Interactive Visualization of Large Ontology Matching Results

Yiting Li

UIN 650514298

Fall 2014

Master of Science Project Report

Department of Computer Science, College of Engineering

University of Illinois at Chicago

Advisor: Isabel F. Cruz

Secondary Committee Member: Angus Forbes

Abstract

In this project, we added to the award winner ontology matching system AgreementMaker the capability to visualize the results of matching large ontologies. We provide a detailed view of the matching results to the users and allow them to navigate through different ontology levels, search, so as to traverse the structure of the source and target ontologies. This project also addresses the issue of user intervention when using a feedback loop strategy instead of focusing on matching algorithms that compute automatically candidate mappings. We predict that capabilities that are offered to users will empower them to make decisions on the correctness of mappings by displaying property clustering. Our approach was implemented using JavaFX and integrates seamlessly with the current implementation of AgreementMaker.

Acknowledgment

I would like to thank my advisor Professor Isabel Cruz and Cosmin Stroe, who gave me much help throughout the whole project. I would also like to thank Professor Cruz for her help with the writing of this project report. Professor Angus Forbes was the second reader of my project. I would like to thank him for his support.

Y.L.

Contents

Chapter 1

Introduction

An ontology provides a vocabulary describing a domain of interest and a specification of the meaning of terms in that vocabulary. An increasing number of organizations are using ontologies to organize their knowledge. However, different ontologies exist for the same knowledge domain. To address this issue, *ontology matching* is needed, which is the process of finding the relationships, called *mappings* between classes or properties of two different ontologies, the *source* and the *target* ontologies [1]. In this project, we focus on class matching. Ontology matching can be performed automatically, manually, or semi-automatically using a combination of both strategies.

A variety of algorithms have been developed for class matching. For example algorithms based on the similarity of the strings in the class labels or those based on the structure of the ontologies. Advanced ontology matching systems, such as AgreementMaker, use combinations of a large variety of algorithms [2, 3]. In this project, we will not be focusing on any particular matching algorithm, but rather on visualizing the results of the ontology matching process so as to enable the involvement of humans in the matching process with the objective of obtaining better results. The quality of a matching algorithm or of a combination of matching algorithms is measured in terms of precision, recall, and F-measure, by comparing the obtained mappings with the mappings that belong to the *gold standard* or *reference alignment*. The OAEI (Ontology Alignment Evaluation Initiative) [1] competition is instrumental for the research community by providing several ontology matching tracks and making some of the reference alignments

[1]http://oaei.ontologymatching.org/

available.

1.1 User Feedback Loop for Ontology Maching

Semi-automatic ontology matching has recently gained considerable attention. In those approaches, those mappings that are believed to be incorrect are presented to users for validation [4, 5]. The workflow consists of a loop where the outcome of the validation step is fed back into the ontology matching process. Visual analytics can help users in validating the mappings [4].

1.2 Visual Analytics

Visual Analytics is the science of analytical reasoning supported by interactive visual interfaces. According to Bertini and Lalanne, it is a new interdisciplinary domain that integrates several domains like: interactive visualization, statistics and data mining, human factors, to focus on analytical reasoning facilitated by interactive visual interfaces [6]. In the realm of ontology matching, a previous visual analytics approach involves the visualization of similarity matrices produced by the different matching algorithms [4]. A similarity matrix contains the results of matching two ontologies. To provide the users with complete information on the matching process, the results of multiple matching algorithms should be part of the visual analytics process [4].

1.3 Visualization

The focus of our project is on the interactive visualization for ontology matching. The visual representation of large-scale data helps people understand and analyze information. Our focus is on ontology matching visualization, instead of simple ontology visualization. While there has been much work done on visualizing ontology structures, while approaches for visualizing ontology matching results are rare. This might be due to the complex relationship between source and target ontology structures. This problem becomes further complicated when matching large ontologies. The display of an ontology as a tree structure using for example the JTree class can be very helpful for small and medium size ontologies, but is not helpful for large ontologies because of the amount of scrolling needed to locate the different mappings. Therefore to better compare and analyze the matching results, a better visual representation of data is required.

Chapter 2

Related Work

There has been little work on ontology matching visualization and especially on ontology matching visualization for large ontologies as is apparent from a recent survey [7]. This survey also does not report on an analysis of the available visualization tools from a user effectiveness viewpoint. Therefore, there is the urgent need to develop truly scalable ontology matching visualization approaches and to evaluate their usability. In what follows we list briefly the most relevant visualization methods we have uncovered in the past few months.

2.1 Cluster Visualization

The cluster representation [8] shows both detailed and general information of matching results and provides in addition a JTree visualization. Users can select the level at which they want to cluster the results. For the visualization of each ontology this approach uses a spring-embedded graph drawing algorithm. As the authors point out, this method is severely constrained by its computation complexity, which is $O(n^3)$ where n is the size of the ontology. Other drawbacks of the approach are that only the results of a single matching algorithm can be visualized. Nodes of each ontology are color coded so as to show whether they have been mapped and the level of similarity found with classes of the other ontology.

Another graph drawing representation that was developed for the AgreementMakerLight system [9], which extends AgreementMaker to very large ontologies, provides a single visualization that also uses a spring-embedded technique where both ontologies and the mappings between classes are displayed. However, it displays only a few mappings at a time [10]. This technique does not allow to compare the results of more than one matching algorithm at once.

2.2 Treemap and Pie Chart Visualizations

The PROMPT+COGZ tool supports multiple visualizations, including one based on TreeMaps and another one that displays pie charts [11]. This tool is designed to provide an overview of the ontologies and potential mappings. TreeMaps have the advantage that they can be used to visualize large amounts of data, but fit in a small area. Forcefully, details cannot be provided for large ontologies. Some details are provided by a pie chart view with information for each branch of the ontology, such as the number of candidate mappings, mapped classes, and classes that are not mapped. We note, however, that for the display of candidate mappings, the tool falls back on a JTree-like visualization, which uses a fish-eye view lens to allow for the display of larger ontologies. Clearly this is overall an advanced visualization tool. However, it does not seem to be able to show concurrent displays of more than one matching algorithm at a time. Together with our approach, this is the only other tool that supports pie charts with the different that our pie charts drive the navigation across all levels of the ontologies, while their navigation appears instead to be driven by the TreeMap visualization.

