User validation in ontology alignment (Extended version)

Zlatan Dragisic[1], Valentina Ivanova[1], Patrick Lambrix[1],
Daniel Faria[2], Ernesto Jiménez-Ruiz[3], and Catia Pesquita[4]

[1] Department of Computer and Information Science,
and Swedish e-Science Research Centre, Linköping University, Sweden
[2] Gulbenkian Science Institute, Portugal
[3] University of Oxford, UK
[4] LaSIGE, Faculdade de Ciências, Universidade de Lisboa, Portugal

Abstract. Involving users during the matching process has been identified as one of the challenges facing the ontology alignment community. Users are typically involved in the validation phase of automatically generated alignments. In this paper we identify and discuss issues that are important when dealing with user validation in ontology alignment. These issues pertain to the user profile as well as the user interfaces and services of the ontology alignment systems. We also provide an overview of how current systems deal with the identified issues. Using experiments from the Interactive matching track of the Ontology Alignment Evaluation Initiative (OAEI) 2015, we further investigate the rarely discussed topic of user expertise and demonstrate the impact of user knowledge on the alignment quality.

1 Introduction

The growth of the ontology alignment area in the past years has led to the development of many ontology alignment tools. In most cases ontology alignment systems apply fully automated approaches where an alignment is generated for a given pair of input ontologies without any interruption of the process. However, it has become clear that user involvement in the alignment process is needed for several reasons, and nearly half of the challenges facing the ontology alignment community identified in [47] are directly related to user involvement. These include *explanation of matching results* to users, fostering *user involvement* in the matching process and *social and collaborative matching*. The lack of evaluation of the quality and effectiveness of user interventions has been identified as one of the general issues after six years of experience in the Ontology Alignment Evaluation Initiative (OAEI) [12].

One important reason for user involvement is the quality of the alignments in terms of correctness and completeness. By allowing a domain expert to validate the alignments, wrong mappings may be detected and removed. A domain expert may also be able to add mappings not detected by the systems. This is in line with the observation by the OAEI organizers [12] that automatic generation of mappings is only a first step towards a final alignment and a validation by a domain expert is needed. Although domain experts can make mistakes, experiments in [26] suggest that as long as the error

rate is less than 30%, the validation is beneficial. Further, it has been shown in experiments that introducing domain expert knowledge already in the alignment generation phase could significantly improve the results and is essential for use cases requiring very accurate mappings [26]. There may also be an upper bound on the fully automatic ontology alignment in terms of precision, recall and F-measure [42]. This has been hypothezised based on the OAEI experience as well as by the fact that, although alignment techniques have become more advanced, comparable improvement of the alignments has not been achieved [21]. User involvement can also have an impact on the alignment process by adjusting system settings and by taking user knowledge into account in the system algorithms.

Although there have been earlier efforts to deal with user intervention and the identification and evaluation of the requirements and techniques (mostly user interfaces, e.g., [14, 17, 21, 28]), recently, with the realization of its importance, this problem is gaining more and more attention. For instance, the Interactive Matching track was introduced in the OAEI 2013 campaign [42] where user validation was simulated using an oracle. The number of user interventions and the time between them were also measured in addition to the influence of the validation on the quality of the alignments. The track was extended in 2015 to also deal with non-perfect domain experts as well as other use cases. In [24] requirements for fostering user support for large-scale ontology alignment were identified and current systems were evaluated.

In this paper we focus on the most common kind of user intervention in ontology alignment systems, i.e., user validation. We chose this kind of user intervention as all systems that support user intervention support some kind of validation of alignments. Furthermore, validation decisions have a large influence on the quality of the alignments. They can also be used to improve the alignment process and make the process more efficient. Our contributions include (i) the identification of issues that are important when dealing with user validation in ontology alignment (Section 2). These issues pertain to the user as well as the user interfaces and the services of the ontology alignment systems. Further, based on the literature, (ii) we provide an overview of how current systems deal with the identified issues (Section 3). Using experiments from the Interactive matching track of the OAEI 2015 campaign, (iii) we investigate the rarely discussed topic of user expertise and show the impact of user knowledge onto the alignment quality (Section 4).

2 Issues regarding user validation

We have identified a number of issues related to the user validation in ontology alignment by reviewing existing systems and literature related to ontology alignment as well as using our long experience in the field. Ontology alignment is a complex task where users are usually actively involved during the validation of automatically computed mappings. It consists of a number of steps where the users need to first become familiar with both ontologies and their formal representations, grasp ontology modelers' views and then understand and decide on the mappings provided by the system (in possibly large-scale cases) or create mappings by themselves [15]. These tasks are cognitively demanding and involve increasing memory load and decision making and are inherently

error-prone because of different levels of expertise, misinterpretations, human biases, etc. [17].

We have addressed these aspects with three categories of issues: issues related to the user expertise (Section 2.1), regarding errors, their impact and approaches to handle varying degrees of expertise; the system services (Section 2.2), concerning how systems formulate user interactions and how they capitalize on user input; and the user interfaces (Section 2.3), focusing on the impact of visualization and interaction strategies on alignment.

2.1 User expertise

An aspect that influences the quality of the validation is the expertise of the users that validate the mapping suggestions.

A first issue is the **knowledge of the user**. In [33] different kinds of oracles representing different levels of user knowledge were discussed. The ideal oracle is an all-knowing user which always validates mapping suggestions in a correct way. The partially correct oracle represents a user who knows part of the domain well. If a mapping suggestion is validated as a correct mapping, it is correct. Otherwise the mapping suggestion may be wrong or the user does not know. An approximation of this case is when using several users (e.g., [6]) and a skeptical approach, where we only consider a mapping suggestion to be a correct mapping if all domain experts agree on the correctness of the mapping suggestion. Users make mistakes (e.g., [4, 22]). One way of measuring the impact of the mistakes is by simulating the user by an oracle with different error rates. In this paper we investigate domain experts who for each validation make a mistake with a probability of x%. In the fully automatic alignment case, there is no domain expert and all mapping suggestions are assumed to be correct.

While important, possible user errors are often disregarded in the existing tools. For example, in [26] user validated relations are fixed, meaning that during the next phase of alignment repairing user validated relations cannot be removed from the final alignment. An implication is that a user error can propagate and that in the repairing phase some other correct relations are removed. Some actions such as ambiguity removal and showing only those mapping suggestions which do not introduce unsatisfiable classes are ways of limiting the impact of erroneous user validations. In another approach [23] if a user makes contradicting validations the system will raise a warning. In a multi-user setting errors may be taken into account using a voting strategy, such that the mapping confidence is proportional to the consensus on the mapping [6].

Another issue pertaining to this discussion is the **area of expertise of the user**, i.e., the difference between the knowledge engineers and domain experts. As ontologies are often domain specific, domain specific knowledge is needed to map them. However, domain experts are often not familiar with knowledge engineering concepts and formal representations (e.g., [4]). In the early days of the knowledge-based systems the role of the domain expert was to provide the domain specific knowledge to the knowledge engineer who was responsible to make it available to the expert system [20]. Later on tools like Protégé aimed to help the domain experts to encode domain knowledge given some training. Although the differences between these two user types are important for the design of every knowledge-based system, there are not many investigations that aim

to formally reveal these differences so they can be addressed when building suitable tool support.

Another aspect that is important is the **experience of the user with the tool**. Tools should support both novice and expert users [38]. The usability of the tool should be such that novice users can focus on the matching process instead of the tool usage. Expert users should be able to use shortcuts and can customize the tool in order to speed up their work.

2.2 System services

Based on **when** the validation is performed, the systems can use the validation decisions in different ways (**stage of involvement**). In the case of validation done *before* - {Sys.a} the matching process, the user provides an initial set of mappings (partial alignment, PA) which are then used by the system to guide the matching process. This can be done in several ways. Some approaches use a PA in the preprocessing phase to reduce the search space [30] or in the matchers [11, 30]. Another use is to utilise the PA to compute a recommendation for the configuration of the matchers' parameters (e.g., [29, 44, 50]).

When the validation is performed *after* - {Sys.b} the automatic matching process, many systems filter out mapping suggestions which are in conflict with user validations before proceeding to the final reasoning and diagnosis phase [23, 26, 31, 40]. While user involvement at the end of the matching can be valuable as a way of filtering mapping suggestions, systems can benefit from user involvement during the matching process as it can be used to guide the process. In addition, this type of interaction gives a possibility of discovering user errors and possible contradictions in the user validations.

If the validation is done *during* - {Sys.c} the matching, the user is providing direct feedback to the system which can then be propagated so that the later recomputations made by the system are guided by the user's feedback (**feedback propagation**, {Sys.f}). Most of the existing approaches accept the user feedback as true and non-debatable and in some cases give it priority over automatically generated mappings. Several systems define a kind of neighborhood for the mappings. For instance, for structural algorithms it may be defined using ancestors and descendants of the concepts in the mappings [31, 40, 45]. Other systems may define a neighborhood of the mappings based on how similar the scores from different matchers are [7, 30]. Mapping confidence is usually propagated from validated mappings to their neighborhood.

Many existing systems test user validated relations against the ontologies and report on possible conflicts which violate the logic within the ontology (unsatisfiable classes, inconsistencies) and possibly ask for revalidations of certain relations ({Sys.g}) to resolve the conflict. However, in most cases they will overlook user validations which are wrong according to the domain but do not cause a conflict. One way to deal with these is to involve multiple users in the validation process such as in [6]. The decisions on whether to accept or reject a mapping can be based on the consensus of the users and can be used to limit the validation of relations which are incorrect according to the domain. Ontology alignment is unlikely a single-person task and involving several users addresses one of the ontology matching challenges.

Some tools also employ multiple iterations of the matching process ({Sys.d}) where in each iteration an alignment from the previous iteration is improved [29]. In this case the validation process might be done multiple times.

