Alignment Results of Anchor-Flood Algorithm for OAEI-2008

Md. Hanif Seddiqui and Masaki Aono

Toyohashi University of Technology, Aichi, Japan
hanif@kde.ics.tut.ac.jp, aono@ics.tut.ac.jp

Abstract. Our proposed algorithm called *Anchor-Flood algorithm*, starts off with *anchors*. It gradually explores concepts by collecting neighbors in concept taxonomy, thereby taking advantage of *locality of reference* in the graph data structure. Then local alignment process runs over the collected small blocks of concepts. The process is repeated for the newly found aligned pairs. In this way, we can significantly reduce the computational time for the alignment as our algorithm concentrates on the aligned pairs and it resolves the scalability problem in ontology alignment over large ontologies. Through several experiments against OAEI-2008 datasets, we will demonstrate the results and the features of our Anchor-Food algorithm.

1 Presentation of the system

The Anchor-Flood algorithm is mainly designed targeting to align two large scale ontologies or one large scale and another small scale ontologies effectively. It does not compare an entity against all the entities in other ontology. The way of selecting the group of entities to be compared is the novelty of our algorithm. Our algorithm operates quite faster over large ontologies as observed in aligning anatomy ontologies and it is depicted in Table 2.

1.1 State, purpose, general statement

The purpose of our Anchor-Flood algorithm is basically ontology matching. However, we used our algorithm in patent mining system to classify a research abstract in terms of International Patent Classification (IPC). Containing mostly general terminologies leads classifying an abstract a formidable task. Automatic extracted taxonomy of related terms available in an abstract is aligned with the taxonomy of IPC ontology with our algorithm succesfully. We also start using the Anchor-Flood in the focus-oriented biomedical applications which generally contain very large ontologies.

To be specific, we only describe our Anchor-Flood algorithm and the results against OAEI 2008 datasets here. For more details, we refer the reader to our semantic website : http://www.kde.ics.tut.ac.jp/~hanif. More elaborate information will be come out soon in our semantic technology geared website.

1.2 Specific techniques used

We implemented Anchor -Flood algorithm in java. Our algorithm contains preprocessing, adaptation module for OAEI 2008, the basic block of algorihtm and the local alignment process.

We created our own persistent model of ontology, as our algorithm requires optimal graph structure of concept taxonomy along with other non-trivial structural and simple lexical information. To collect the necessary information in repository, we use the ARP triple parser of jena module. Fig 1 shows the basic block of Anchor-Flood algorithm to comprehend easily. However, it has complex process of collecting small blocks of concepts and related properties dynamically.

As a part of preprocessing, we also normalize the lexical information and extract the derivative relations, like inherited restrictions etc.

The basic part of Anchor-Flood algorithm is depicted in Fig. 1. Starting off an anchor, Anchor-Flood algorithm collects neighboring concepts which includes super concepts, siblings and subconcepts of certain depth to form a pair of blocks across ontologies, as the neighbors of similar concepts might also be similar [5]. Local alignment process aligns concepts and their related properties based on lexical information [2, 7, 8], semantic information [4] and structural relations [1, 3, 4]. Found aligned pairs are considered for further processing. Hence, it burst out with a pair of aligned block in a compacked part of the ontologies, giving the taste of segmentation [6].

Multiple anchors from different part of ontologies confirm a fair collection of aligned pairs as a whole.

1.3 Adaptations made for the evaluation

The Anchor-Flood algorithm needs an anchor to start off. Therefore, we used another tiny program module, which is capable of extarcting some probable aligned pairs as anchors. The tiny program is attached inside along with our basic algorithm to produce a system. It uses lexical information and some statistical relational information to extract a small number of aligned pairs from different part of ontologies. The program is essentially small, simple and faster. We also removed the subsumption module of our algorithm to make it more faster.

1.4 Link to the system and parameters file

The version of Anchor-Flood for OAEI-2008 can be downloaded from our website: http://www.kde.ics.tut.ac.jp/~hanif/res/anchor_flood.zip

1.5 Link to the set of provided alignments (in align format)

The results for OAEI-2008 are available at our website: http://www.kde.ics.tut.ac.jp/~hanif/res/aflood.zip

2 Results

In this section, we describe the results of Anchor-Flood algorithm against the benchmark and anatomy ontologies provided by the OAEI 2008 campaign.