2.3 Matrix Visualization

A matrix visualization where the classes of both ontologies are placed along the X and Y axes provides a more comprehensive view of the matching process as compared with the aforementioned methods because it allows for the whole mapping space to be visualized with equal detail. We know of two such visualizations: the one provided by iMerge [12] and the one provided by AgreementMaker [4]. Both systems support multiple visualizations, including a traditional JTree-like visualization for each ontology with connections between the two ontologies showing the mappings. Like the systems already mentioned, these two systems do not scale to very large ontologies, a problem that could be remedied because of the usual sparsity of similarity matrices. AgreementMaker has the distinct capability of allowing for the comparison of different matching algorithms side by side and simultaneous navigation across the various similarity matrices. The color intensity supported by AgreementMaker, which depicts the matching confidence for each mapping, adds an extra dimension to the visualization without adding extra space.

Figure 2.1 shows the visual interface for matrix visualization of AgreementMaker, which is called *Visual Analytics Panel* because it is used to support the visual analytics process. The top toolbar controls the matching process. Colored squares represent mappings that are correct (green), missed (red), or falsely positive (blue). The intensity of the colors represents the similarity value in the range [0,1]. The overall panel highlights a vector of points for the same mapping (the signature vector). The two leftmost plots represent the similarity matrices for two of the algorithms. The third plot represents the weighted combination of all the algorithms. The rightmost plot represents the disagreement matrix among the algorithms in shades of grey [4].

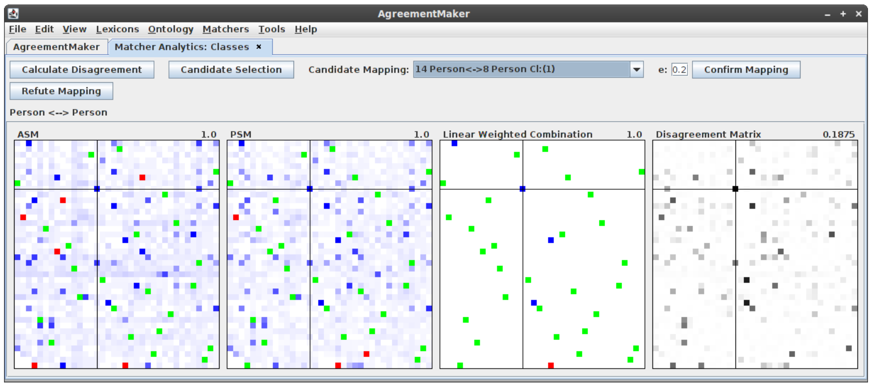

AgreementMaker
File Edit View Lexicons Ontology Matchers Tools Help
AgreementMaker
Matcher Analytics: Classes
Calculate Disagreement
Candidate Selection
Candidate Mapping: 14 Person<->8 Person Cl:(1)
e: 0.2
Confirm Mapping
Refute Mapping
Person <--> Person
ASM 1.0
PSM 1.0
Linear Weighted Combination 1.0
Disagreement Matrix 0.1875

Figure 2.1: The visual analytics panel of AgreementMaker [4].

.

Chapter 3

Visualization

3.1 Design Criteria

We have made the following choices and present their rationale:

- We do not display all the mappings at once. If we did so, it would be difficult to find a visualization whose size does not depend on the number of mappings or on the size of the ontologies involved.

- To supplement the detailed view, we allow for ontology navigation, exploration and searching.

- We want to focus on the mappings one level at a time and aggregate the results for the children of the nodes at that level. As long as navigation and searching functions are available, users can easily locate any single ontology node in the whole structure and see the mappings they need.

- We chose a visualization based on pie charts. The reason of this choice is that no matter how large the data size is, the pie chart requires always the same modest area. Given this, we can display the results of more than one matching algorithm at a time.

- When matching two classes, the most valuable information is the similarity value found by the matching algorithm. The visualization can give priority to those mappings that maximize the similarity value between two nodes.

- We want to make apparent the differences between matching algorithms.
- We enable user feedback acquisition.

3.2 Pie Chart Visualization

We start by describing how we visualize the information in terms of pie charts. For each visualization there are two pie charts, when corresponding to a source node S, called the *current node*, and a pie chart corresponding to a target node T. We show the percentage of their children whose matching similarity falls in a particular range. Figure 3.1 shows those ranges, namely 81%-100%, 61%-80%, 41%-60%, and ≤40%. For example, 41% of the children nodes of S have matching similarity values in the range 81%-100%. Figure 3.1 also highlights a mapping between two nodes, A that is a child of S and B that is a child of T.

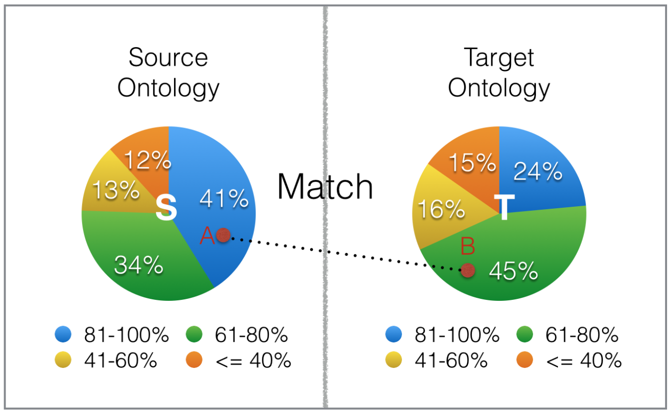


Figure 3.1: User interface prototype that displays matching results.

Figure 3.2 shows schematically subgraphs of both ontologies with roots S and T, their children, among which there are subclasses A and B, the siblings of A and B, and their children (and grandchildren). To enable navigation along the ontologies, users should be able to traverse the ontology "vertically" from parents to children but also "horizontally" from a node to its siblings.