Another aspect that further complicates the matching problem and consequently the user participation in the process, is the growing size and complexity of the ontologies. As users are a rare and valuable resource, limiting the user intervention is needed. Reducing the user interventions, but at the same time effectively combining manual validation with automatic computations is a challenge [26, 41]. Most existing approaches limit the number of mapping suggestions (**suggestions selection** - {Sys.e}) by employing threshold values for different matchers. However, there are a number of more advanced filtering approaches. One approach is to limit the number of mapping suggestions by filtering them with respect to some principle, e.g., consistency, locality and conservativity principles [25] or quality checks [3]. Several systems employ automatic decisions for where a concept is suggested to be mapped to multiple concepts. When a user validates a mapping suggestion as correct, then the other suggestions are automatically validated as incorrect [26, 31]. Another approach for limiting user involvement is to ask questions whose answers have the highest impact in the validation process. This could be done by identifying the top suggestions on which different matchers disagree [7] or by using a similarity propagation graph that allows for selecting the most informative questions [45].

Aspects of user validation related to system services can be measured in a number of ways. In order to evaluate the amount of user involvement in the matching process, it is possible to measure the number of questions (mapping suggestions) the system asks the user. This should be compared to the total number of pairs of concepts between the ontologies or to the total number of mapping suggestions the system produces. The impact that different measures for minimizing user involvement might have, can be tested by comparing precision and recall. Given that user error and levels of expertise can have significant impact on the matching results, these can be tested by simulating a user with different error rates.

2.3 User interface

A graphical user interface (UI) is an indispensable part of every interactive system and will enable efficient manual validation. Below we discuss features related to the visualization of the alignments and address basic and more advanced interactions to be supported by ontology alignment systems in the context of reducing memory load and supporting the decision making process.

2.3.1 Alignment Presentation

Visualizing ontologies and alignments takes advantage of the human's most powerful perception channel - the visual system and provides means for enhanced exploration. Encoding properties of entities and mappings with different graphical primitives allows to quickly convey information for them and their structure and compare them. Visually presenting information and making it easily retrievable will relieve the user from remembering it. Reducing users' working memory load is important given its limited

capacity (the working memory can hold 3 ± 1 items [49]). For users new to a system the UI related items compete for the capacity of their working memory. Grouping mappings together by different criteria is one way to reduce the items stored in the user working memory and also helps in identifying patterns, such as regions with many/few mappings, which otherwise will not be immediately perceivable {UI.d}. Visually depicting different types of mappings and their provenance makes this information readily accessible to the users {UI.e}. A powerful technique adopted by many systems to address these two points is color-coding. Providing the distinction between potential and verified mappings has been identified as a requirement in a theoretical framework for cognitive support during the alignment process based on cognitive and decision making theories [17]. Many of the requirements from the framework aim to address the limited capacity of the working memory—provide user annotations {UI.m}, context and definitions of terms {UI.f}, trial execution to help the user understand the consequences of his actions {UI.i}, temporary mappings ({UI.o}, {UI.e}) and history of actions to keep track of the sequences of his actions and revert them ({UI.a}). Other features can also contribute to reducing the user memory load — maintaining the user focus in one area of the ontology [40], preserving the user's mental model, for instance, forced-directed graph drawing algorithms may change the ontology layout after rerunning.

The **Visual Information Seeking Mantra** [46] defines seven high-level tasks to be supported by information visualization interfaces which aim at enhanced data exploration and retrieval. Six of them were further refined for the purposes of ontology visualization in [27]—overview, zoom, filter, details-on-demand, relate and history. Advanced search and filtering provide easy access to the information instead of recalling it from memory. Several systems provide search services, but they have serious shortcomings [24]. Further, providing advanced navigation (e.g., linking and brushing [17, 35]) helps for enhanced exploration and retrieval of information.

The field of **visual analytics**, {UI.b}, combines data mining and interactive visualization techniques with the powerful human visual system to aid analytic reasoning and obtaining insights into (large) data sets. As ontologies and alignments grow in size and complexity these techniques have the potential to quickly reveal information for large alignments [7]. Some first attempts to visually support data analysis in ontology alignment were proposed in [32, 35]. In [32] an evaluation framework for ontology alignment strategies was developed and insights obtained for the alignment process were showcased. In [35] two analytic questions that users attempt to answer for an alignment were considered - are there any clusters in the data and what are the relationships between their members. Recently, [2] identified four analytic questions for which domain experts need support from the tools and extended the visual analytics features of [5].

Alternative views {UI.c} provide complementary information and facilitate the perception of different features ([5, 17, 29, 35]), e.g., multiple inheritance is easier to perceive in graphs than in indented lists [19]. The views could be connected by interactive navigational techniques which greatly enhance the exploration, such as linking and brushing, where selection in one of the views changes the presentation in the other. Furthermore, different views would be more suitable for different alignment tasks [19].

Ontology alignment is by nature a decision making process where the user needs to make a choice between a number of alternatives with the goal of most accurately rep-

resenting the relationships between two ontologies. As discussed in [49] "The primary cognitive activity in decision making is the evaluation of each possible choice and the determination of the one most likely to achieve current goals". Recommendations and ranking ({UI.g}), visual analytics interfaces ({UI.b}) and alternative views ({UI.c}) contribute to the decision making process by explicitly presenting different alternatives and providing comparisons. An important support for the evaluation of possible choices is the **explanation of mapping suggestions.** Two things are very relevant in this context—presenting the provenance, or justification, of a mapping ({UI.h}), and depicting the consequences of the user choice ({UI.i}). The latter is particularly relevant for the decision making process when showing the consequences of user actions helps the user in exploring and evaluating their choices and deciding how to proceed.

Justifications have been identified as one of the future challenges of ontology matching [41]. While important, they are often only presented as confidence values. Some systems provide some information on the immediate parents/children of the terms involved in the mapping ([23, 26]). In [15] the context and definitions of the terms involved in the mapping as well as the reasons why it was suggested/accepted/rejected are identified and presented as provenance information to the user {UI.f}. The importance of the context can be seen in a study in [39] where the authors compared crowdsourcing (Amazon Mechanical Turk) with domain experts in a task where users had to verify hierarchical statements in ontologies. The results have shown that domain experts achieve better results, however when provided with concept definitions (in addition to concept names) the difference between the two user types was not statistically significant. In [17] some basic information in natural language on why a certain mapping was chosen is provided, e.g., that two concepts were mapped because they have the same suffix.

Another aspect of the justifications is their complexity. In [13] three types of justifications were identified since while provenance information and process traces might be enough for expert users, these are not enough for ordinary users and possibly domain experts who lack knowledge of the ontology matching process. These types are proof presentation, strategic flow and argumentation approach. In the proof presentation approach the explanation represents a proof of inference of a mapping suggestion (depicted by either a formal proof or natural language or a visualization). For example, in [48] simple and short natural language explanations without any technical details are given; in [29] relevant parts of the ontologies are visualized. In the strategic flow approach the explanation is in the form of a decision flow which describes the provenance of the acquired mapping suggestion, e.g., in [10] these are in the form of dependency graphs. Finally, in the argumentation approach, the system gives arguments for or against certain mapping suggestions. These are often not used for explanations but more as a way of achieving consensus in multi-user environments.

2.3.2 Alignment Interaction

Apart from the visual information seeking tasks discussed above, other interactions are required to support the validation process. In the validation process the basic interaction done by the user is to either accept or reject a certain mapping suggestion {UI.j}. It may also happen that a system has not suggested a mapping that is needed according to the user, in these cases the system has to provide functionality to add a mapping manually {UI.k} (e.g., [1, 5, 16, 29]).

The functions above are mostly supported by the systems. Other identified functions with fewer support are functions such as adding metadata, i.e., user annotations [17, 29] and creating temporary mappings {UI.o} in order to support temporary decisions [35], searching {UI.l} and filtering as other requirements for minimizing the user's cognitive load [1, 5, 17, 35]. In some cases, searching and filtering relate to searching/filtering parts of the ontologies related to terms in mapping suggestions [35, 17] while in other cases these relate to searching/filtering of mapping suggestions [5, 17, 29]. The mapping process is a continuous process and the tool has to accommodate interruptions (sessions, {UI.n})—saving and loading the ontologies and mappings is often possible but this usually does not preserve any provenance information.

3 Issues vs Systems

In Table 1 we give an overview of the identified issues related to the system and user interface and show if these are addressed in a number of systems. The systems considered in our study are those which incorporate the user validation in the alignment process and have a mature user interface. In the table ✓ marks that all of the listed items are supported by the system while - marks that the issue is not covered by the system. Combinations such as ✓- and ✓- - mark that one or two of the listed items are not supported. Below we give more details about these systems as well as how they address the issues.

AgreementMaker In AgreementMaker, [5, 7, 8] an initial alignment is computed which is then iteratively altered according to the user feedback {Sys.d}. For every mapping the system creates a signature vector containing its similarity values calculated by different matchers. The signature vector is then used during the processes of mapping suggestions selection and feedback propagation {Sys.e} {Sys.f}. For every mapping a disagreement metric is calculated—the disagreement is high when the similarity values computed by the different matchers are in a wide interval. The system presents the top-k mappings with the highest disagreement values for user validation. The mappings are clustered based on their vectors and the user feedback is propagated to those with similar vectors to the vector of the mapping validated by the user applying a linear function for both accepted and rejected mappings; already validated mappings are not updated any longer and the same mapping is not shown again to the user, e.g., it is not validated twice. The user can adjust the size of the feedback propagation cluster via a threshold. The approach described in [6] also discusses blocking propagation in the context of multi-user alignment. In this case the feedback propagation can be controlled via the consensus of users for a given mapping suggestion {Sys.g−}. Conflict detection is not discussed).