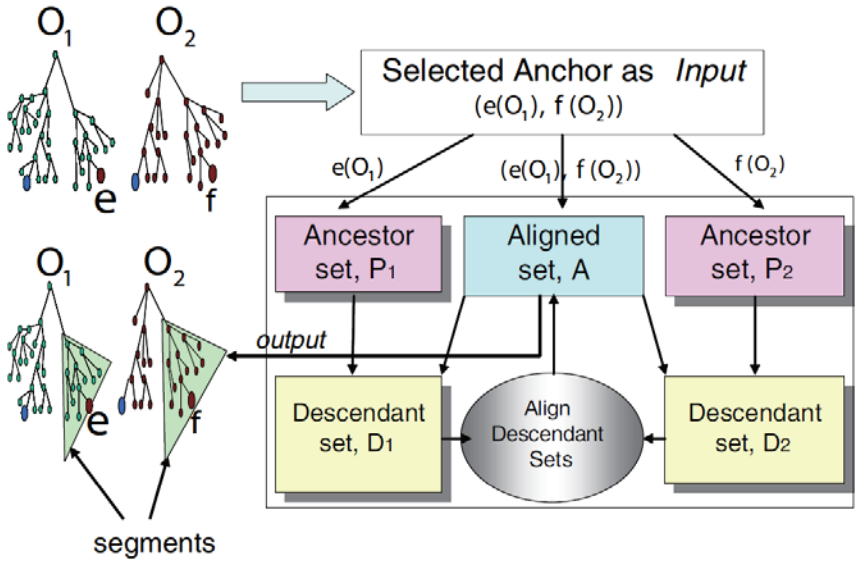

Fig. 1. The figure shows the process of Anchor-Flood algorithm where anchor is taken as an input to produce a segmented alignment. Multiple anchors produce a fair collection of aligned pairs as a whole.

2.1 Benchmark

On the basis of the nature, we can divide the benchmark dataset into five groups: #101-104, #201-210, #221-247, #248-266 and #301-304. We described the performance of our Anchor-Flood algorithm over each of the groups and depicted in . The overall summary over 1xx, 2xx and 3xx are also figured in .

#101-104 Table 1 shows that Anchor-Flood algorithm produces perfect precision and recall in this group.

#201-210 Although the lexical information of the ontologies are suppressed or modified, their structures remain quite similar. Therefore traversing the structure with taxonomy and relation works better for this group.

#221-247 The structures of the candidate ontologies are altered. However, the dynamic block collector of our Anchor-Flood algorithm can collect concepts and properties as the ontologies are small in size. Therefore, it can still produce good precision and recall.

#248-266 This is the most difficult group for our Anchor-Flood algorithm, as the structure and the lexical information altered significantly. However, the subgroups with xxx-2 through xxx-8 are seemingly easier to align.

Table 1. Summary of the average precision, recall and total elapsed time.

#	Prec.	Rec.	F-Measure	Total Time (sec)
1xx	1.00	1.00	1.00	2.67
2xx	0.93	0.69	0.79	87.08
3xx	0.88	0.79	0.83	3.43
Average	0.93	0.70	0.77	

#301-304 Anchor-Flood algorithm in this group works well even after removing the subsumption module from our main algorithm. Both structural and lexical analysis works well in this group.

2.2 Anatomy

In this test, the real world cases of anatomy for Adult Mouse Anatomy (2744 classes) and NCI Thesaurus (3304 classes) for human anatomy are included. These are relatively large compared to benchmark ontologies. The actual effectivity of our Anchor-Flood algorithm shows with faster operational time. It collects 1187 aligned pairs within only 1.09 minutes in our Core2 Duo 2.4MHz processor with 2GB of memory. Table 2 shows the summary of the performance on the anatomy task.

Table 2. The Anchor-Flood algorithm collects aligned pairs from anatomy ontologies quickly. The Table shows the brief summary of the output.

Total Aligned Pair	Required Time (m)
1187	1.09

3 General comments

In this section, we want to introduce comments on the results of Anchor-Flood algorithm and the way to improve the proposed system

3.1 Comments on the results

The main strength of our Anchor-Flood algorithm is the way of minimizing the comparisons between entities, which leads enhancement in performance. It has some better scope in the field of ontology versioning of small specific domain ontologies comparing with other large ontologies.

