Visualization for Large Concept Matching

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:

Abstract

In this project, we integrated AgreementMaker [1] with JavaFX seamlessly and developed a way to visualize matching results of ontology with large size. We provide both general and detailed view of matching results to users. User is able to navigate through different ontology levels, search and see the structure of the source and target ontology pairs. This project also involves the issue of user intervention instead of only focusing on matching algorithms to compute candidate mappings, we provide user feedback loop with property clustering to help user makes decisions.

Acknowledgment

I would like to thank my guides Professor Isbel Cruz and Cosmin Stroe, who had given me much help throughout the whole project.

Y.L.

Contents

Chapter 1

Introduction

An ontology typically provides a vocabulary describing a domain of interest and a specification of the meaning of terms in that vocabulary. More corporations have started using ontology for storing their knowledge. However, more than one ontology can be used for the same knowledge. To address this issue, one has to either devise a mechanism to merge the ontology automatically or, let human to do the matching manually. That is what we called ontology matching, and it is the process of finding the relationship between concepts/properties between different ontology. In this project we focus on the concept matching.

Different techniques have been examined in concept matching. For example the string/language based algorithm, the structural based algorithm, the combination strategies and filtering techniques. In this project we will not focusing on any particular matching algorithm, but focusing on visualizing the ontology matching process and the methods to involve human in the matching process to help get a better mapping.

We decompose this project into two parts: visualization and visual analytics. To some point, good visualization can be regarded as a prerequisite of effective analytics. Some previous work has been done on the visualization part, but most of the two parts are challenging and remain undeveloped, which makes this project interesting.

1.1 Visualization

Visual representation of large scaled data helps people understand and analyze information or process. Here it is very necessary to point out that this project is doing ontology matching visualization, instead of pure ontology visualization. There has been much work done on visualizing ontology structures, while visualizing ontology matching results are barely found. This might be due to the complex relationship between source and target ontology structures. Moreover, another focus is that the visualization is applied to large ontology, which has little work been done so far. Simply showing a list or a tree structure of ontology does little help when the size of the ontology tend to be very large, due to the limitation of memory spaces. To better compare and analyze the matching results, a better representation of data is required.

1.2 Visual Analytics

Visual Analytics is the science of analytical reasoning supported by interactive visual interfaces [2]. 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. Visual Analytics often involves the similarity matrices. A similarity matrix is built for each pair of concepts, using the Linear Weighted Combination (LWC) matcher, which processes the weight average for the different similarity results [3]. Assume that the source and target ontology have size M and N, then the similarity matrix is of size $M \times N$. When M and N are large, the matrix becomes more complicated. Additionally, it would be much more challenging if multiple matching algorithms are involved in the analytics.

1.3 User Feedback Loop

This project involves the issue of user intervention instead of only focusing on matching algorithms to compute candidate mappings. Ontology is represented by language and languages are known to be ambiguous. Compared to machines, human can better match those concepts by using detailed knowledge about the world.

This project aims at improving the matching results instead of improving the matching algorithms.

Chapter 2

Related Work

Although little, there has already been some work on ontology matching visualization [4]. Each of them has advantages of presenting information, also some unsatisfying features. In the following part we will introduce and analyze each method briefly.

2.1 Cluster representation

The cluster representation [5] does a good job in showing both detailed and general information of matching results. Users can select the level they want to cluster for the results. However, this method is severely constrained by its computation complexity, which is $O(n^3)$ where n is the size of the ontology. Furthermore, it can only be applied in one algorithm, when multiple algorithms are used, colors indicating the matching confidence might be confusing.

2.2 Treemap view

Treemap view [4] of ontology matching does well in showing much information in a certain area. Besides that, detailed matching information is provided in another small window. However, when ontology tends to be large, the lower the level, the smaller the area, and the information can not be conveyed clearly. In the contrast, upper level ontology usually have smaller number than their children, but they take more area, which does not reasonably deals with the problem.

2.3 Matrix representation

This matrix representation [6] surpasses the aforementioned two methods because it makes full use of space and color information, showing as the following figure. The color indicating each matching confidence actually works as the third dimension without adding more space. Problem occurs when the size of the similarity matrix grows with ontology size. Also, because of the sparsity of the similarity matrix, showing them all is actually a waste of space.

Chapter 3

Visualization

3.1 Prototype

After problem analyzing and related approach investigation, below are the summarized criteria for this project:

- Display both general and detailed information of matching results
- Ontology navigation, exploration and searching
- Intuitive display for large ontology
- Ability to reveal the difference between matching algorithms
- User feedback acquisition

Based on the criteria above, we developed the prototype in the first stage.