The ontologies are visualized as trees and their mappings are depicted as color-coded lines representing the matcher that calculated the similarity value (the value itself is shown as a number without an explanation {UI.h−−}). As described in [8] comments and additional information are shown when a concept is selected {UI.f} {UI.a}. A visual analytics panel {UI.b} {UI.a} helps in comparing the similarity values calculated

Table 1: Issues addressed by state-of-the-art systems

				Agreement Maker [5, 7, 8]	AlViz [35]	AML [18, 43]	Cogz Prompt [16, 17, 40]	COMA [1]	LogMap [26]	SAMBO [29, 31]	RepOSE [23]
System	Stage of involvement	{Sys.a}-before,{Sys.b}-after, {Sys.c}-during, {Sys.d}-iterative		{Sys.d}	{Sys.b}	{Sys.b}	{Sys.d}	{Sys.c}	{Sys.b}	{Sys.d}	{Sys.b}
	Suggestions Selection	{Sys.e} threshold/advanced filtering		✓	✓-	✓	-	✓-	✓	✓-	✓-
	Feedback Propagation	{Sys.f} recomputation		✓	-	-	✓	✓	✓- -	✓	✓-
		{Sys.g} conflict detection /blocking/revalidation		✓-(*)	-	✓	✓- -	-	✓	✓	✓
User Interface	Alignment Presentation	{UI.a} 7 visual info-seeking tasks		✓	✓	✓- -	✓	✓- -	-	✓- -	✓- -
		{UI.b} visual analytics		✓	-	-	-	-	-	-	-
		{UI.c} alternative views		✓	✓	✓	✓	-	-	✓	-
		{UI.d} grouping		✓	✓	✓	✓	-	-	✓	✓
		{UI.e} validated/candidates/temporary mappings		✓-	✓-	-	✓	✓- -	-	✓-	✓-
		{UI.f} metadata and context		✓	-	✓	✓	-	✓	✓- -	✓-
		{UI.g} ranking/recommendations		-	✓- -	-	✓- -	-	✓-	-	✓
		Mapping Explanation	{UI.h} provenance of a mapping (confidence value, explanation, justification complexity)	✓- -	✓- -	✓- -	✓-	✓- -	✓- -	-	✓- -
			{UI.i} impact of decisions/ consequences of actions	✓-	-	-	✓- -	-	✓-	-	✓- -
	Alignment Interaction	{UI.j} accept/reject mapping		✓	✓-	✓-	✓	✓-	✓	✓	✓
		{UI.k} create/refine mapping		✓	✓	✓	✓	✓	-	✓	✓-
		{UI.l} search		-	✓	✓	✓	✓	-	✓	-
		{UI.m} user annotation		-	-	-	✓	-	-	✓	-
		{UI.n} session		✓-	✓-	✓- -	✓-	✓-	✓	✓	✓-
		{UI.o} create temporary mapping		-	✓	-	✓	-	-	-	-

by the different matchers, their combination and 'disagreement'; it also shows the mappings in the same cluster {UI.c} {UI.d}. The user can adjust the size of the cluster and visualize its members thus visualizing the impact of the validations {UI.g−}. In the visual analytics panel the mappings are represented with matrices (one per matcher) and are color-coded to represent accepted, rejected and manually created mappings {UI.e−} (temporary mappings are not supported). The tool supports all {UI.a} of the seven information visualization seeking mantra tasks with a different level of coverage: overview is supported by observing the lines representing the mappings; filtering by different criteria {UI.a} can reduce the number of mappings shown to the user (by threshold, by matcher, by number of mappings per concept); undo and redo {UI.a} are supported as described in [8] but it is not clear if they are supported in later versions and how undoing an action would affect the propagation algorithm. During the validation process the user can accept, reject {UI.j} and create mappings manually {UI.k} (6 mappings types are supported). All mappings calculated for a particular concept are shown to the user [36]. Sessions {UI.n−} are not directly supported but load and save operations provide indirect support.

AlViz AlViz [35, 34] is a Protégé plugin which uses multiple views to visualize an alignment produced by another plugin {UI.c}. During the alignment process each ontology is represented as a pair of views—a tree and a small world graph—i.e., four in total. The views are connected by the linking and brushing paradigm where navigation in one of the views changes the representation in the other. The nodes in the ontology are clustered according to a selected relationship, called also a mutual property, and level of detail, where the sizes of the clusters depend on the number of nodes in them and the colors are determined by one out of three strategies {UI.d}. Color-coding shows the degree of similarity and the type of the association but explicit similarity values are not provided {UI.h−−}.

Mappings are edited, (indirectly) accepted and rejected {UI.j−} in the tree views by using toolbar buttons for choosing between one out of six types of mappings, called associations {UI.k} {UI.e−}. There is no clear distinction between mappings and mapping suggestions {UI.e−}. Temporary decisions for questionable mappings are supported by a tracking button {UI.e−}{UI.o}. A search field under the tree view is provided for each ontology {UI.l}

The tool supports many of the tasks from the information-seeking mantra {UI.a}—overview by small world graphs, zoom in to selected level of detail, filter by mutual property, details-on-demand by tooltips and labels, history by list of activities and undo/redo buttons, and relate by different comparison strategies. Ranking and recommendations at a mapping level are not provided but the color-coding of the clusters can help in the identification of interesting regions and starting points {UI.g−−}. Sessions can be considered supported by save and load {UI.n−}.

AML AML [18, 43] is a lightweight ontology matching system that focuses primarily on scalability and coherence. While its matching algorithms are fully automated, AML supports user interaction after the matching procedure {Sys.b}. It employs an interactive selection algorithm based on the similarity scores produced by its various

matching algorithms {Sys.e} for picking candidates for revision, also taking into account ambiguous mappings. Additionally, it employs an interactive repair algorithm that addresses conflicting mappings {Sys.g}. However, neither algorithm is yet available through its user interface. AML's user interface displays two different views of the alignment {UI.c}: a local graph view, where the user can visualize an individual mapping and its local context, including related mappings {UI.d}; and a list view that serves as an overview and enables the user to find further details about each mapping (both structural and lexical) upon clicking {UI.f}. Additionally, the latter also enables the user to review and reject mapping candidates {UI.j−}. AML's menu enables the user to customize the graph view, and offers the functionalities of searching the alignment for mappings {UI.l} and creating new mappings {UI.k} by searching through the ontologies. While AML doesn't explicitly implement sessions, it does allow the user to save and load the alignment at any stage, and thus interrupt and resume their revision at will {UI.n−−}.

CogZ/Prompt CogZ [17, 15] builds on PROMPT [40]. It is an extension of PROMPT's user interface and uses its other components to address cognitive requirements defined in [17]. The matching process is iterative. The tool starts by making a list of initial mapping suggestions which are then presented to the user. Based on the user's validations, the tool will check for conflicts and proceed in building additional suggestions which are based on the user's previous input {Sys.d} {Sys.f}. The examples of conflicts considered by the tool are name conflicts, dangling references, redundancy in the class hierarchy, slot value restrictions that violate class inheritance [40] {Sys.g−−} (only conflict detection, tool automatically deals with the conflicts, possible revalidations are not discussed).

The ontologies in CogZ are represented as trees and mappings are represented as dashed lines between concepts. Hovering over a mapping will show the explanation for the mapping. The explanations are short natural language texts describing the reasons why a certain mapping was selected {UI.h−} (confidence value not shown). Users can also explore the neighborhoods of the terms in a mapping suggestion {UI.d} {UI.f} {UI.i−−}. The user can define manually new mappings as well as add annotations {UI.k} {UI.m}. Users can mark a mapping suggestion as a temporary mapping {UI.o} {UI.e}. When the user is validating a mapping {UI.j} the tool presents mappings related to the parts of the ontology where the user is currently working on {UI.d}. The candidate-heavy regions in this view can be identified by parts of the ontologies with large concentration of lines between them {UI.g−−}, but no recommendations are provided at a single mapping level. In addition, the user interface provides searching {UI.l}/filtering {UI.a} for both ontologies and mapping suggestions. All of the tasks considered in the visual information seeking manta are supported {UI.a} and the system keeps track of user's previous decisions and the user can at any time inspect the mapping suggestions as well as already completed validations. In this way, the user can follow his/her progress.

In [15] the authors show the alternate tree-map view {UI.c} which provides an overview of the ontology and mapping suggestions. In this view ontologies are partitioned into parts and color-coded depending on the number of mapping suggestions.

Thus, candidate-heavy regions can be identified through different color intensities. In addition to this, this view provides a pie-chart for each branch of the ontology which contains numbers of mapping suggestions, mapped concepts and concepts without association.

The mappings can be stored and loaded from file thus the whole mapping process does not need to be done in one occasion {UI.n−}.

COMA++ COMA++ [37, 1] is a system for aligning large schemas and ontologies. The system supports the fragment-based matching strategy. In this strategy, the system applies a divide-and-conquer approach where it aims at identifying similar fragments of ontologies which are then matched. The user can validate the output from each phase of this process (matching pairs of fragments) and the user's validation will be used in subsequent computations {Sys.c} {Sys.f}.

The ontologies are shown as trees and mappings are shown as lines between them; the similarity values are color-coded in the lines' colors {UI.h−−} and are shown on hover. Therefore, regions with a large number of lines represent regions with many mapping suggestions (candidate-heavy regions). Under each ontology there is a search box {UI.l}. The tool has limited support for the information seeking tasks with filter, history and relate not supported {UI.a−−}.

The system allows saving and loading generated mappings to the mapping repository {UI.n−}. Users can reject {UI.j−} or manually add new mappings {UI.k}, it supports complex mappings, not only equivalence mappings. The system does not differentiate in the interface between mapping suggestions and validated mappings neither supports temporary mappings {UI.e−−}. If a user validates a mapping suggestion it is assigned the highest confidence value.