The next question we address from the interface viewpoint is how to combine both types of navigation. We provide a list that represents the children of a node and a tree view that represents the siblings of the current node.

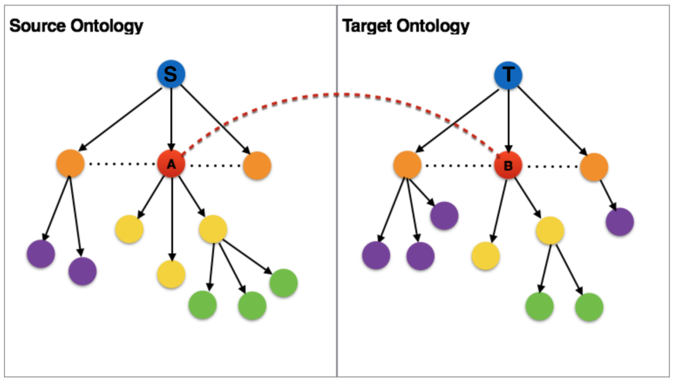


Figure 3.2: Two ontology subgraphs showing a mapping between nodes A and B.

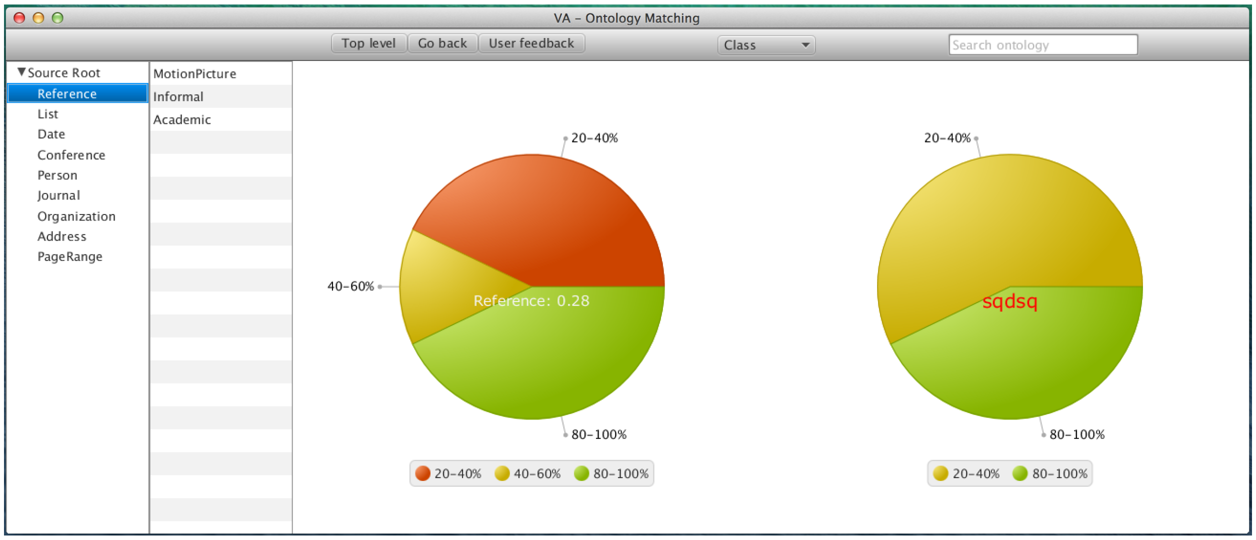


Figure 3.3: Main panel.

The main panel of the user interface for the first version of the prototype is shown in Figure 3.3. In the center area there are the two pie charts previously discussed.

Immediately left of the pie charts there is a list. When the users click on a pie chart slice, the list contains the ontology nodes with similarity value within the corresponding range, ordered by that value. Clicking on a node in the list leads to an update of the pie charts, as that node becomes the current node. The leftmost part of the interface contains the tree view. It shows all the siblings of the current node. When clicking on a node in the tree view, both the pie charts and the lists will be updated, reflecting the change of the current node. On the top right there is a search box. Upon entering the name of the node in the search box, the left pie chart will display that source node and the right pie chart will display the target node that matches to the source one. Accordingly, the tree and list view on the left will be updated as well. In addition, for an easier navigation we provide additional functions such as “go to the top level”, “go to the previous level”, and “switching between class and property”.

Upon entering the source ontology node’s name in the search box and pressing the enter key on the keyboard, the left pie chart will display the source node that is searched and the right pie chart will display the target node that matches to the source one. Accordingly, the two lists on the left will be updated as well.

3.3 Algorithm Comparison

To compare the results of more than one matching algorithm, we need to visualize their results at the same time. We have upgraded the main panel to load multiple results as shown in Figure 3.4.