The main principle of this design is for large ontology, we have to reduce the amount of data every time we represent information to users. In other word, we need to present the most valuable data within a certain size. So we consider to provide general processed information instead of raw results. Further more, consider that users do not need so much information at one time, we show the results level by 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 result they need.

The information we use here is the matching similarity confidence, represented by a pie chart. The reason of using the pie chart is that, no matter how large the data size is, the pie chart requires only a small area. In addition, this feature proved to be an advantage then comparing performances of different algorithms.

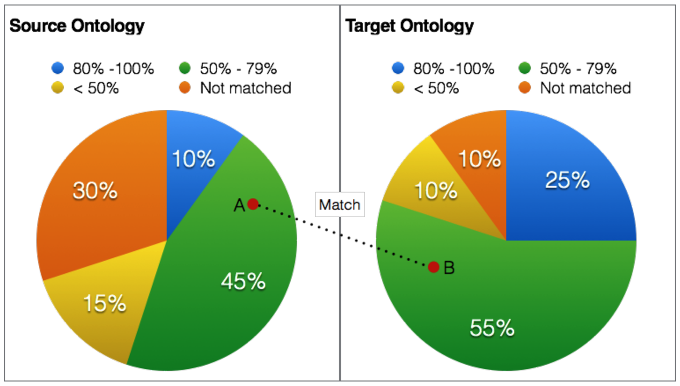


Figure 3.1: Prototype mathcing result interface

As shown in Fig.3.1, we divide the similarity into different ranges. For a certain source node *S*, we show the percentage of its children's matching confidence that fall in the corresponding range. In the right pie chart, matched target node *T* of *S* shows its children's matching results in the same way.

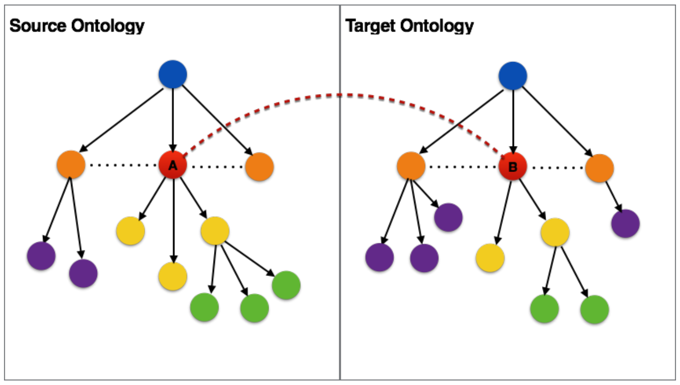


Figure 3.2: Prototype structure interface

Besides the general information, we also provide detailed information about matching results and ontology structures. To deal with large ontology, we make use of the fact that users tend to pay more attention to structure-close ontology. So we only show the parent, children and siblings of the current node in view. This is clearly illustrated by Fig.3.2.

The next question is how to combine the general and detailed view together. In order to let users easily see both two views, we provide a list represents the children in range and a treeview represents the siblings of the current node.

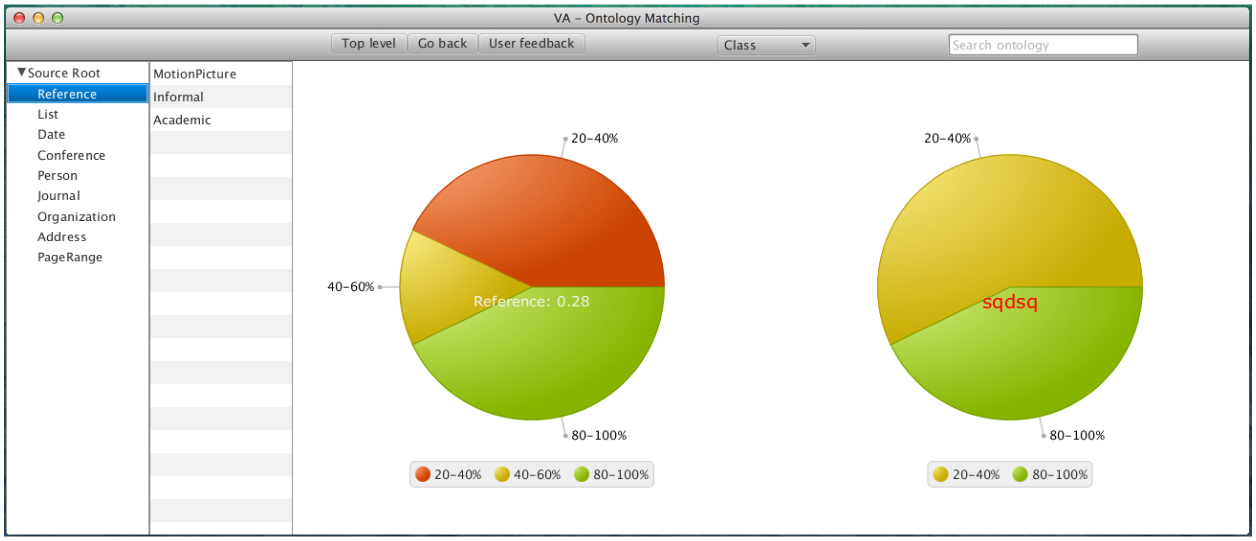


Figure 3.3: Main panel

Based on the prototype we developed the first version of this application which is shown in Fig.3.3.