LogMap LogMap [26] is an ontology matching system that implements scalable reasoning and diagnosis algorithms, which minimize any logical errors introduced by the matching process. LogMap supports user interaction during the matching process, which is essential for use cases requiring accurate mappings.

LogMap presents to the user only the mappings that are not "clear cut" cases {Sys.b} {Sys.e}, for which user feedback would be highly beneficial. The number of such mappings can still be significant; hence, it is crucial to reduce the number of questions to the human expert by applying automatic decisions based on users' feedback. Automatic decisions based on a particular user decision to accept or reject a mapping ({UI.j}) are made according to two criteria: ambiguity and conflictness. That is, additional mappings will be (automatically) rejected/accepted if they were in conflict (i.e., lead to an unsatisfiable class) or ambiguous (i.e., share the source or target entity) with the user accepted/rejected mapping {Sys.f−−} {Sys.g}.

Each candidate mapping is presented to the user with a confidence value {UI.h−−}, information about the context/scope of the matched entities (i.e., superclasses and subclasses), and lexical information such as synonyms {UI.f}. In addition, the ambiguous mappings and mappings in conflict are also presented in order to help the user understand the consequences of the feedback {UI.i−}. The user can validate the mappings in one or several sessions {UI.n} or end the interactive process at any time (the remaining

cases are decided heuristically). Finally, LogMap ranks mappings {UI.g−} according to their impact on other mappings (i.e., mapping that have other mappings in conflict are shown first).

SAMBO SAMBO [29, 31] is a session-based {UI.n} ontology alignment system. The sessions are in the form of interruptible computation sessions. Users can thus begin the validation process even before the completion of the computation. In addition, users can specify points of interruptions, e.g., by specifying the number of concept pairs that need to be processed before the interrupt and the validation process. The system is iterative and computation sessions can reuse results from previous validations {Sys.d} {Sys.f}. The validation decisions are also used in the recommendation of settings for the alignment algorithms [50]. A PA can be used to reduce the search space [30] {Sys.a}. Selection of mappings suggestions is controlled via different combination and filtering strategies using the similarity values for different matchers ({Sys.e−}). It is possible to include the reasoner to check the consistency of the validated mapping suggestions. If conflicts are found they will be reported to the user (supports only checks for problems within the logic in ontologies, users are asked to revalidate, {Sys.g−}).

The user interface is implemented in the form of tabs where each tab is related to one part of the alignment process. The system groups together related mapping suggestions {UI.d}. Related mappings are those which share the same terms. For every mapping suggestion, the user can select to either reject it or accept it as either a subsumption relation or equivalence relation {UI.j}. The system also allows manual creation of mappings {UI.k} as well annotation of decisions {UI.m}, showing the annotation is however not easy {UI.f−−}. In the manual mode {UI.c}, the user is presented with a tree view of the ontologies and needs to select one concept from each tree and the relation between them in order to create a mapping manually {UI.e−}, temporary mappings are not supported. In this view, the user can also search the ontologies {UI.l}.

Mainly the history task is supported from the seven information seeking tasks {UI.a−−}— the system logs all user decisions which can then be reviewed. An undo button is supported as well. The user can also review remaining mapping suggestions which need to be dealt with. Relate and details on demand are indirectly supported by grouping mappings for a single concept together in the list view.

RepOSE RepOSE [23] is based on an integrated taxonomy[5] alignment and debugging framework. The system can be seen as an ontology alignment system with a debugging component for detecting and repairing modelling defects (missing and wrong subsumption relations/mappings) in taxonomy networks (both in the alignments and ontologies). The alignment process goes through three phases - generation of mapping suggestions, validation and repairing {Sys.b}. Selection of mappings suggestions is only controlled by threshold values for different matchers {Sys.e}. During the repairing step for every accepted mapping the user is given possibility to add a mapping which would make the accepted mapping derivable. A limited form of recomputation as user feedback is used

[5] Other versions of RepOSE deal with slightly more expressive ontologies, but focus more on the debugging phase, e.g., [51].

in the repairing process {Sys.f}. The system checks for contradictions after each group of suggestions is validated and after a repairing action and does not allow the current action to take place if such are found {Sys.g}.

During the validation phase the mapping suggestions are shown as graphs in groups where the last group in the list contains the most suggestions {UI.g} {UI.d}, thus the user can choose the area to start with. The nodes are color-coded according to their hosting ontology and the edges - the state of the represented mappings - mapping suggestions, added/rejected mappings ({UI.e−}, temporary mappings are not supported). The justifications of the mappings inferred from the taxonomy network are presented to the user as graphs, for the rest a tooltip that appears when the user hovers on an edge shows the similarity value calculated by the (combination of) matchers ({UI.f−}{UI.h−−}—only derivation path, no information on why the mapping was selected). If the current user action contradicts with previous actions an error message is shown to the user {UI.i−−}. Users can accept/reject mappings {UI.j}. During the validation and repairing processes recommendations based on external knowledge are provided {UI.g} for every mapping. Creating arbitrary mappings is not possible but some flexibility is provided during the repairing phase where the user can create a mapping that entails the accepted mapping (mappings can be refined in the repairing phase {UI.k−}). The system supports equivalence and subsumption mappings. Sessions are indirectly supported (users can save/load mappings {UI.n−}). Some overview and details-on-demand (regarding mappings) tasks are supported ({UI.a−−}).

4 Experiments

In this section we report on experiments that address the issue of the knowledgeability of the domain expert and its impact on the performance of alignment.

4.1 Set up

The SEALS client used to evaluate systems in OAEI was modified to allow interactive matchers to pose questions regarding the correctness of a mapping to an oracle. The interactive matcher can present a mapping suggestion to the oracle, which then tells the system whether the mapping is correct or wrong. A request is considered distinct if one of the concepts or the relationship in a mapping has changed in comparison with previous requests. The oracles can be perfect or answer with a predefined error rate. In this experiment the error rates are 10%, 20% and 30%. (Or_w^x represents an oracle with x % error rate.)

We used three datasets of the OAEI: Conference, Anatomy and Large Biomedical Ontologies (Large Bio). The Conference dataset covers 16 ontologies describing the domain of conference organization. We only use the test cases for which an alignment is publicly available (altogether 21 alignments/tasks). The Anatomy dataset includes two ontologies (1 task), the Adult Mouse Anatomy (AMA) ontology and a part of the National Cancer Institute Thesaurus (NCI) describing the human anatomy. Large Bio consists of 6 tasks with different sizes ranging from tens to hundreds of thousands

classes and aims at finding alignments between the Foundational Model of Anatomy (FMA), SNOMED CT, and the National Cancer Institute Thesaurus (NCI).

The evaluations of the Conference and Anatomy datasets were run on a server with 3.46 GHz (6 cores) and 8GB RAM allocated to the matching systems. Each system was run three times and the final result of a system for each error rate represents the average of these runs. For the Conference dataset with the ra1 alignment, we considered macro-average of precision and recall of different ontology pairs, while the number of interactions represent the total number of interactions in all tasks. Finally, the three runs are averaged. The Large Bio dataset evaluation (each system was run once) was run on a Ubuntu Laptop with an Intel Core i7-4600U CPU @ 2.10GHz x 4 and allocating 15GB of RAM.

4.2 Systems and interaction

We used the participants from the OAEI Interactive track 2015[6]: AML, JarvisOM, LogMap, and ServOMBI. We note that not all of these systems have user interfaces, but they do have the functionality to ask questions to an oracle.

The systems involve the user in different points of the execution and explore the user input in different ways. Apart from JarvisOM, the systems all make use of user interactions exclusively in post-matching steps. LogMap and AML request feedback on selected mapping suggestions. AML filters mapping suggestions based on the user validations. It also uses a query limit and other strategies to minimize the user interaction. LogMap interacts with the user to decide on mapping suggestions which are not clear-cut cases. ServOMBI employs the user to validate all its mapping suggestions and uses the validations and a stable marriage algorithm to decide on the final alignment. JarvisOM is based on an active learning strategy known as query-by-committee. At every iteration JarvisOM asks the user for pairs of entities that have the highest disagreement between committee members and lower average euclidean distance. In the last iteration, the classifiers committee is used to generate the alignment.

4.3 Results

The first six columns in each of the tables present the results obtained in this and the non-interactive Anatomy track (marked as Prec non., F-m. non and Rec. non in the tables). The ”Prec. Oracle”, ”Rec. Oracle” and ”F-m. Oracle” columns contain the evaluation results ”according to the oracle”, meaning against the oracle’s alignment i.e. the reference alignment as modified by the randomly introduced errors. Next, ”Tot Req.”, ”Dist. Req.” give the total number and number of distinct requests to the oracle. ”TP”, ”TN”, ”FP” and ”FN give the number of true positive, true negative, false positive and false negative answers from the oracle.

Results for the anatomy dataset Tables 2, 3, 4 and 5 present the results for the Anatomy dataset for the all-knowing oracle and oracles with three different error rates.

[6] http://oaei.ontologymatching.org/2015/interactive/

Figure 1 shows the time intervals between the questions to the user/oracle for the different systems and error rates for the three runs (the runs are depicted with different colors).

For each system, its strategy for introducing user interaction, leads to an improvement of precision and F-measure while the recall improves or stays the same. JarvisOM displays the largest improvement —the F-measure improves almost 4,5 times and the size of the alignment generated by the system also grows around 2,5 times.

We first compare the performance of each of the four systems with an all-knowing oracle (Or_w^0 - Table 2), in terms of precision, recall and F-measure, to the non-interactive versions of the systems (these are the first 6 columns in the corresponding tables). For each system, its strategy for introducing user interaction, leads to an improvement of precision and F-measure while the recall improves or stays the same. JarvisOM displays the largest improvement —the F-measure improves almost 4,5 times and the size of the alignment generated by the system also grows around 2,5 times.