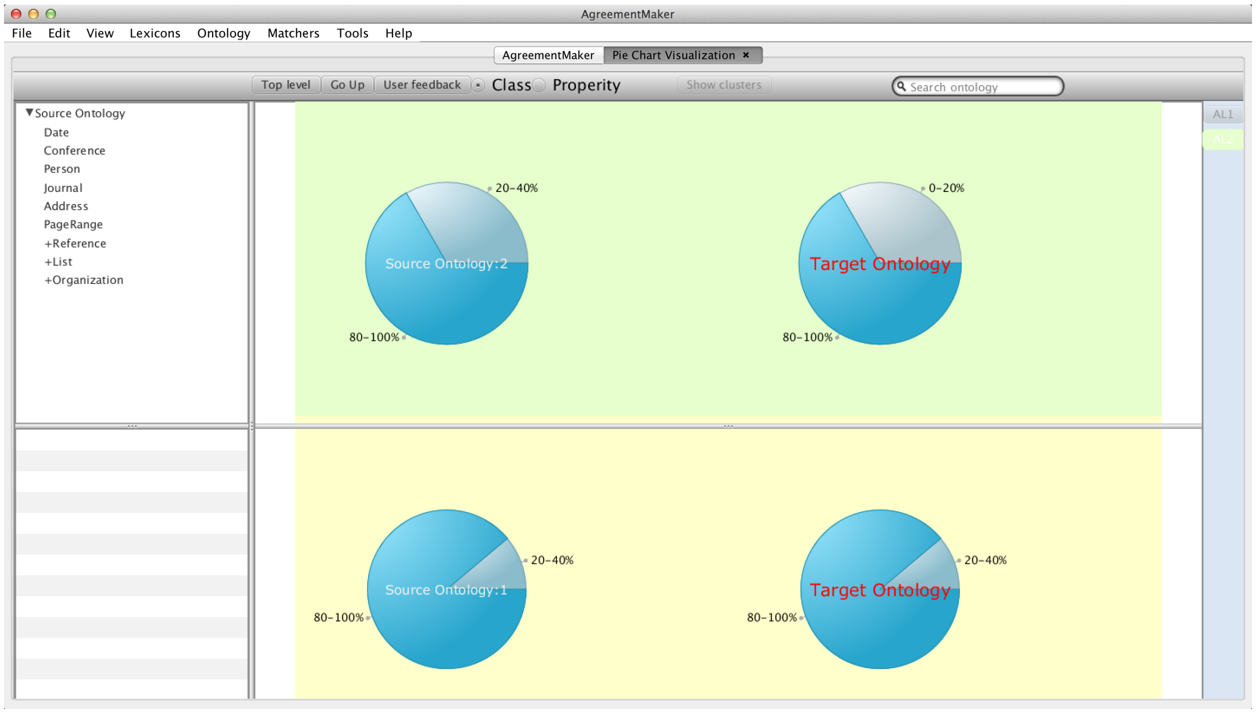


Figure 3.4: Updated main panel.

We are making full use of the containers in JavaFX, in order to manage the elements within the limited space. We use two tile panels to display two sets of ontology pairs, the upper panel is the main panel and the lower one is the sub panel. The tree view shows the siblings of the source ontology in the main panel and the list view shows the children nodes of that ontology.

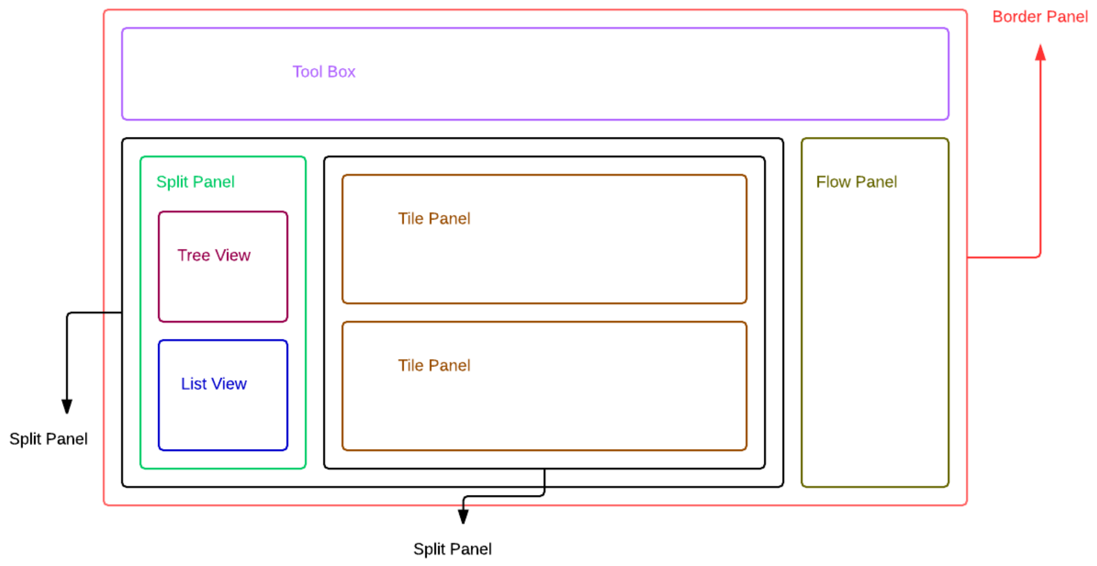


Figure 3.5: Graphical User Interface.

3.4 Interaction with Multiple Ontologies

Figure 3.5 displays the organization of the user interface. We divide the main panel into two sets. The top set in green is the main set and the set below in yellow is the sub set. Each set has a pair of source and target ontologies. The column bar on the rightmost side shows the algorithms we have loaded into the application. The main set shows the result of the second algorithm we load, which is represented by the green button. The rest of the upgraded panel remains the same as described previously.

In this way, we can choose any loaded algorithm to be the main set or the sub set. The difference between the main set and the sub set is that all lists get updated according to the changes to the main set. The reason for this design is that we have limited space on the screen panel and we need to choose the most significant information to display.

When the users click on the main set pie chart slice, all features including all pie charts and the lists will be updated. To change the matching algorithm of the main set, the users first click on the current main set algorithm button, then click on any other algorithm button to set it as the new main set.

Chapter 4

Interactive Ontology Matching

We list here our objectives for an interactive mechanism for matching ontologies that can assist users in a user feedback loop kind of system:

- Show candidate mappings for validation to the users; candidate mappings are determined by automatic matching algorithms.
- Register the validation choices made by the users..
- Allow for class and property navigation to assist users in their validation decisions.
- Allow users to search for a specific class and navigate through the classes.
- Support the creation of new mappings that are missed by the automated process.

4.1 Interactive Workflow

Because automatic matching methods do not always provide complete or correct mappings, the combination of user validation with the automatic methods can lead to better results than the automatic methods alone.

The interactive workflow is shown in Figure 4.1. There exists a user feedback loop (UFL) approach, in that the results provided by the user are fed back into the matching process.

