge that still remains is how to partition the wrong mappings into subtopics contained in the ontology such that a user can only be shown wrong mappings within a subtopic which he or she has the most experience in. By this we believe that more quality tags will be generated hence pushing the quality of repaired alignment even higher.

References

[1] T. R. Gruber, "A translation approach to portable ontology specifications," *Knowledge Acquisition*, pp. 199–220, 1993.

[2] P. Euzenat, J., Shvaiko, "Ontology Matching," *Springer, Heidelberg*, 2007.

[3] P. Shvaiko and J. Euzenat, "Ontology Matching: State of the Art and Future Challenges," *IEEE Transactions on Knowledge and Data Engineering*, pp. 158–176, 2013.

[4] L. Otero-Cerdeira, F. J. Rodríguez-Martínez, and A. Gómez-Rodríguez, "Ontology matching: A literature review," *Expert Systems with Applications*, pp. 949–971, 2015.

[5] M. Nentwig, M. Hartung, A.-c. Ngonga, and E. Rahm, "A Survey of Current Link Discovery Frameworks," *Semantic Web âĂŞ Interoperability, Usability, Applicability Journal*, pp. 1–17, 2015.

[6] D. H. Ngo and Z. Bellahsene, "Overview of YAM++âĂŤ(not) Yet Another Matcher for ontology alignment task," *Journal of Web Semantics*, pp. 30–49, 2016.

[7] E. Jiménez-Ruiz and B. Cuenca Grau, "LogMap: Logic-based and scalable ontology matching," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, pp. 273–288, 2011.

[8] C. Meilicke, "Alignment Incoherence in Ontology Matching - dissertation - meilicke," Ph.D. dissertation, Universitat Mannheim, 2011.

[9] E. Santos, D. Faria, C. Pesquita, and F. M. Couto, "Ontology alignment repair through modularization and confidence-based heuristics," *PLoS ONE*, pp. 1–17, 2015.

[10] C. Pesquita, D. Faria, E. Santos, and F. M. Couto, "To repair or not to repair: Reconciling correctness and coherence in ontology reference alignments," *CEUR Workshop Proceedings*, pp. 13–24, 2013.

[11] F. Osorno-Gutierrez, N. W. Paton, and A. A. A. Fernandes, "Crowdsourcing feedback for pay-as-you-go data integration," *CEUR Workshop Proceedings*, pp. 32–37, 2013.

[12] I. Cruz, F. Loprete, and M. Palmonari, "Pay-As-You-Go Multi-user Feedback Model for Ontology Matching," *International Conference on Knowledge Engineering and Knowledge Management*, pp. 80–96, 2014.

[13] F. Shi, J. Li, J. Tang, G. Xie, and H. Li, "Actively learning ontology matching via user interaction," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, pp. 585–600, 2009.

[14] C. Conroy, D. O'Sullivan, and D. Lewis, "A 'Tagging' approach to ontology mapping," *CEUR Workshop Proceedings*, pp. 1–5, 2007.

[15] P. Mika, "Ontologies Are Us: A Unified Model of Social Networks and Semantics," *Web Semantics.*, pp. 5–15, 2007.

[16] Z. Dragisic, V. Ivanova, P. Lambrix, D. Faria, E. Jiménez-Ruiz, and C. Pesquita, "User validation in ontology alignment," in *Proceedings of the International Semantic Web Conference*, 2016.

[17] V. Ivanova, P. Lambrix, and J. Å Berg, "Requirements for and evaluation of user support for large-scale ontology alignment," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2015, pp. 3–20.

[18] E. Jiménez-Ruiz, B. C. Grau, and Y. Zhou, "LogMap 2.0: Towards Logic-based, Scalable and Interactive Ontology Matching," in *SWAT4LS*, 2011, pp. 45–46.

[19] D. de Vaus, "Surveys In Social Research," *Routledge*, 2002.

[20] E. Jiménez-Ruiz, B. C. Grau, Y. Zhou, and I. Horrocks, "Large-scale interactive ontology matching: Algorithms and implementation," *Frontiers in Artificial Intelligence and Applications*, pp. 444–449, 2012.

[21] K. Belhajjame, N. W. Paton, A. A. A. Fernandes, C. Hedeler, and S. M. Embury, "User Feedback as a First Class Citizen in Information Integration Systems," *Conference on Innovative Data Systems Research (CIDR âĂŹ11)*, pp. 175–183, 2011.

[22] M. Davis, "Semantic Technology," *Director*, pp. 348–363, 2006.

[23] W. Eddine, M. Tarek, and S. Ben, "XMap: Results for OAEI 2016," *Proceedings of the 11th International Workshop on Ontology Matching*, 2016.