3.2 Application Workflow

Fig.3.3 shows the major user interface of this application. Same as the prototype we discussed above, in the center area there are two pie charts, representing the matching result distribution of source and target ontology separately.

Left to the pie charts there is a list. When users click on a pie chart slice, it shows the matching ontology nodes with matching confidence falling in the sliced area, ordered by similarity confidence. Clicking on a node in the list leads to an update of the pie charts. The leftmost part is a tree view. It shows all siblings of the current node. When clicking on a node in the tree view, both the pie charts and the lists will be updated. On the top right there is a search box. When entering the name of the node and press enter key on the keyboard, the left pie chart will show the target node and everything on the panel will be updated correspondingly. Last but not the least, for better navigation we provide additional functions such as “go to the top level”, “go to the previous level” and “switching between class and property”.

To view the application workflow please see Fig.3.4

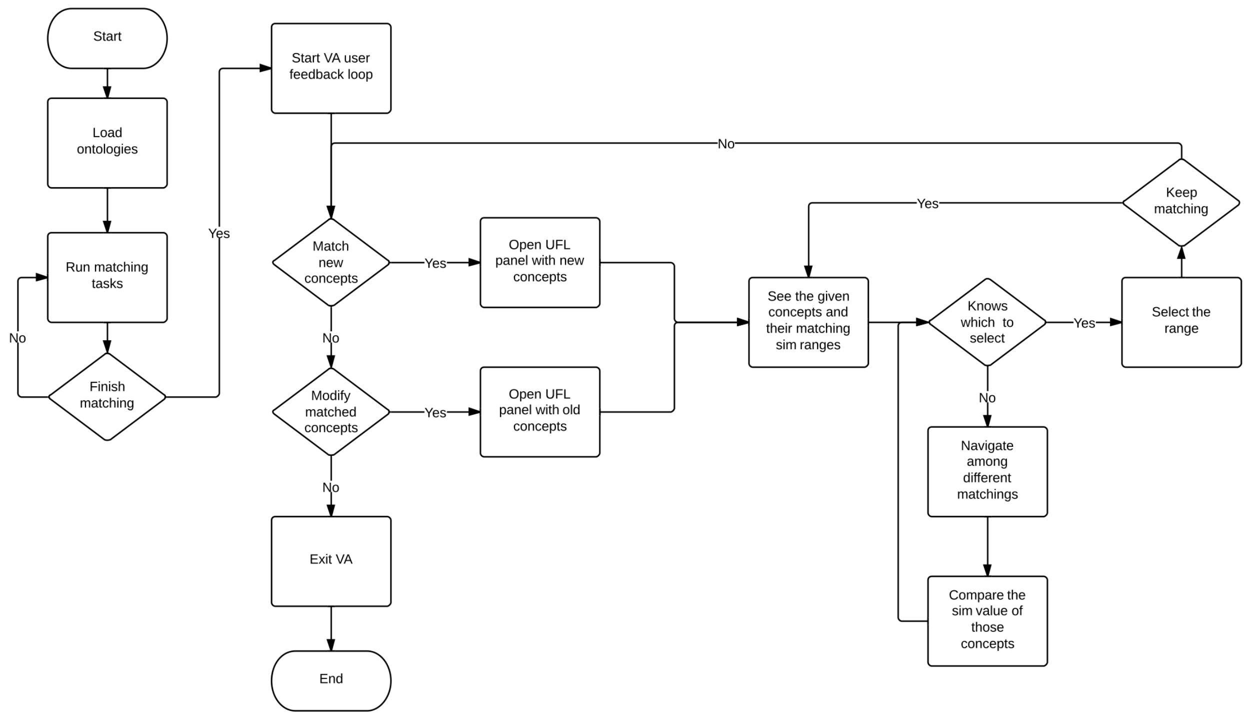

Start
Load ontologies
Run matching tasks
Finish matching
No
Yes
Start VA user feedback loop
No
Match new concepts
Yes
Open UFL panel with new concepts
No
Modify matched concepts
Yes
Open UFL panel with old concepts
No
Exit VA
End
See the given concepts and their matching sim ranges
Knows which to select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

Figure 3.4: Workflow

3.3 Graphic User Interface

We are making full use of the containers in JavaFx, in order to manage the elements within the limited space. See Fig.3.5 for the graphic user interface design. 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 (this will be talked in more detail in the next chapter). The treeview shows the siblings of the source ontology in the main panel and the listview shows the children nodes of a particular portion which user clicks on of the source ontology.