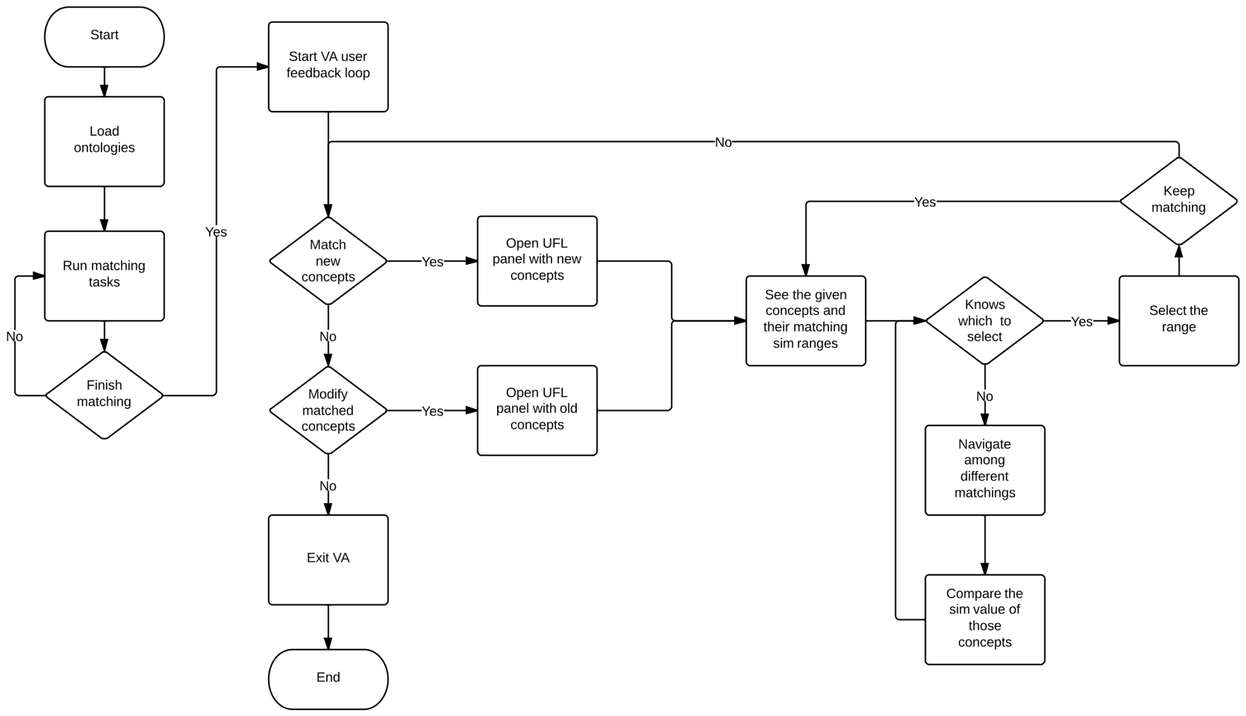
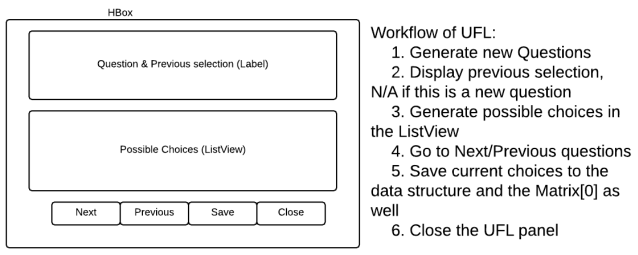

Start
Load ontologies
Run matching tasks
Finish matching
No
Yes
Start VA user feedback loop
Match new concepts
Yes
No
Open UFL panel with new concepts
Modify matched concepts
Yes
No
Open UFL panel with old concepts
Exit VA
End
See the given concepts and their matching sim ranges
Knows which to select
Yes
No
Select the range
Keep matching
Yes
No
Navigate among different matchings
Compare the sim value of those concepts

Figure 4.1: Workflow of the interactive process.

4.2 User Interface

The user interface must display one or more mappings to be validated by users. When showing more than one mapping, users will be asked to choose one among them. The similarity information among classes is the basis for the users to make their decisions, which we will supplement with information about the properties of those classes.

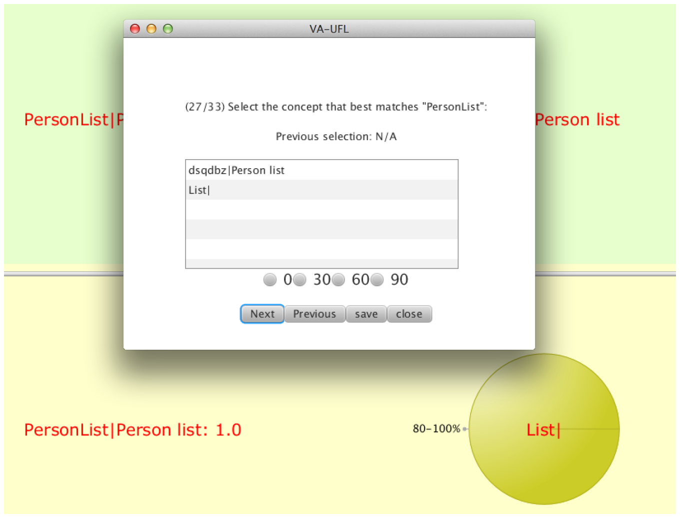
Figure 4.2 shows the design of the panel that enables the intervention of the users.


HBox
Question & Previous selection (Label)
Possible Choices (ListView)
Next
Previous
Save
Close

Workflow of UFL:
1. Generate new Questions
2. Display previous selection, N/A if this is a new question
3. Generate possible choices in the ListView
4. Go to Next/Previous questions
5. Save current choices to the data structure and the Matrix[0] as well
6. Close the UFL panel

Figure 4.2: User interface schematic representation.

Figure 4.3 shows the interface where the users can select their choice when presented with several possibilities.


(27/33) Select the concept that best matches "PersonList":

Previous selection: N/A

dsqdbz|Person list
List|

PersonList|Person list: 1.0

Figure 4.3: Selection of mappings.