With the introduction of an erroneous oracle/user and moving towards higher error rates, system performance starts to slightly deteriorate in comparison to the all-knowing oracle. However, the changes in the error rates influence the four systems differently in comparison to the non-interactive results. While the AML performance with an all-knowing oracle is better on all measures with respect to the non-interactive results, the precision drops in the Or_w^{20} and Or_w^{30} cases (Tables 4 and 5), while the recall stays higher than the non-interactive results for all error rates. LogMap behaves in the opposite way, i.e., recall in the Or_w^{20} and Or_w^{30} cases drops below the non-interactive results, while the precision stays higher in all error rates. For ServOMBI the F-measure and recall drop below the non-interactive results already in the Or_w^{10} case (Or_w^{10} - Table 3), but the precision is higher in all cases. In contrast JarvisOM performs better in the interactive cases on all measures than in the non-interactive cases where it achieved very low values for all measures. We note the large drop in precision (around 35 percentage points) for JarvisOM with the growing error rates in comparison to the other three systems where the drop in precision is between 1 to 5 percentage points. This could be explained by the fact that JarvisOM asks only few questions and is therefore very sensitive to false positives and false negatives. Another interesting observation is that, with the exception of AML, the performance of the systems also declines as the error increases with regard to the oracle's reference. This means that the impact of the errors is linear for AML (i.e., one erroneous response from the oracle, leads to only one error from AML) but supralinear for the others.

Regarding requests AML, JarvisOM and LogMap do not present the same question again to the user, while ServOMBI does.

For AML the size of the alignment and the number of (distinct) requests to the oracle is stable across the different error rates. JarvisOM uses very few requests and this number is stable across the different error rates. Another notable difference is the varying size of the alignment generated by JarvisOM which almost doubles in the Or_w^{20} case comparing to the all-knowing oracle. For LogMap the number of requests grows with the error rate together with a slight grow in the alignment size. ServOMBI asks the user for every mapping suggestion found and the number of distinct requests for ServOMBI

Tool	Prec.	Prec. non	F-m.	F-m. non	Rec.	Rec. non	Prec. Oracle	F-m. Oracle	Rec. Oracle	Tot. Reqs.	Dist. Reqs.	TP	TN	FP	FN	Time
AML	0.97	0.96	0.96	0.94	0.95	0.93	0.97	0.96	0.95	312.0	312.0	73.0	239.0	0.0	0.0	49
JarvisOM	0.86	0.36	0.75	0.17	0.67	0.11	0.86	0.75	0.67	7.0	7.0	4.0	3.0	0.0	0.0	213
LogMap	0.98	0.91	0.91	0.88	0.85	0.85	0.98	0.91	0.85	590.0	590.0	287.0	303.0	0.0	0.0	24
ServOMBI	1.00	0.96	0.76	0.75	0.62	0.62	1.00	0.76	0.62	2136.0	1128.0	955.0	173.0	0.0	0.0	711

Table 2: Anatomy dataset – Or_w^0

Tool	Prec.	Prec. non	F-m.	F-m. non	Rec.	Rec. non	Prec. Oracle	F-m. Oracle	Rec. Oracle	Tot. Reqs.	Dist. Reqs.	TP	TN	FP	FN	Time
AML	0.96	0.96	0.95	0.94	0.95	0.93	0.97	0.96	0.95	317.3	317.3	66.3	218.0	23.0	10.0	45
JarvisOM	0.76	0.36	0.68	0.17	0.67	0.11	0.76	0.68	0.67	7.0	7.0	3.3	3.0	0.3	0.3	214
LogMap	0.96	0.91	0.89	0.88	0.83	0.85	0.96	0.89	0.83	609.0	609.0	261.3	288.3	33.7	25.7	25
ServOMBI	1.00	0.96	0.71	0.75	0.55	0.62	1.00	0.74	0.59	2198.7	1128.0	857.3	156.3	16.7	97.7	563

Table 3: Anatomy dataset – Or_w^{10}

Tool	Prec.	Prec. non	F-m.	F-m. non	Rec.	Rec. non	Prec. Oracle	F-m. Oracle	Rec. Oracle	Tot. Reqs.	Dist. Reqs.	TP	TN	FP	FN	Time
AML	0.94	0.96	0.94	0.94	0.94	0.93	0.97	0.96	0.95	321.7	321.7	66.3	186.7	52.3	16.3	47
JarvisOM	0.53	0.36	0.60	0.17	0.71	0.11	0.53	0.60	0.71	8.0	8.0	4.7	1.0	1.3	1.0	214
LogMap	0.95	0.91	0.88	0.88	0.82	0.85	0.95	0.88	0.81	630.0	630.0	233.0	274.0	69.0	54.0	24
ServOMBI	0.99	0.96	0.66	0.75	0.49	0.62	1.00	0.71	0.55	2257.0	1128.0	767.3	131.3	41.7	187.7	571

Table 4: Anatomy dataset – Or_w^{20}

Tool	Prec.	Prec. non	F-m.	F-m. non	Rec.	Rec. non	Prec. Oracle	F-m. Oracle	Rec. Oracle	Tot. Reqs.	Dist. Reqs.	TP	TN	FP	FN	Time
AML	0.93	0.96	0.93	0.94	0.94	0.93	0.97	0.96	0.95	306.0	306.0	54.0	168.7	61.3	22.0	48
JarvisOM	0.51	0.36	0.49	0.17	0.53	0.11	0.51	0.49	0.53	7.3	7.3	4.0	1.7	1.0	0.7	214
LogMap	0.94	0.91	0.87	0.88	0.82	0.85	0.92	0.86	0.80	663.0	663.0	200.7	270.7	105.3	86.3	24
ServOMBI	0.99	0.96	0.60	0.75	0.43	0.62	1.00	0.68	0.52	2329.7	1128.3	663.3	129.0	44.3	291.7	447

Table 5: Anatomy dataset – Or_w^{30}

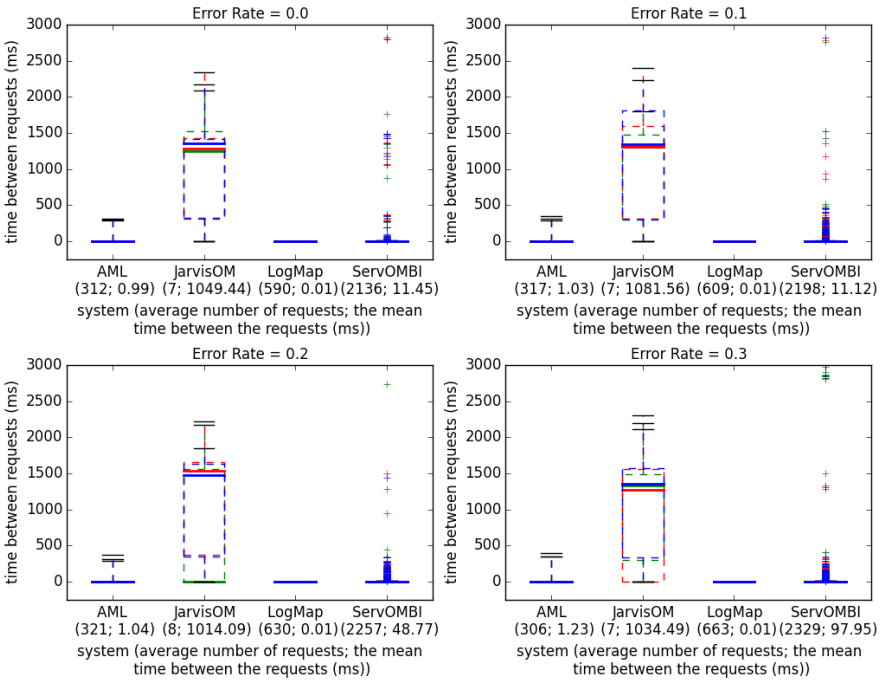

Fig. 1: The Y axis depicts the time intervals between the requests to the user/oracle (whiskers: Q1-1,5IQR, Q3+1,5IQR, IQR=Q3-Q1). The labels under the system names show the average number of requests and the mean time between the requests for the three runs.

stays stable for the different rates. The total number of requests is almost double the distinct ones while the size of the alignment drops when introducing higher error rates.

The run times between the different error rates do not significantly change for AML, LogMap and JarvisOM. The ServOMBI run time decreases with the increase of the error rate. In comparison to the non-interactive track, LogMap's and JarvisOM's run times do not change. AML's run time is between 10 to 20 % higher in the interactive track. ServOMBI's run time is higher in the non-interactive track.

For an interactive system the time intervals at which the user is involved in an interaction are important. Figure 1 shows the boxplots regarding the time periods at which the systems present a question to the user. Across the three runs and different error rates the AML and LogMap request intervals are around 1 and 0 milliseconds respectively. On the other hand, while the requests periods for ServOMBI are under 10 ms in most of the cases there are some outliers requiring more than a second. Furthermore, a manual inspection of the intervals showed that in several cases it takes more than 10 seconds between the questions to the user and in one extreme case 250 seconds. The requests intervals for this system increase at the last 50–100 questions. JarvisOM displays a delay in its requests in comparison to the other systems. The average interval at which a

question is presented to the user is 1 second with about half of the requests to the user taking more than 1.5 seconds. However, it issues the questions during the alignment process and not as a post processing step.

Results for the conference dataset Tables 6, 7, 8 and 9 present the results for the Conference dataset with the perfect oracle and oracles with three different error rates. Figure 2 shows the average requests intervals per task (21 tasks in total per run) between the questions to the oracle for the different systems and error rates for all tasks and the three runs (the runs are depicted with different colors). The first number under the system names is the average number of requests and the second number is the average period of the average requests intervals for all tasks and runs.