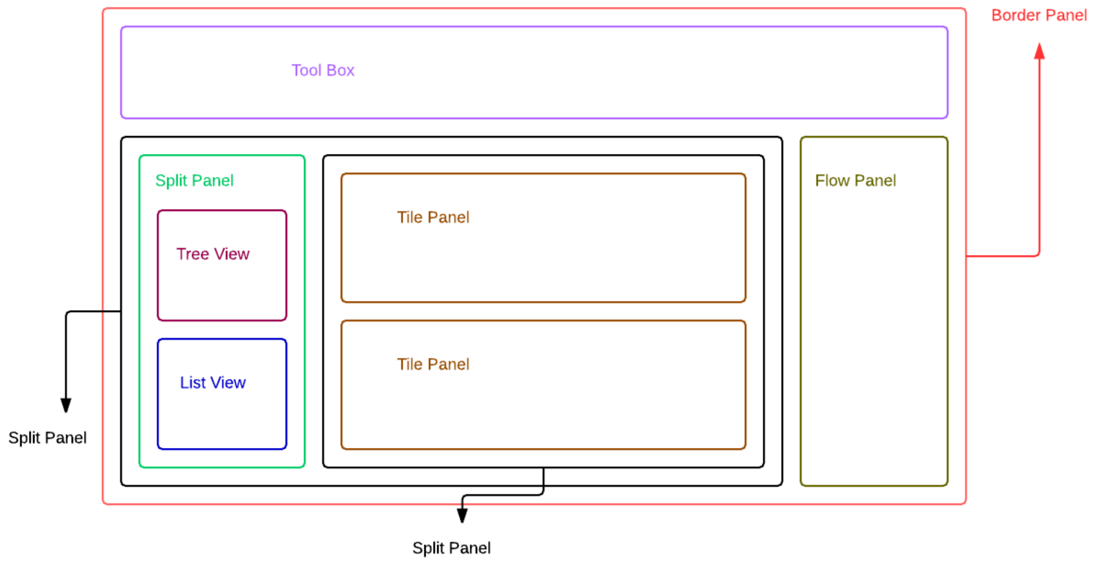


Figure 3.5: GUI

Chapter 4

Ontology Comparsion

4.1 Loading Multiple Ontology

In our prototype, we design the application for displaying one matching result, in order to compare different algorithm performances, we need to load multiple results at one time. Here we upgrade the main panel for loading multiple results, which is shown in Fig. 4.1.