4.3 Property Clustering

Falcon-AO [13] is a practical ontology matching system with acceptable to good performance and a number of remarkable features. PBM (Partition-based block matching) is one of the distinguishing features of Falcon-AO and it is a divide-and-conquer method for partition-based block matching that is practically applicable to large class hierarchies. Based on both structural affinities and linguistic similarities, two large class hierarchies are partitioned into small blocks. Then, blocks from different hierarchies are matched based on the relatedness between concepts called *anchors* for which similarity values are high. PBM performs clustering based on structural proximity, for example the distance between classes in the class hierarchy, and the overlapping between the domains of properties of a class. In our user feedback loop strategy when choosing among mappings, we can take the clustering idea as a reference. When comparing two matching results generated by different matching algorithms, we therefore not only provide their matching similarities but also the properties that share the same domain. In this way, users will be able to see more valuable information. PBM generates clusters based on the overlapping between the domains of the properties of both classes.

We provide an example in Figure 4.4. We use the OAEI ontologies for the conference track for the purposes of this testing. We first load two ontologies, edas.owl (source ontology) and ekaw.owl (target ontology) and we discover that there are two possible mappings:

- ConferenceEvent maps to Event with matching confidence 0.79, using the Parametric String matcher.
- ConferenceEvent matches to Conference with matching confidence 0.61, using the Vector-based Multi-word Matcher (VMM) [14].

In the reference alignment the correct matching shows as:

- ConferenceEvent matches to Conference with matching confidence 1.00

Before asking users to make choices, we generate floating panels to show users the properties. As we can see from Figure 4.4, concept *ConferenceEvent* has properties *hasAttendee*, *hasEndDateTime*, *hasLocation*, *hasProgramme*, and *hasStartDateTime*, which have the same domain as ConferenceEvent; concept Event has properties *eventOnList*, *hasEvent*, *heldIn*, *organisedBy*, and *partOfEvent*, which have the same domain as Event; for the *Conference* concept, no properties were found.

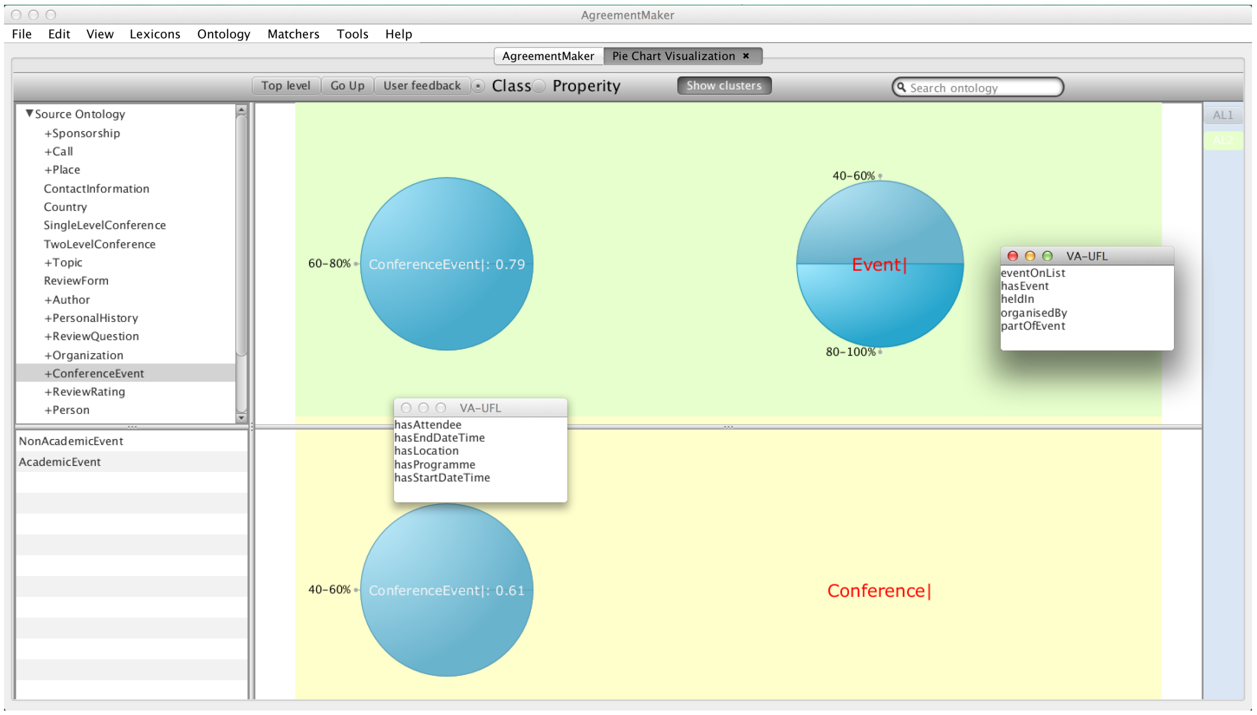
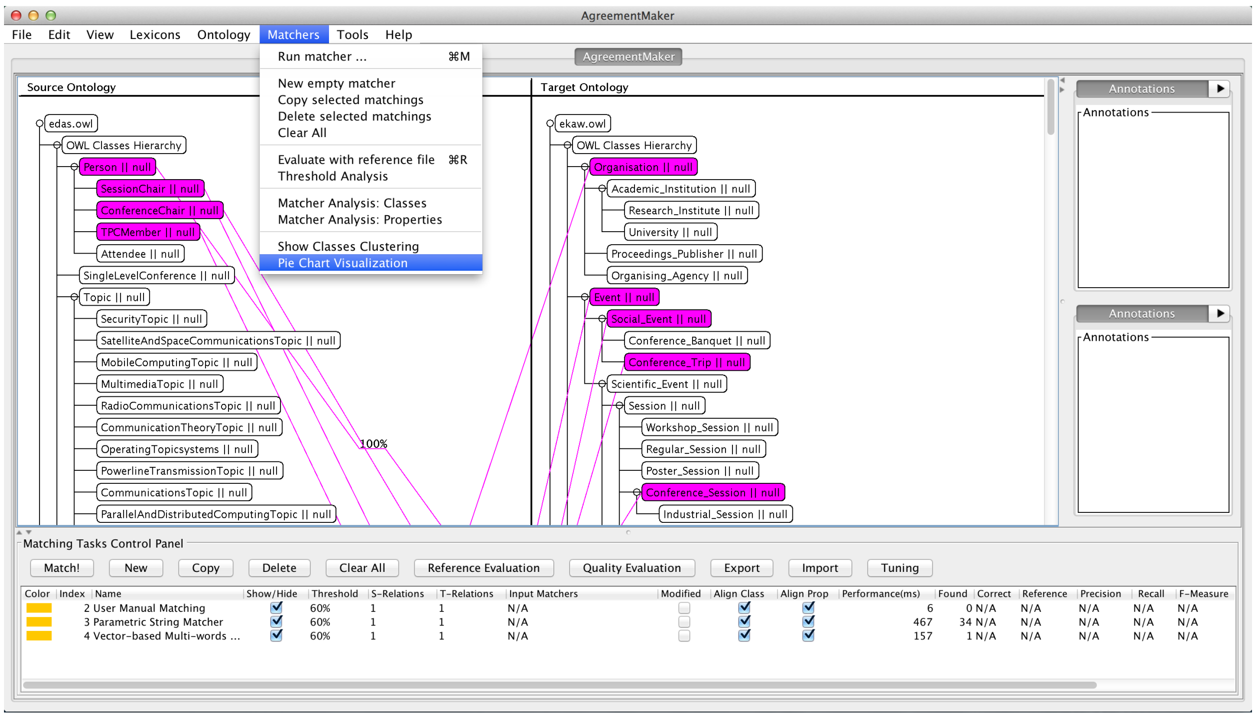