We first compare the performance of the systems with an all-knowing oracle (Table 6) to the performances of their non-interactive versions for the Conference track. The biggest improvement in F-measure is achieved by ServOMBI with 23 percentage points. Other systems also show substantial improvements, AML improves the F-measure by 8, JarvisOM by 13 and LogMap by around 4 percentage points. In the case of ServOMBI and LogMap interaction with the user improved precision while recall was improved only slightly. On the other hand, for JarvisOM recall improved substantially while keeping a similar level of precision. Finally, the precision for AML improved by 10 and recall by 6 percentage points which contributed to a higher F-measure.

As expected, the results start deteriorating when introducing the error in the oracle's answers. Interestingly, even with the Or_w^{30} (Table 9) most systems perform similar (with respect to the F-measure) to their non-interactive version. For example, AML's F-measure in the case with Or_w^{30} is only 1 percentage point worse than the non-interactive one. The most substantial difference is in the case of ServOMBI with an oracle Or_w^{30} where the system achieves around 5 percentage points worse result w.r.t. F-measure than in the non-interactive version. Again closer inspection shows that different systems are affected in different ways when errors are introduced. For example, if we compare the all-knowing oracle and the Or_w^{30} case, we can see that for AML, precision is affected by 11 and recall by 6 percentage points. In the case of JarvisOM, precision drops by 19 while recall drops by only 4 percentage points. LogMap is affected in a similar manner and its precision drops by 9 while the recall drops by only 3 percentage points. Finally, the most substantial change is in the case of ServOMBI where the precision drops from 100% to 66% and the recall shows a drop of 22 percentage points. Like in the Anatomy dataset, LogMap and ServOMBI also show a drop in performance in relation to the oracle's reference with the increase of the error rate, which indicates a supralinear impact of the errors. AML again shows a constant performance that reflects a linear impact of the errors. Also JarvisOM shows a constant performance, which is a different behavior than in the anatomy case.

When it comes to the number of requests to the oracle, 3 out of 4 systems make around 150 requests while ServOMBI does most requests, namely 550. AML, JarvisOM and LogMap do not repeat their requests while around 40% of requests made by ServOMBI are repeated requests. Across the three runs and different error rates the AML and LogMap mean times between requests for all tasks are less than 3 ms. On the other hand, mean time between requests for ServOMBI and JarvisOM are around 30 and 10

Tool	Prec.	Prec. non	F-m.	F-m. non	Rec.	Rec. non	Prec. Oracle	F-m. Oracle	Rec. Oracle	Tot. Reqs.	Dist. Reqs.	TP	TN	FP	FN	Time
AML	0.94	0.84	0.82	0.74	0.72	0.66	0.94	0.82	0.72	147.0	147.0	53.0	94.0	0.0	0.0	28
JarvisOM	0.81	0.84	0.65	0.52	0.55	0.37	0.81	0.65	0.55	154.0	154.0	38.0	116.0	0.0	0.0	39
LogMap	0.87	0.80	0.72	0.68	0.62	0.59	0.87	0.72	0.62	157.0	157.0	52.0	105.0	0.0	0.0	27
ServOMBI	1.00	0.56	0.79	0.57	0.65	0.59	1.00	0.79	0.65	535.0	295.0	156.0	139.0	0.0	0.0	50

Table 6: Conference dataset – Or_w^0

Tool	Prec.	Prec. non	F-m.	F-m. non	Rec.	Rec. non	Prec. Oracle	F-m. Oracle	Rec. Oracle	Tot. Reqs.	Dist. Reqs.	TP	TN	FP	FN	Time
AML	0.91	0.84	0.79	0.74	0.71	0.66	0.94	0.82	0.73	147.3	147.3	48.3	85.3	8.7	5.0	27
JarvisOM	0.73	0.84	0.61	0.52	0.53	0.37	0.77	0.64	0.55	154.0	154.0	34.3	107.0	10.3	2.3	38
LogMap	0.83	0.80	0.69	0.68	0.60	0.59	0.84	0.69	0.59	157.7	157.7	45.7	93.3	12.3	6.3	27
ServOMBI	0.89	0.56	0.70	0.57	0.57	0.59	1.00	0.78	0.64	555.3	299.3	137.7	126.3	16.7	18.7	51

Table 7: Conference dataset – Or_w^{10}

Tool	Prec.	Prec. non	F-m.	F-m. non	Rec.	Rec. non	Prec. Oracle	F-m. Oracle	Rec. Oracle	Tot. Reqs.	Dist. Reqs.	TP	TN	FP	FN	Time
AML	0.87	0.84	0.77	0.74	0.69	0.66	0.94	0.82	0.73	149.0	149.0	45.0	76.3	17.3	10.3	27
JarvisOM	0.67	0.84	0.58	0.52	0.52	0.37	0.77	0.65	0.56	155.0	155.0	28.7	97.3	22.7	6.3	38
LogMap	0.81	0.80	0.69	0.68	0.59	0.59	0.81	0.68	0.58	158.7	158.7	40.0	84.7	22.0	12.0	27
ServOMBI	0.80	0.56	0.61	0.57	0.50	0.59	1.00	0.77	0.62	554.7	295.7	122.0	110.7	29.0	34.0	50

Table 8: Conference dataset – Or_w^{20}

Tool	Prec.	Prec. non	F-m.	F-m. non	Rec.	Rec. non	Prec. Oracle	F-m. Oracle	Rec. Oracle	Tot. Reqs.	Dist. Reqs.	TP	TN	FP	FN	Time
AML	0.83	0.84	0.73	0.74	0.66	0.66	0.94	0.82	0.73	148.7	148.7	35.3	68.0	24.7	20.7	27
JarvisOM	0.62	0.84	0.56	0.52	0.51	0.37	0.74	0.65	0.58	154.3	154.3	24.3	88.0	32.0	10.0	39
LogMap	0.78	0.80	0.67	0.68	0.59	0.59	0.79	0.66	0.57	154.0	154.0	37.7	70.7	31.3	14.3	27
ServOMBI	0.66	0.56	0.52	0.57	0.43	0.59	1.00	0.77	0.63	589.7	308.0	105.0	103.7	48.0	51.3	50

Table 9: Conference dataset – Or_w^{30}

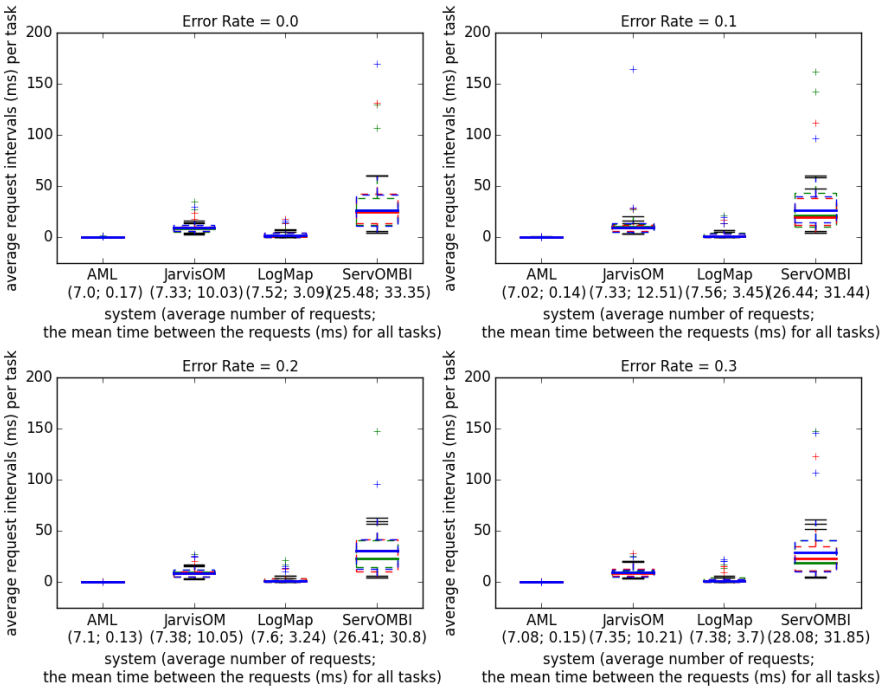

Fig. 2: The Y axis depicts the average time between the requests per task in the Conference dataset (whiskers: Q1-1,5IQR, Q3+1,5IQR, IQR=Q3-Q1). The labels under the system names show the average number of requests and the mean time between the requests (calculated by taking the average of the average request intervals per task) for the three runs and all tasks.

ms respectively. While in most cases there is little to no delay between requests, there are some outliers. These are most prominent for ServOMBI where some requests were delayed for around 2 seconds which is substaintally longer than the mean.

Results for the Large Bio dataset Tables 10, 11, 12 and 13 present the results for the Large Bio dataset for the all-knowing oracle and oracles with three different error rates.

Figure 3 shows the average requests intervals per task (6 tasks in total) between the questions to the oracle for the different systems and error rates for all tasks and a single run. The first number under the system names is the average number of requests and the second number is the average period of the average requests intervals for all tasks in the run.