[24] P. G. Ipeirotis, F. Provost, and J. Wang, "Quality management on Amazon Mechanical Turk," *Proceedings of the ACM SIGKDD Workshop on Human Computation - HCOMP '10*, p. 64, 2010.

[25] I. F. Cruz, C. Stroe, and M. Palmonari, "Interactive user feedback in ontology matching using signature vectors," *Proceedings - International Conference on Data Engineering*, pp. 1321–1324, 2012.

[26] C. Sarasua, E. Simperl, and N. F. Noy, "CrowdMap: Crowdsourcing Ontology Alignment with microtasks," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7649 LNCS, pp. 525–541, 2012.

[27] S. Melnik, H. Garcia-Molina, and E. Rahm, "Similarity Flooding: A Versatile Graph Matching Algorithm," *Data Engineering*, pp. 117–128, 2002.

[28] S. Massmann, S. Raunich, D. Aumüller, P. Arnold, and E. Rahm, "Evolution of the COMA match system," *CEUR Workshop Proceedings*, pp. 49–60, 2011.

[29] T. T. Dang, A. Gabriel, S. Hertling, P. Roskosch, M. Wlotzka, J. R. Zilke, F. Janssen, and H. Paulheim, "HotMatch results for OEAI 2012," *CEUR Workshop Proceedings*, pp. 145–151, 2012.

[30] A. Groß, M. Hartung, T. Kirsten, and E. Rahm, "GOMMA results for OAEI 2012," *International Semantic Web Conference*, 2012.

[31] M. Gulić, B. Vrdoljak, and M. Banek, "CroMatcher - Results for OAEI 2015," 2015.

[32] E. Peukert and S. Massmann, "Comparing similarity combination methods for schema matching," *Journal of GI Jahrestagung*, pp. 692–701, 2010.

[33] Y. Song, L. Zhang, and C. L. Giles, "Automatic tag recommendation algorithms for social recommender systems," *ACM Transactions on the Web*, pp. 1–31, 2011.

[34] P. K. Chan, M. D. Schlag, and J. Y. Zien, "Spectral K-way ratio-cut partitioning and clustering," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1088–1096, 1994.

[35] D. A. Spielman and S. H. Teng, "Spectral partitioning works: Planar graphs and finite element meshes," *Linear Algebra and Its Applications*, pp. 284–305, 2007.

[36] L. Hagen, S. Member, and A. B. Kahng, "New Spectral Methods for Ratio Cut Partitioning and Clustering," *IEEE Transactions On Computer Aided Design*, 1992.

[37] J. Malik, S. Belongie, T. K. Leung, and J. Shi, "Contour and Texture Analysis for Image Segmentation," *International Journal of Computer Vision*, pp. 7–27, 2001.

[38] S. P. Borgatti and D. S. Halgin, "Analyzing Affiliation Networks," *The Sage Handbook of Social Network Analysis*, pp. 417–433, 2011.

[39] G. Salton, "Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer," *Analysis and Retrieval of Information by Computer*, p. 530, 1989.

[40] C. Cattuto, D. Benz, A. Hotho, and G. Stumme, "Semantic Analysis of Tag Similarity Measures in Collaborative Tagging Systems," *Data Engineering*, p. 5, 2008.

[41] B. Mohar, "Some applications of Laplace eigenvalues of graphs," *Graph Symmetry: Algebraic Methods and Applications*, pp. 225–275, 1991.

[42] U. V. Luxburg, "A Tutorial on Spectral Clustering A Tutorial on Spectral Clustering," *Statistics and Computing*, pp. 395–416, 2006.

[43] B. Mohar, "The Laplacian Spectrum of Graphs," in *Proc. 6th Quadrennial Intl. Conf. on Theory and Applications of Graphs*, 1988, pp. 871–898.

[44] K. M. Hall, "An r-Dimensional Quadratic Placement Algorithm," *MANAGEMENT SCIENCE*, pp. 219–229, 1970.

[45] C. H. Q. Ding, X. He, H. Zhab, M. Gu, and H. D. Simon, "A min-max cut algorithm for graph partitioning and data clustering," in *Proceedings 2001 IEEE International Conference on Data Mining*. IEEE, 2001, pp. 107 – 114.

[46] C. J. Alpert and S.-z. Yao, "Spectral Partitioning: The More Eigenvectors, The Better," in *32nd Design Automation Conference*. IEEE, 1995, pp. 195 – 200.

[47] A. Sánchez, D., Batet, M., Isern, D., & Valls, "Ontology-based semantic similarity: A new feature-based approach." *Expert Systems with Applications*, pp. 7718–7728, 2012.

[48] P. Ochieng and S. Kyanda, "A statistically-based ontology matching tool," *Distributed and Parallel Databases*, 2017.