Figure 4.4: Property clustering.

4.4 Integration with AgreementMaker

Using the AgreenmentMaker system, the source and target ontologies are visualized side by side using a tree paradigm. The control panel (see Figure 4.5) allows users to run a variety of matchers. In this example, two automatic matchers have been activated (the Parametric String matcher and the Vector-based Multi-word Matcher) in addition to manual matching. The picture depicts the display of the two ontologies and the mappings obtained in this way.



Color	Index	Name	Show/Hide	Threshold	S-Relations	T-Relations	Input Matchers	Modified	Align Class	Align Prop	Performance(ms)	Found	Correct	Reference	Precision	Recall	F-Measure
🟨	2	User Manual Matching	✓	60%	1	1	N/A	☐	✓	✓	6	0	N/A	N/A	N/A	N/A	N/A
🟨	3	Parametric String Matcher	✓	60%	1	1	N/A	☐	✓	✓	467	34	N/A	N/A	N/A	N/A	N/A
🟨	4	Vector-based Multi-words ...	✓	60%	1	1	N/A	☐	✓	✓	157	1	N/A	N/A	N/A	N/A	N/A

Figure 4.5: Integration with AgreementMaker.

Every set of mappings in AgreementMaker is represented by the MatchingTask class. The MatchingTask class contains the following elements: a matching algorithm, its associated parameters (e.g., the similarity threshold), and the mappings produced by the execution of the matcher.

To open our visualization system, users can select the "Pie Chart Visualization" tab of the drop down menu, as shown in the Figure. The key point in the integration of the pie chart visualization with AgreementMaker is that we pass all the MatchingTask instances from AgreementMaker to the pie chart visualization.

After selecting the "Pie Chart Visualization" tab, another tab will show up and the pie charts will be initialized automatically (see Figure 3.4). Users can easily switch between the tree and pie chart visualizations by clicking on the available tabs at the top of the display.

Chapter 5

Conclusions and Future Work

In our project we devised a visualization method for large ontology matching that has the following features:

1. Representation Dimension

   - Visual representation for source and target ontologies and for mappings between classes in those ontologies.
   - Focus on a small part of the ontologies at a time allowing to quickly access other parts of the ontologies.
   - Display of the similarity values between classes and provide an overview of the similarity values for the children of those classes.

2. Interaction Dimension

   - User-driven navigation of classes and properties.
   - Ability to search for a specific class and to traverse the ontologies vertically (parents and children of a class) and horizontally (siblings of a class) .

3. Analysis and Decision Making

   - Display several possible mappings to the users, so that they can choose among them, as part of a user feedback loop strategy that combines automatic with manual matching methods.

- Manages the mapping decisions made by the users, by keeping track of them.

4. Implementation

    - Uses JavaFX.
    - Integrates with AgreementMaker, therefore can be used in conjunction with other visualization methods.

Clearly, there are several directions for future work. The first one is that we would like to extend the comparison of matching algorithms to more than two at a time. Figure 5.1 shows our preliminary design to compare more than two algorithms. Each node in the structure is a pie chart that can be traversed recursively. The center pie chart represent the current class and the outer pie charts are the matching results of different matching algorithms. We connect the center and outer charts with a line: the longer the line, the smaller the matching confidence value.

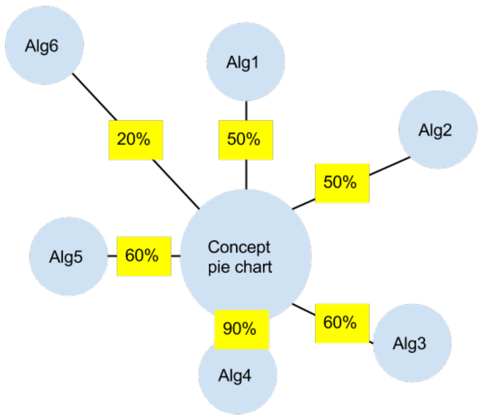


Figure 5.1: Comparison of several matching algorithms.