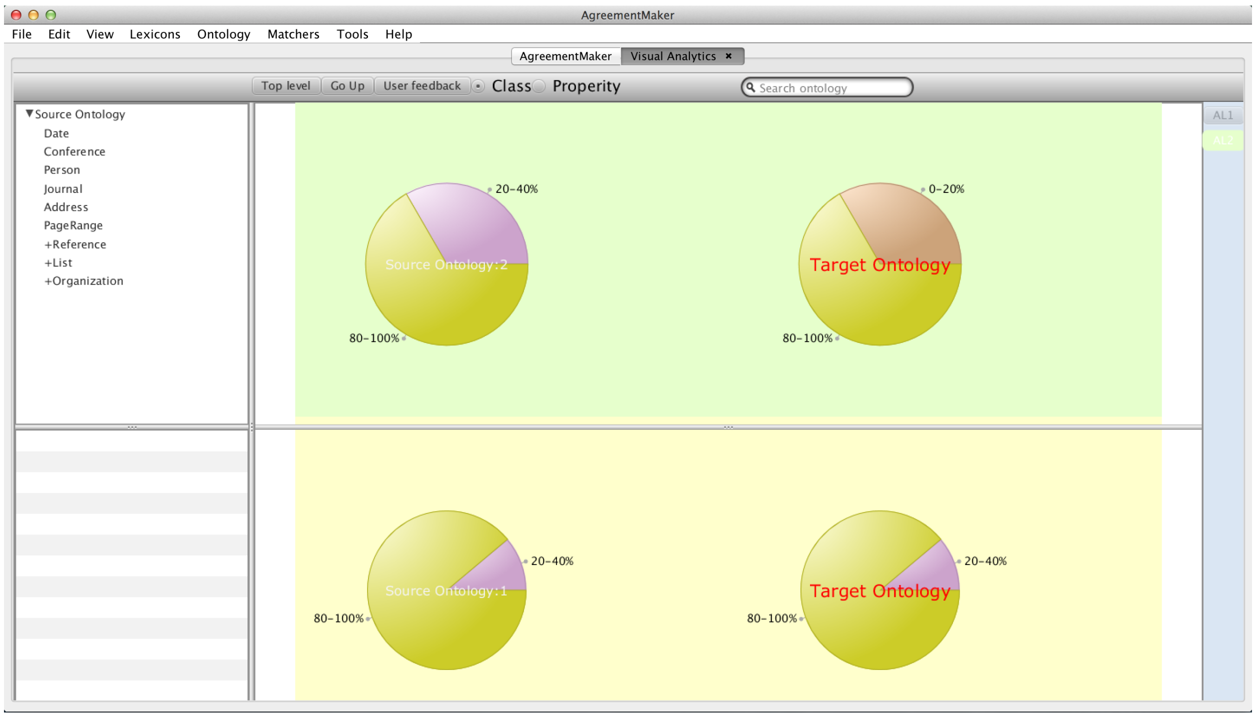


Figure 4.1: Updated Main panel

We divide the main panel into to two sets. The top set in green is the main set and the below set in yellow is the subset. Each set has a pair of source and target ontology. The column bar on the rightmost shows the algorithms we have loaded into the application. The main set shows the result of the second algorithm we load, which represented by the green button. The rest of the upgraded panel remains the same.

In this way, we can choose any loaded algorithm to be the main set or the subset. The difference between main set and subset is that all lists update according to the change of 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.

4.2 Comparing and Switching

As we mentioned above, when users click on the main set pie chart slice, all features including all pie charts and the lists will be updated. For switching the algorithm of the main set, 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 5

User Feedback Loop

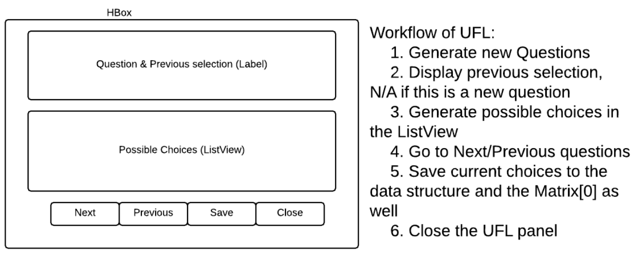
According to [4], we come up with the basic principles for the user feedback loop.

- Provide ambiguous mappings to users, the application will detect the most ambiguous alignments and show them to users.
- Let users internally make mapping decisions and mark the decisions that users already made.
- User-driven navigation of concepts and properties for comparing.
- Search for a specific concept and to go back and forward for choosing the ambiguous alignments.
- Users can manually create mappings that are missed by the automated process.

5.1 Ambiguous Selection

In AgreementMaker, mappings are stored in a matrix data structure. We gather all the mapping matrices generated by different algorithms and select all different alignments. Those alignments are the ambiguous selections. We show the ambiguous selections with the matching confidence information to users and let them choose the best matching ones. Notices that even accurate matching algorithms cannot avoid incorrect matching, that is the reason to ask users to make decisions on ambiguous selections.

Fig.5.1 shows the design of the URL panel and its workflow.

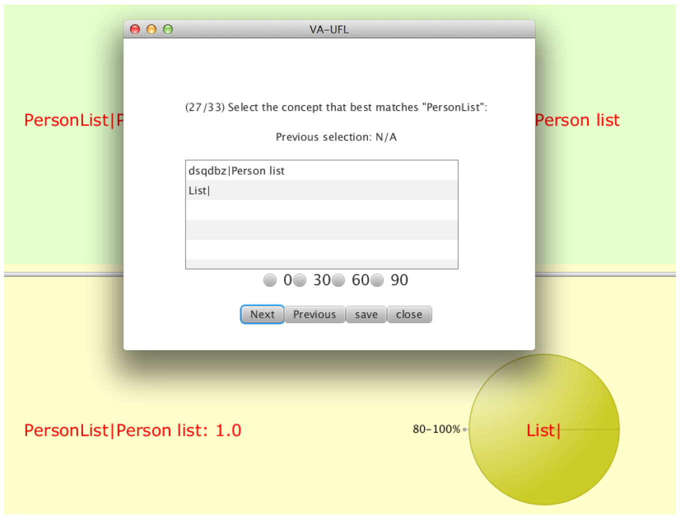

HBox
Question & Previous selection (Label)
Possible Choices (ListView)
Next
Previous
Save
Close

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 5.1: UFL design

Whenever the matching process finished, user is able to click on the UFL button and begin to see the ambiguous matching, then decide the better matching based on their own knowledge.