The weak points are: it has still rooms of improving alignments based on axioms, semantic similarity, and structures of ontologies.

3.2 Discussions on the way to improve the proposed system

The subsumption module of our algorithm takes much time. Our next plan is to improve the alignments on the basis of axioms and structures and improving the subsumption module as well.

4 Conclusion

Ontology matching is very important part of establishing interoperability among semantic application as the core of every semantic application is ontology. We implemented faster algorithm to align specific interrelated parts across ontologies, which gives the flavor of segmentation. Pair of segmented aligned part across ontology can be used versioning ontologies, e.g. cancer ontology versioning with the general diseases ontology. The anatomical ontology matching shows the effectiveness of our Anchor-Flood algorithm. Moreover, the experimental experience in the OAEI 2008 campaign will influence us building comprehensive ontology matching system removing the limitations of our algorithm in the future.

Acknowledgements

This study was partially supported by Global COE Program gFrontiers of Intelligent Sensingh from Japan's Ministry of Education, Culture, Sports, Science and Technology.

References

1. P. Bouquet, L. Serafini, S. Zanobini, Semantic Coordination: A New Approach and an Application, Proceedings of the 2nd International Semantic Web Conference (ISWC2003), Sanibel Island, Florida, USA (2003) 130–145.
2. J. Euzenat, P. Valtchev, Similarity-based Ontology Alignment in OWL-Lite, Proceedings of the 16th European Conference on Artificial Intelligence (ECAI2004), Valencia, Spain (2004) 333–337.
3. F. Giunchiglia, P. Shaiko, Semantic Matching, The Knowledge Engineering Review 18 (03) (2004) 265–280.
4. F. Giunchiglia, P. Shvaiko, M. Yatskevich, S-Match: an Algorithm and an Implementation of Semantic Matching, Proceedings of the 1st European Semantic Web Symposium (ESWS2004), Heraklion, Greece (2004) 61–75.
5. S. Melnik, H. Garcia-Molina, E. Rahm, Similarity Flooding: A Versatile Graph Matching Algorithm and its Application to Schema Matching, Proceedings of the 18th International Conference on Data Engineering (ICDE 2002), San Jose, CA (2002) 117–128.
6. J. Seidenberg, A. Rector, Web Ontology Segmentation: Analysis, Classification and Use, Proceedings of the 15th International Conference on World Wide Web (WWW2006), Edinburgh, Scotland (2006) 13–22.
7. G. Stoilos, G. Stamou, S. Kollias, A String Metric for Ontology Alignment, Proceedings of the 4th International Semantic Web Conference (ISWC2005), Galway, Ireland (2005) 623–637.
8. W. E. Winkler, The State of Record Linkage and Current Research Problems, Technical report, Statistical Research Division, U.S. Census Bureau, Washington 1999.

Appendix: Raw Results

The tests are carried out on an Intel Core 2 Duo 2.4MHz desktop machine with 2GB DDR2 memory under Windows XP Professional operating system and Java 1.6.0_02 compiler

Matrix of Results

The following table contains the results of Anchor-Flood algorithm in the benchmark test. The table includes precision (Prec.), recall (Rec.) and processing time. The processing time includes construction of model, execution of algorithm to produce aligned pairs and writing the results into a .rdf file.

#	Prec.	Rec.	F-Measure	Time (sec)
101	1.00	1.00	1.00	0.94
101	1.00	1.00	1.00	0.94
103	1.00	1.00	1.00	0.89
104	1.00	1.00	1.00	0.84
201	0.96	0.84	0.90	0.92
201-2	1.00	0.86	0.92	0.91
201-4	1.00	0.79	0.88	0.89
201-6	0.98	0.91	0.94	0.89
201-8	0.95	0.58	0.72	0.91
202	1.00	0.78	0.88	0.86
202-2	1.00	0.86	0.92	0.86
202-4	1.00	0.92	0.96	0.84
202-6	1.00	0.88	0.94	0.86
202-8	0.91	0.42	0.57	0.84
203	1.00	1.00	1.00	0.86
204	1.00	0.97	0.98	0.91
205	0.89	0.57	0.69	0.94
206	0.96	0.84	0.90	0.92
207	0.96	0.84	0.90	0.95
208	1.00	0.96	0.98	0.89
209	0.91	0.49	0.64	0.92
210	0.96	0.80	0.87	0.86
221	1.00	1.00	1.00	0.91
222	1.00	1.00	1.00	0.88
223	1.00	1.00	1.00	0.95
224	1.00	1.00	1.00	0.81
225	1.00	1.00	1.00	0.88
228	1.00	1.00	1.00	0.72
230	0.94	1.00	0.97	0.88
231	1.00	1.00	1.00	0.88