With an all-knowing oracle (Table 10), AML, LogMap and ServOMBI all improved their performance in comparison with their non-interactive versions in the Large Bio track. The biggest improvement in F-measure was achieved by LogMap with 4, followed by AML with 3, then ServOMBI with 2 percentage points. AML showed the greatest improvement in terms of recall, but also increased its precision substantially; LogMap had

Tool	Prec.	Prec. non	F-m.	F-m. non	Rec.	Rec. non	Prec. Oracle	F-m. Oracle	Rec. Oracle	Tot. Reqs.	Dist. Reqs.	TP	TN	FP	FN	Time
AML	0.94	0.91	0.85	0.82	0.77	0.75	0.94	0.85	0.77	10217	10217	5126	5091	0	0	2877
LogMap	0.97	0.90	0.83	0.79	0.73	0.71	0.97	0.83	0.73	27436	27436	17050	10386	0	0	3803
ServOMBI*	1.00	0.97	0.85	0.83	0.74	0.74	1.00	0.85	0.74	21416	9424	8685	739	0	0	726

Table 10: Large Bio dataset – Or_w^0

Tool	Prec.	Prec. non	F-m.	F-m. non	Rec.	Rec. non	Prec. Oracle	F-m. Oracle	Rec. Oracle	Tot. Reqs.	Dist. Reqs.	TP	TN	FP	FN	Time
AML	0.93	0.91	0.84	0.82	0.76	0.75	0.94	0.85	0.77	10217	10217	4624	4658	485	450	2913
LogMap	0.94	0.90	0.80	0.79	0.70	0.71	0.94	0.80	0.70	28890	28890	15753	10659	1181	1297	3963
ServOMBI*	1.00	0.97	0.80	0.83	0.67	0.74	1.00	0.83	0.72	22920	9502	8063	726	85	628	695

Table 11: Large Bio dataset – Or_w^{10}

Tool	Prec.	Prec. non	F-m.	F-m. non	Rec.	Rec. non	Prec. Oracle	F-m. Oracle	Rec. Oracle	Tot. Reqs.	Dist. Reqs.	TP	TN	FP	FN	Time
AML	0.92	0.91	0.82	0.82	0.75	0.75	0.94	0.85	0.77	10217	10217	4196	4081	1049	891	2930
LogMap	0.92	0.90	0.78	0.79	0.68	0.71	0.91	0.77	0.68	30426	30426	14286	10707	2669	2764	3912
ServOMBI*	0.99	0.97	0.74	0.83	0.59	0.74	1.00	0.81	0.69	23968	9541	7431	661	192	1257	713

Table 12: Large Bio dataset – Or_w^{20}

Tool	Prec.	Prec. non	F-m.	F-m. non	Rec.	Rec. non	Prec. Oracle	F-m. Oracle	Rec. Oracle	Tot. Reqs.	Dist. Reqs.	TP	TN	FP	FN	Time
AML	0.91	0.91	0.82	0.82	0.75	0.75	0.94	0.85	0.77	10217	10217	3737	3637	1537	1306	2959
LogMap	0.90	0.90	0.77	0.79	0.68	0.71	0.87	0.74	0.65	31504	31504	13035	10147	4307	4015	3874
ServOMBI*	0.98	0.97	0.68	0.83	0.52	0.74	1.00	0.79	0.66	25580	9600	6818	652	256	1874	618

Table 13: Large Bio dataset – Or_w^{30}

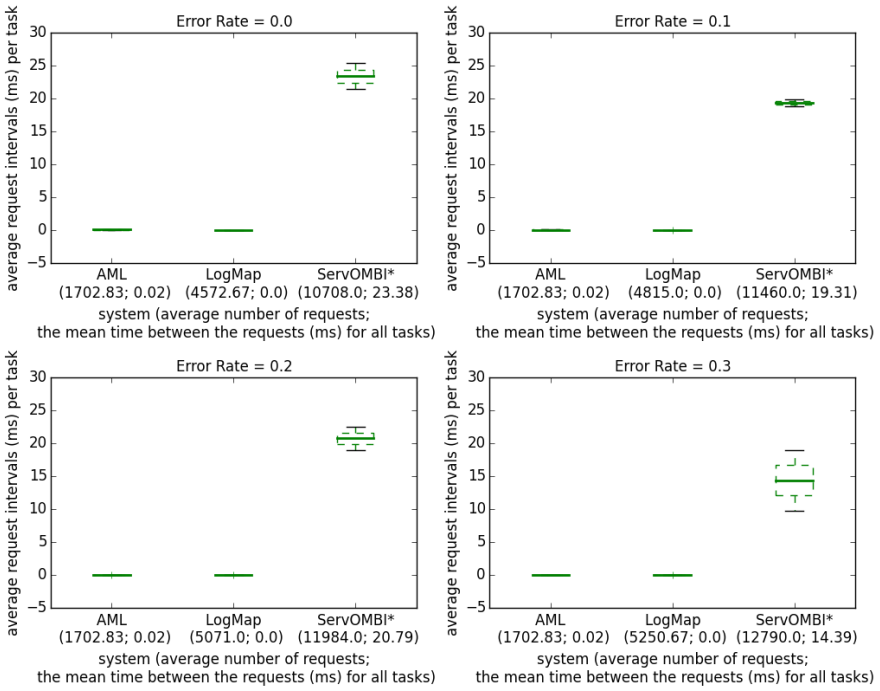

Fig. 3: The Y axis depicts the average time between the requests per task in the Large Bio dataset (6 tasks) (whiskers: Q1-1,5IQR, Q3+1,5IQR, IQR=Q3-Q1). The labels under the system names show the average number of requests and the mean time between the requests (calculated by taking the average of the average request intervals per task) for the three runs and all tasks.

the greatest improvement in terms of precision, but also showed a significant increase in recall; and ServOMBI improved essentially only with regard to precision, obtaining 100% as in the other datasets.

The introduction of user errors had a very different effect on the three systems: AML shows a slight drop in performance of 3 percentage points in F-measure between Or_w^0 and Or_w^{30} (Table 13), and is only slightly worse than its non-interactive version with Or_w^{30}; LogMap shows a more pronounced drop of 6 percentage points in F-measure; and ServOMBI shows a substantial drop of 17 percentage points in F-measure. Unlike in the other datasets, all systems are affected significantly by the error with regard to both precision and recall. Like in the other datasets, AML shows a constant performance in relation to the oracle's reference, indicating a linear impact of the errors, whereas the other two systems decrease in performance as the error increases, indicating a supralinear impact of the errors.

Regarding the number of requests to the oracle, AML was the more sparing system, with only 10,217, whereas LogMap made almost three times as many requests (27,436). ServOMBI was again the more inquisitive system, with 21,416 requests on only the

two smallest tasks in the dataset (for comparison, AML made only 1,823 requests on these two tasks and LogMap made 6,602). As in the other datasets, ServOMBI was the only system to make redundant requests to the oracle. Interestingly, both LogMap and ServOMBI increased the number of requests with the error, whereas AML had a constant number of requests.

Figure 3 presents a comparison between the systems regarding the average time periods for all tasks at which the system presents a question to the user. Across the different error rates the average requests intervals for all tasks for AML and LogMap are around 0 millisecond. For ServOMBI they are slightly higher (25 milliseconds on average) but a manual inspection of the results shows some intervals larger than 1 second (often those are between some of the last requests the system performs).

4.4 Discussion

Our experiments corroborate the suggestion in [26] that as long as the error rate is less than 30%, the validation is beneficial. The performance of the systems deteriorated with the increase of the error rate. Errors had different impact on different systems reflecting the different interactive strategies employed by the systems. In some cases erroneous answers from the oracle had the highest impact on the recall, in other cases on the precision, and in others still both measures were significantly affected. The impact of the errors was linear in some systems and supralinear in others. A supralinear impact of the errors indicates that the system is making inferences from the user and thus deciding on the classification of multiple mapping suggestions based on user feedback about only one. This is an effective strategy for reducing the burden on the user, but leaves the matching system more susceptible to user errors. An extreme example of this is JarvisOM on the Anatomy dataset, as it uses an active-learning approach based on solely 7 user requests, and consequently is profoundly affected when faced with user errors given the size of the Anatomy dataset alignment. JarvisOM behaves very differently in the Conference dataset, showing a linear impact of the errors, as in this case 7 requests (which is the average number it makes per task) represent a much more substantial portion of the Conference alignments (50%) and thus leads to less inferences and less impact of errors.

As ServOMBI employs the user to validate all its mapping suggestions, there are much more user requests than for the other systems, and in being the system most dependent on the user, is also the one most affected by user errors.

Regarding the number of user requests, we note that ServOMBI and LogMap generally increased the number of requests with the error, whereas AML and JarvisOM kept their number approximately constant. The increase is natural, as user errors can lead to more complex decision trees when interaction is used in filtering steps and inferences are drawn from the user feedback (such as during alignment repair) which leads to an increased number of subsequent requests. JarvisOM is not affected by this because it uses interaction during matching and makes a fixed 7-8 requests per matching task, whereas AML prevents it by employing a maximum query limit and stringent stopping criteria.

Two models for system response times are frequently used in the literature [9]. The Shneiderman model takes a task-centred view and uses four categories according to task

complexity: typing, mouse movement (50-150 ms), simple frequent tasks (1 s), common tasks (2-4 s) and complex tasks (8-12 s). It is suggested that the user is more tolerable to delays with the growing complexity of the task at hand. Unfortunately, no clear definition is given for how to define the task complexity. The Seow model looks at the problem from a user-centred perspective by considering the user expectations towards the execution of a task. The categories are instantaneous (100-200 ms), immediate (0.5-1 s), continuous (2-5 s) and captive (7-10 s). Ontology alignment is a cognitively demanding task and can fall into the third or fourth categories in both models. In this regard the response times (in our paper the request intervals) observed with the Anatomy dataset (with the exception of several measurements for ServOMBI) fall into the tolerable and acceptable response times in both models. The same applies for the average requests intervals for the 6 tasks in the Large Bio dataset. The average request intervals for the Conference dataset are lower (with the exception of ServOMBI) than those discussed for the Anatomy dataset. However, very low system response times may not always be needed as the user needs sufficient time for the validation of mapping suggestions.

5 Conclusions and Future Work

User validation is increasingly relevant in ontology alignment. Despite the advances in automated techniques, user validation remains critical in achieving higher quality alignments. In this paper we identified a number of issues concerning user validation in the ontology alignment process, focusing on three issues that highlight three relevant aspects of user involvement in ontology alignment: the influence of user expertise, the exploration of user input by the ontology alignment systems services and the impact of user interfaces in supporting interaction and involvement. These issues have received very little concerted attention in the field. Through literature review we showed how different state-of-the-art systems deal with these issues, providing a categorization of systems in what regards user involvement support. We also conducted experiments where we demonstrated the impact of user validation and of user errors on the quality of the alignment. Ascertaining the impact of user expertise is particularly relevant for domains where expert effort is costly, which is certainly the case of the biomedical domains employed in some of our experiments.