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

Previous selection: N/A

dsqdbz|Person list
List|

Figure 5.2: Ambiguous Selection

5.2 Property Clustering

Falcon-AO [7] 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 respectively. Then, blocks from different hierarchies are matched based on the relatedness between them via anchors. PBM do the clustering based on the structural proximity, for example the distance between classes in the class hierarchy, and the overlapping between the domains of properties. In our user feedback loop, we can take the clustering idea as a reference. When comparing two matching results generated by different matching algorithms, we not only provide the matching similarities but also the properties that share the same domain of the two concepts. In this way, users will be able to see more valuable information. PBM generates clusters based on the overlapping between the domains of properties, in our user feedback loop we use the same idea to make clusters to help user better match the concepts by using their knowledge.

See Figure 5.3 for example. We use the anatomy datasets for testing purpose. We first load two ontologies, edas.owl (source ontology) and ekaw.owl (target ontology) and there is an ambiguous matching:

- ConferenceEvent matches to Event with matching confidence 0.79 (matcher: Parametric String)
- ConferenceEvent matches to Conference with matching confidence 0.61 (matcher: Vector-based Multi-words)

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 5.3, concept ConferenceEvent has properties hasAttendee, hasEndDateTime, hasLocation, hasProgramme and hasStartDateTime, those properties have the same domain as ConferenceEvent; concept Event has properties eventOnList, hasEvent, heldIn, organisedBy and partOfEvent, those properties have the same domain as Event; for concept Conference, there is no properties found.

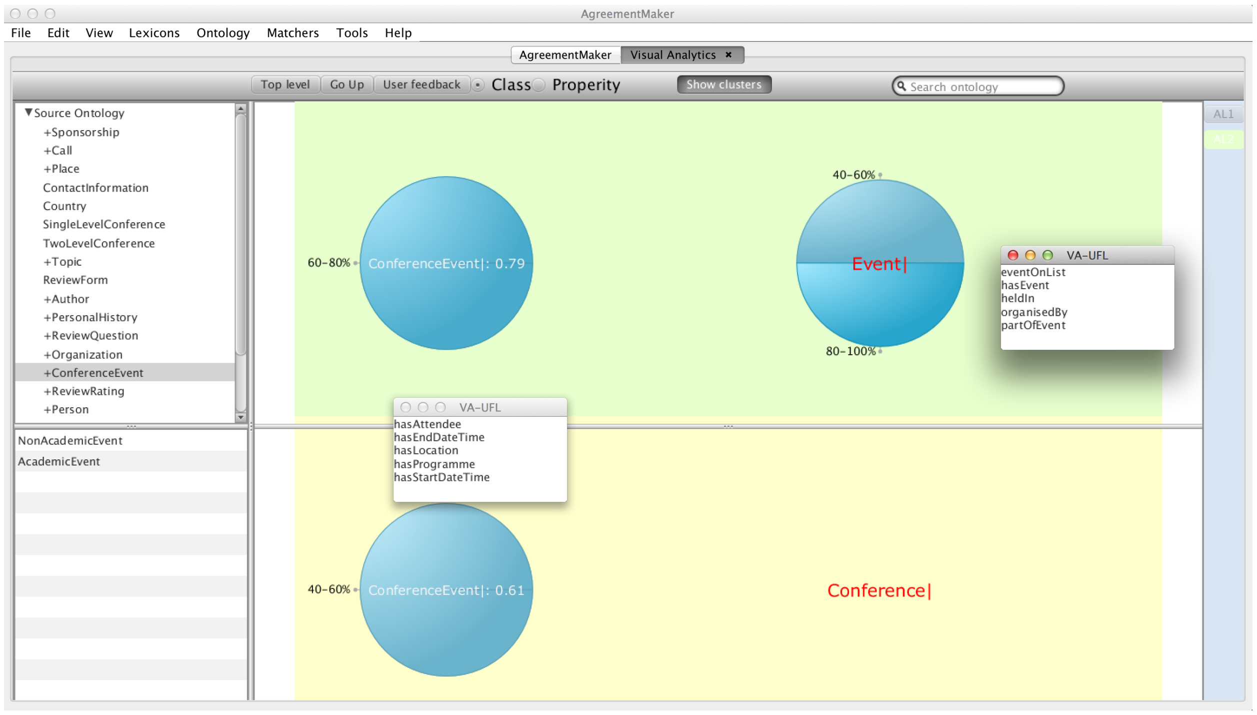
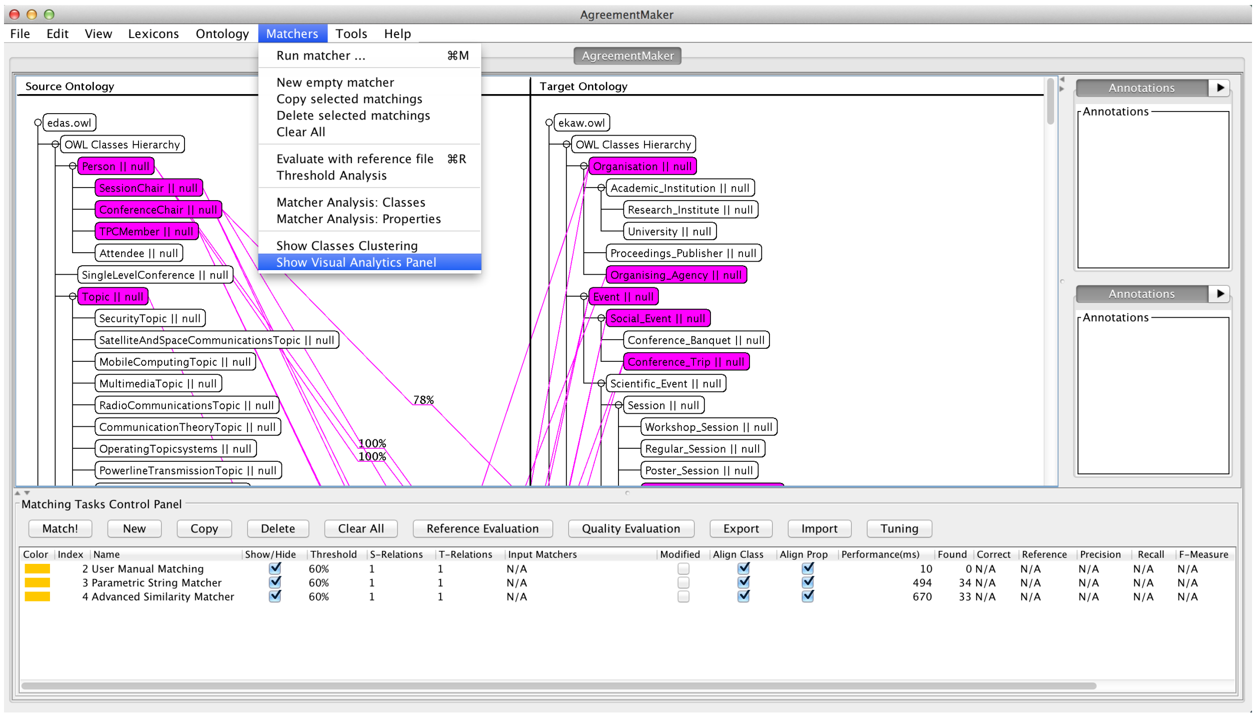