232	1.00	1.00	1.00	0.83
233	1.00	1.00	1.00	0.74
236	1.00	1.00	1.00	0.74
237	1.00	1.00	1.00	0.81
238	1.00	1.00	1.00	0.91
239	0.97	1.00	0.98	0.70
240	0.97	1.00	0.98	0.8 0
241	1.00	1.00	1.00	0.69
246	0.97	1.00	0.98	0.74
247	0.97	1.00	0.98	0.84
248	0.50	0.12	0.19	0.84
248-2	0.99	0.78	0.87	0.81
248-4	0.97	0.64	0.77	0.84
248-6	0.94	0.51	0.66	0.86
248-8	0.74	0.32	0.45	0.92
249	0.92	0.24	0.38	0.86
249-2	1.00	0.86	0.92	0.88
249-4	1.00	0.90	0.95	0.84
249-6	0.93	0.58	0.71	0.88
249-8	0.95	0.77	0.85	0.84
250	1.00	0.33	0.50	0.91
250-2	1.00	0.85	0.92	0.91
250-4	1.00	0.73	0.84	0.84
250-6	1.00	0.64	0.78	0.88
250-8	1.00	0.48	0.65	0.84
251	1.00	0.32	0.48	0.91
251-2	1.00	0.85	0.92	0.91
251-4	1.00	0.76	0.86	0.92
251-6	0.97	0.62	0.76	0.91
251-8	0.92	0.47	0.62	0.92
252	0.80	0.08	0.15	0.92
252-2	0.97	0.79	0.87	0.97
252-4	0.97	0.79	0.87	0.92
252-6	0.98	0.80	0.88	0.95
252-8	0.98	0.80	0.88	0.92
253	0.46	0.11	0.18	0.86
253-2	0.99	0.78	0.87	0.86
253-4	0.99	0.68	0.81	0.88
253-6	0.93	0.52	0.67	0.84
253-8	0.79	0.34	0.48	0.86
254	1.00	0.27	0.43	0.75
254-2	1.00	0.82	0.90	0.75
254-4	1.00	0.70	0.82	0.73

254-6	1.00	0.61	0.76	0.77
254-8	1.00	0.42	0.59	0.73
257	0.50	0.06	0.11	0.73
257-2	1.00	0.82	0.90	0.70
257-4	0.95	0.64	0.76	0.72
257-6	0.88	0.45	0.60	0.72
257-8	0.82	0.27	0.41	0.72
258	1.00	0.31	0.47	0.84
258-2	1.00	0.86	0.92	0.89
258-4	1.00	0.76	0.86	0.88
258-6	0.97	0.62	0.76	0.86
258-8	0.91	0.46	0.61	0.84
259	0.11	0.01	0.02	0.84
259-2	0.98	0.80	0.88	0.91
259-4	0.98	0.80	0.88	0.92
259-6	0.98	0.80	0.88	0.89
259-8	0.97	0.79	0.87	0.91
260	0.92	0.38	0.54	0.72
260-2	0.96	0.86	0.91	0.75
260-4	0.96	0.76	0.85	0.74
260-6	0.95	0.66	0.78	0.75
260-8	0.94	0.55	0.69	0.77
261	0.82	0.27	0.41	0.77
261-2	0.96	0.82	0.88	0.77
261-4	0.97	0.85	0.91	0.75
261-6	0.97	0.85	0.91	0.77
261-8	0.96	0.82	0.88	0.83
262-2	1.00	0.79	0.88	0.74
262-4	1.00	0.61	0.76	0.72
262-6	1.00	0.42	0.59	0.73
262-8	1.00	0.21	0.35	0.78
265	0.50	0.07	0.12	0.70
266	0.25	0.03	0.05	0.72
301	0.98	0.82	0.89	0.81
302	0.83	0.60	0.70	0.80
303	0.75	0.79	0.77	0.94
304	0.95	0.95	0.95	0.88
Total	0.93	0.70	0.80	91.45