There are a number of directions for future work. In order to enhance usability, there is a need for best practices and guidelines for good UI design which should then be aligned with the ontology matching system design. In order to identify the requirements for different user types more studies are needed with real users, with varying degrees of expertise. Moreover, studies dedicated to alignment validation by multiple users, and to minimizing user burden are also needed, in order for instance to investigate the integration of expert knowledge and crowdsourcing.

Acknowledgments. We thank the Swedish e-Science Research Centre (SeRC), the Swedish national graduate school for computer science (CUGS) and the EU FP7 projects VALCRI (FP7-IP-608142) and Optique (grant agreement 318338) for financial support. This work has been partially supported by the Fundação da Ciência e Tecnologia through funding of LaSIGE Research Unit, ref.UID/CEC/00408/2013.

References

1. D Aumüller, H H Do, S Maßmann, and E Rahm. Schema and ontology matching with COMA++. In *SIGMOD*, pages 906–908, 2005.
2. J Aurisano, A Nanavaty, and I Cruz. Visual analytics for ontology matching using multi-linked views. In *VOILA*, pages 25–36, 2015.
3. E Beisswanger and U Hahn. Towards valid and reusable reference alignments-ten basic quality checks for ontology alignments and their application to three different reference data sets. *J Biomedical Semantics*, 3(S-1):S4, 2012.
4. C Conroy, R Brennan, D O'Sullivan, and D Lewis. User Evaluation Study of a Tagging Approach to Semantic Mapping. In *ESWC*, pages 623–637, 2009.
5. I Cruz, F Antonelli, and C Stroe. Agreementmaker: efficient matching for large real-world schemas and ontologies. *Proc VLDB Endowment*, 2(2):1586–1589, 2009.
6. I Cruz, F Loprete, M Palmonari, C Stroe, and A Taheri. Quality-based model for effective and robust multi-user pay-as-you-go ontology matching. *Semantic Web J*, 2015.
7. I Cruz, C Stroe, and M Palmonari. Interactive user feedback in ontology matching using signature vectors. In *ICDE*, pages 1321–1324, 2012.
8. I Cruz, W Sunna, N Makar, and S Bathala. A visual tool for ontology alignment to enable geospatial interoperability. *J Visual Languages & Computing*, 18(3):230–254, 2007.
9. J Dabrowski and E Munson. 40 years of searching for the best computer system response time. *Interacting with Computers*, 23(5):555–564, 2011.
10. R Dhamankar, Y Lee, A Doan, A Halevy, and P Domingos. imap: discovering complex semantic matches between database schemas. In *SIGMOD*, pages 383–394, 2004.
11. S Duan, A Fokoue, and K Srinivas. One size does not fit all: Customizing ontology alignment using user feedback. In *ISWC*, pages 177–192. 2010.
12. J Euzenat, C Meilicke, P Shvaiko, H Stuckenschmidt, and C Trojahn dos Santos. Ontology alignment evaluation initiative: six years of experience. *J Data Semantics*, XV:158–192, 2011.
13. J Euzenat and P Shvaiko. User Involvement. In *Ontology Matching*, pages 353–375. 2013.
14. S Falconer and N Noy. Interactive techniques to support ontology matching. In Z Bellahsene, A Bonifati, and E Rahm, editors, *Schema Matching and Mapping*, pages 29–51. 2011.
15. S Falconer, N Noy, and M-A Storey. Towards Understanding the Needs of Cognitive Support for Ontology Mapping. In *OM*, 2006.
16. S Falconer, N Noy, and M-A Storey. Ontology mapping - a user survey. In *OM*, pages 49–60, 2007.
17. S Falconer and M-A Storey. A Cognitive Support Framework for Ontology Mapping. In *ISWC/ASWC*, pages 114–127. 2007.
18. D Faria, C Martins, A Nanavaty, D Oliveira, B Sowkarthiga, A Taheri, C Pesquita, F M Couto, and I F Cruz. AML results for OAEI 2015. In *OM*, 2015.
19. B Fu, N Noy, and M-A Storey. Eye tracking the user experience-an evaluation of ontology visualization techniques. *Semantic Web J*, 2014.
20. J H Gennari, M A Musen, R W Fergerson, W E Grosso, M Crubzy, H Eriksson, N F Noy, and S W Tu. The evolution of Protégé: an environment for knowledge-based systems development. *Int J Human-Computer Studies*, 58(1):89–123, 2003.
21. M Granitzer, V Sabol, K W Onn, D Luckose, and K Tochtermann. Ontology Alignment—A Survey with Focus on Visually Supported Semi-Automatic Techniques. *Future Internet*, pages 238–258, 2010.
22. V Ivanova, J L Bergman, U Hammerling, and P Lambrix. Debugging taxonomies and their alignments: the ToxOntology-MeSH use case. In *WoDOOM*, pages 25–36, 2012.

23. V Ivanova and P Lambrix. A unified approach for aligning taxonomies and debugging taxonomies and their alignments. In *ESWC*, pages 1–15. 2013.
24. V Ivanova, P Lambrix, and J Åberg. Requirements for and evaluation of user support for large-scale ontology alignment. In *ESWC*, pages 3–20. 2015.
25. E Jiménez-Ruiz, B Cuenca Grau, I Horrocks, and R Berlanga. Logic-based assessment of the compatibility of umls ontology sources. *J Biomedical Semantics*, 2(S-1):S2, 2011.
26. E Jiménez-Ruiz, B Cuenca Grau, Y Zhou, and I Horrocks. Large-scale Interactive Ontology Matching: Algorithms and Implementation. In *ECAI*, pages 444–449, 2012.
27. A Katifori, C Halatsis, G Lepouras, C Vassilakis, and E G Giannopoulou. Ontology visualization methods - a survey. *ACM Computing Surveys*, 39(4):10, 2007.
28. P Lambrix and A Edberg. Evaluation of ontology merging tools in bioinformatics. In *Pacific Symposium on Biocomputing*, pages 589–600, 2003.
29. P Lambrix and R Kaliyaperumal. A Session-Based Approach for Aligning Large Ontologies. In *ESWC*, pages 46–60. 2013.
30. P Lambrix and Q Liu. Using partial reference alignments to align ontologies. In *ESWC*, pages 188–202, 2009.
31. P Lambrix and H Tan. SAMBO - a system for aligning and merging biomedical ontologies. *J Web Semantics*, 4(3):196–206, 2006.
32. P Lambrix and H Tan. A tool for evaluating ontology alignment strategies. *J Data Semantics*, VIII:182–202, 2007.
33. P Lambrix, F Wei-Kleiner, Z Dragisic, and V Ivanova. Repairing missing is-a structure in ontologies is an abductive reasoning problem. In *WoDOOM*, pages 33–44, 2013.
34. M Lanzenberger, J Sampson, and M Rester. Ontology visualization: Tools and techniques for visual representation of semi-structured meta-data. *J UCS*, 16(7):1036–1054, 2010.
35. M Lanzenberger, J Sampson, M Rester, Y Naudet, and T Latour. Visual ontology alignment for knowledge sharing and reuse. *J Knowledge Management*, 12(6):102–120, 2008.
36. Y Li, C Stroe, and I F. Cruz. Interactive visualization of large ontology matching results. In *VOILA*, pages 37–48, 2015.
37. S Massmann, S Raunich, D Aumüller, P Arnold, and E Rahm. Evolution of the COMA match system. In *OM*, pages 49–60, 2011.
38. J Nielsen. *Usability Engineering*. 1993.
39. N Noy, J Mortensen, P Alexander, and M Musen. Mechanical turk as an ontology engineer? In *ACM Web Science*, pages 262–271, 2013.
40. N Noy and M Musen. Algorithm and Tool for Automated Ontology Merging and Alignment. In *AAAI*, pages 450–455, 2000.
41. L Otero-Cerdeira, F J Rodríguez-Martínez, and A Gómez-Rodríguez. Ontology matching: A literature review. *Expert Systems with Applications*, 42(2):949–971, 2015.
42. H Paulheim, S Hertling, and D Ritze. Towards Evaluating Interactive Ontology Matching Tools. In *ESWC*, pages 31–45. 2013.
43. C Pesquita, D Faria, E Santos, J Neefs, and F M. Couto. Towards Visualizing the Alignment of Large Biomedical Ontologies. In *DILS*, pages 104–111, 2014.
44. D Ritze and H Paulheim. Towards an automatic parameterization of ontology matching tools based on example mappings. In *OM*, pages 37–48, 2011.
45. F Shi, J Li, J Tang, G Xie, and H Li. Actively Learning Ontology Matching via User Interaction. In *ISWC*, pages 585–600. 2009.
46. B Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *IEEE Symposium on Visual Languages*, pages 336–343, 1996.
47. P Shvaiko and J Euzenat. Ontology Matching: State of the Art and Future Challenges. *Knowledge and Data Engineering*, 25(1):158–176, 2013.
48. P Shvaiko, F Giunchiglia, P Da Silva, and D McGuinness. Web explanations for semantic heterogeneity discovery. In *ESWC*, pages 303–317. 2005.

49. E Smith and S Kosslyn. *Cognitive Psychology: Mind and Brain*. 2013.
50. H Tan and P Lambrix. A method for recommending ontology alignment strategies. In *ISWC/ASWC*, pages 494–507. 2007.
51. F Wei-Kleiner, Z Dragisic, and P Lambrix. Abduction framework for repairing incomplete el ontologies: Complexity results and algorithms. In *AAAI*, pages 1120–1127, 2014.