Figure 5.3: Property clustering

5.3 Integrating with AgreementMaker

In AgreementMaker, after executing the loaded algorithms, we first select the mapping result matrix with the highest matching similarity confidence. We make a copy of this matrix and regard it as the user defined matrix. Then we start the user feedback loop process and modify the user defined matrix with user selected alignments.



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	☐	✓	✓	10	0	N/A	N/A	N/A	N/A	N/A
🟨	3	Parametric String Matcher	✓	60%	1	1	N/A	☐	✓	✓	494	34	N/A	N/A	N/A	N/A	N/A
🟨	4	Advanced Similarity Matcher	✓	60%	1	1	N/A	☐	✓	✓	670	33	N/A	N/A	N/A	N/A	N/A

Figure 5.4: Integrating with AM

Chapter 6

Limications and Future works

This application is able to compare between different algorithm performances. However, it displays two algorithms each time. When users need to compare multiple algorithms they need to switch between algorithms using buttons. The current design is not able to display multiple algorithms at one time due to space limitation of the screen.

In order to solve the problem above, we can use the structure shown in Fig.6.1.

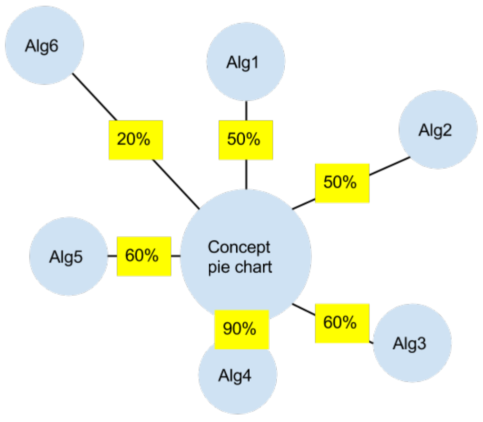


Figure 6.1: Updated Structure

Each node in the structure is a piechart with recursion properties. The center piechart represent the current concept and the outer piecharts are the matching results of different matching algorithms. We connect the center and outer charts with a line and the longer the line, the smaller the matching confidence value. We came up with this design to solve the "display multiple matchers" issue. For the visualization part this is able to display more matchers at a time.