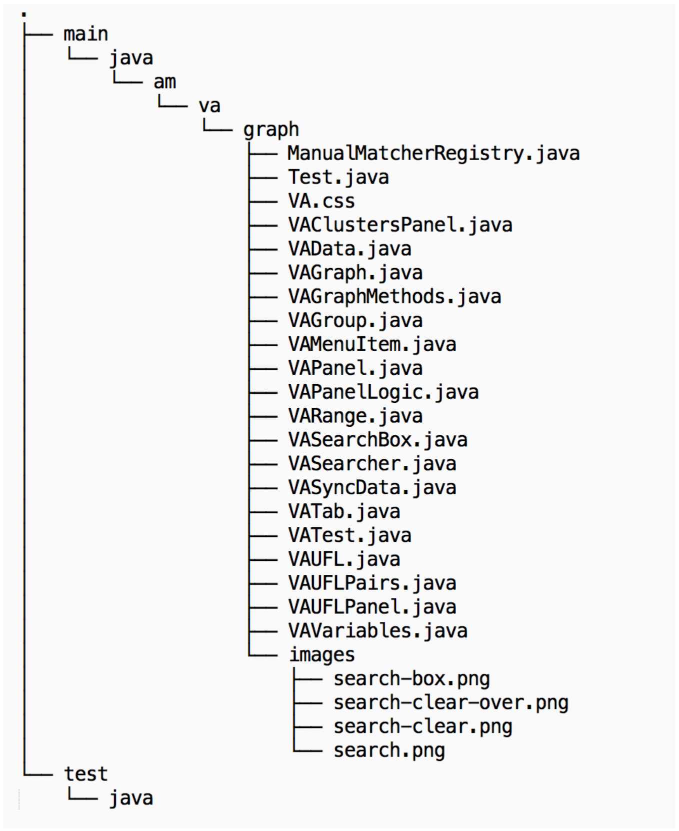
It may be the case that no single visualization strategy works separately, especially for very large ontologies. AgreementMaker already provides several different strategies [2, 4]. Experiments would be needed to determine the usability and effectiveness of the different strategies when used separately or in coordination with one another.

Appendix A

Additional Information

A.1 File list

This project is a branch of the AgreementMaker, see Figure A.1 for the file list.



```
.
├── main
│   └── java
│       └── am
│           └── va
│               └── graph
│                   ├── ManualMatcherRegistry.java
│                   ├── Test.java
│                   ├── VA.css
│                   ├── VAClustersPanel.java
│                   ├── VAData.java
│                   ├── VAGraph.java
│                   ├── VAGraphMethods.java
│                   ├── VAGroup.java
│                   ├── VAMenuItem.java
│                   ├── VAPanel.java
│                   ├── VAPanelLogic.java
│                   ├── VARange.java
│                   ├── VASearchBox.java
│                   ├── VASearcher.java
│                   ├── VASyncData.java
│                   ├── VATab.java
│                   ├── VATest.java
│                   ├── VAUFL.java
│                   ├── VAUFLPairs.java
│                   ├── VAUFLPanel.java
│                   ├── VAVariables.java
│                   └── images
│                       ├── search-box.png
│                       ├── search-clear-over.png
│                       ├── search-clear.png
│                       └── search.png
└── test
    └── java
```

Figure A.1: File list

Bibliography

[1] J. Euzenat and P. Shvaiko, *Ontology Matching.* Heidelberg (DE): Springer-Verlag, 2007.

[2] I. F. Cruz, F. Palandri Antonelli, and C. Stroe, "AgreementMaker: Efficient Matching for Large Real-World Schemas and Ontologies," *PVLDB*, vol. 2, no. 2, pp. 1586–1589, 2009.

[3] ——, "Efficient Selection of Mappings and Automatic Quality-driven Combination of Matching Methods," in *ISWC International Workshop on Ontology Matching (OM)*, ser. CEUR Workshop Proceedings, vol. 551, 2009, pp. 49–60.

[4] I. F. Cruz, C. Stroe, and M. Palmonari, "Interactive User Feedback in Ontology Matching Using Signature Vectors," in *IEEE International Conference on Data Engineering (ICDE).* IEEE, 2012, pp. 1321–1324.

[5] F. Shi, J. Li, J. Tang, G. Xie, and H. Li, "Actively Learning Ontology Matching via User Interaction," in *International Semantic Web Conference (ISWC)*, ser. Lecture Notes in Computer Science, vol. 5823. Springer, 2009, pp. 585–600.

[6] E. Bertini and D. Lalanne, "Investigating and Reflecting on the Integration of Automatic Data Analysis and Visualization in Knowledge Discovery," *SIGKDD Explorations Newsletter*, vol. 11, no. 2, pp. 9–18, May 2010.

[7] V. Ivanova and P. Lambrix, "User Involvement for Large-Scale Ontology Alignment," in *International Workshop on Visualizations and User Interfaces for Knowledge Engineering and Linked Data Analytics*, 2014.

[8] M. Lanzenberger and J. Sampson, "AlViz - A Tool for Visual Ontology Alignment," in *Conference on Information Visualization (IV).* IEEE Computer Society, 2006, pp. 430–440.

[9] D. Faria, C. Pesquita, E. Santos, M. Palmonari, I. F. Cruz, and F. M. Couto, “The AgreementMakerLight ontology matching system,” in *International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE)*. Springer, 2013, pp. 527–541.

[10] C. Pesquita, D. Faria, E. Santos, J.-M. Neefs, and F. M. Couto, “Towards Visualizing the Alignment of Large Biomedical Ontologies,” in *Data Integration in the Life Sciences (DILS)*, 2014.

[11] S. M. Falconer and M.-A. D. Storey, “A Cognitive Support Framework for Ontology Mapping,” in *International Semantic Web Conference/Asian Semantic Web Conference (ISWC/ASWC)*, ser. Lecture Notes in Computer Science, vol. 4825. Springer, 2007, pp. 114–127.

[12] Z. El Jerroudi and J. Ziegler, “iMERGE: Interactive Ontology Merging (poster),” in *International Conference on Knowledge Engineering and Knowledge Management Knowledge Patterns (EKAW)*, 2008.

[13] W. Hu and Y. Qu, “Falcon-AO: A Practical Ontology Matching System,” *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 6, no. 3, pp. 237–239, 2008.

[14] I. F. Cruz, F. Palandri Antonelli, C. Stroe, U. Keles, and A. Maduko, “Using AgreementMaker to Align Ontologies for OAEI 2009: Overview, Results, and Outlook,” in *ISWC International Workshop on Ontology Matching (OM)*, ser. CEUR Workshop Proceedings, vol. 551, 2009, pp. 135–146.