Chapter 7

Conclusion

Compare with the criteria we proposed previously, this application fulfills all of our standards as follows:

1. Representation Dimension
   - Provide a visual representation for source and target ontologies
   - Focus on a small part of the ontologies but also able to show the overview.
2. Analysis and Decision Making
   - Provide ambiguous mappings for users, the system will detect the most ambiguous alinements made by different algorithm and show them to users.
   - Let users internally make mapping decisions and mark the decisions that users already made.
3. Interaction Dimension
   - User-driven navigation of concepts and properties.
   - Searching for a specific concept and the abilities to go back and forward.

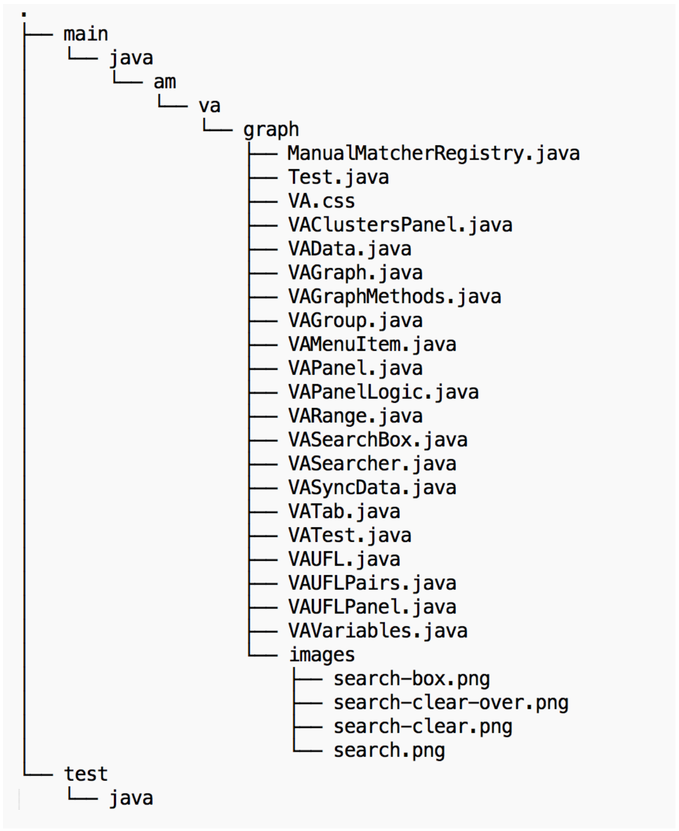
Additionally, nice interface of JavaFX also provides good user experience. However, there still remains much potential of improvement.

Appendix A

Additional Information

A.1 File list

This project is a branch of the AgreementMaker, see Fig.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

A.2 Main functions

- Load data files that generated by AgreementMaker.
- Display visualization for multiple algorithms, divide the screen into N parts, each of which represent a matching result for one algorithm.
- Generate a list of ambiguous concepts/properties and their alignments, once user runs the user feedback loop, iterate that list and let user decide which one to match.
- One to many matching aliments are provides, it means that user is able to choose from multiple concepts that could be matched to the given one. Matching is represented by percentage.
- Assign the new matching confidence selected by user to the ontology matrix and send data back to AgreementMaker.

Bibliography

[1] Adivs.Laboratory, "AgreementMaker Official Site," http://agreementmaker.org/, [Online; AgreementMaker].

[2] E. Bertini and D. Lalanne, "Investigating and reflecting on the integration of automatic data analysis and visualization in knowledge discovery," *ACM SIGKDD Explorations Newsletter*, vol. 11, no. 2, pp. 9–18, 2010.

[3] I. F. Cruz, F. P. Antonelli, and C. Stroe, "Agreementmaker: Efficient matching for large real-world schemas and ontologies," *Proc. VLDB Endow.*, vol. 2, no. 2, pp. 1586–1589, Aug. 2009. [Online]. Available: http://dx.doi.org/10.14778/1687553.1687598

[4] S. M. Falconer and M. Storey, "A cognitive support framework for ontology mapping," in *The Semantic Web*. Springer, 2007, pp. 114–127.

[5] M. Lanzenberger and J. Sampson, "Alviz - a tool for visual ontology alignment," in *In IV '06: Proceedings of the conference on Information Visualization*. IEEE Computer Society, 2006, pp. 430–440.

[6] I. F. Cruz, C. Stroe, and M. Palmonari, "Interactive user feedback in ontology matching using signature vectors," in *Data Engineering (ICDE), 2012 IEEE 28th International Conference on*. IEEE, 2012, pp. 1321–1324.

[7] 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.