Hertuda Results for OEAI 2012

Sven Hertling

Technische Universität Darmstadt
hertling@ke.tu-darmstadt.de

Abstract. Hertuda is a very simple element based matcher. It shows that tokenization and a string measure can also yield in good results. It is an improved version of the first version submitted to the OAEI 2011.5.

1 Presentation of the system

1.1 State, purpose, general statement

Hertuda is a first idea of an element based matcher with a string comparison. It generates only homogeneous matchings, that are compatible with OWL Lite/DL. This means that classes, data properties and object properties are handled separately. As a result there are three thresholds that can be set independently. One for class to class, object to object property and data to data property. A simple overall threshold sets all sub-thresholds to the same value.

Over all concepts a cross product is computed. If the confidence of a comparison is higher than the threshold for this type of matching, then it is added to the resulting alignment. For each concept all labels, comments and URI fragments are extracted. Then these terms form a set. To compare two concepts, respectively sets of terms, each element of the first set is compared with each element of the second set. The best value is the similarity measure for these concepts.

A preprocessing step for term comparison is to tokenize it. All camel case terms or terms with underscores or hyphens in it, are split into single tokens and converted to lower case. Therefore *writePaper*, *write-paper* and *write_paper* will all result in two tokens, namely $\{write\}$ and $\{paper\}$.

Afterwards a similarity matrix is computed with the Damerau–Levenshtein distance [1, 2]. The average of the best mappings are then returned as the similarity between two token sets. Figure 1 depicts schematically the algorithm of Hertuda.

1.2 Specific techniques used

The final matching system contains of the string matching approach and a filter for removing alignments that are not considered in the reference alignment. The system is depicted in figure 2.

The filter removes all alignments that are true, but are not in the reference alignment. The removed mappings are mostly from upper level ontologies like *dublin core* or *friend of a friend*.

```
void function hertuda() {
  for each type in {class, data property, object property}
    for each concept cOne in ontology one
      for each concept cTwo in ontology two
        if(compareConcepts(cOne, cTwo) > threshould(type)){
          add alignment between cOne and cTwo
        }
}

float compareConcepts(Concept cOne, Concept cTwo) {
  for each termOne in {label(cOne), comment(cOne), fragment(cOne)}
   for each termTwo in {label(cTwo), comment(cTwo), fragment(cTwo)}
     conceptsMatrix[termOne, termTwo] = compareTerms(termOne, termTwo)

  return maximumOf(conceptsMatrix)
}

float compareTerms(String tOne, String tTwo) {
  tokensOne = tokenize(tOne)
  tokensTwo = tokenize(tTwo)

  tokensOne = removeStopwords(tokensOne)
  tokensTwo = removeStopwords(tokensTwo)

  for each x in tokensOne
   for each y in tokensTwo
    similarityMatrix[x, y] = damerauLevenshtein(x, y)

  return bestAverageScore(similarityMatrix)
}
```

Fig. 1. Algorithm for Hertuda

1.3 Adaptations made for the evaluation

There are no specific adaptions made. The overall threshold for a normalised Damerau–Levenshtein distance is set to 0.88.

1.4 Link to the system and parameters file

The tool version submitted to OAEI 2012 can be downloaded from `http://www.ke.tu-darmstadt.de/resources/ontology-matching/hertuda`

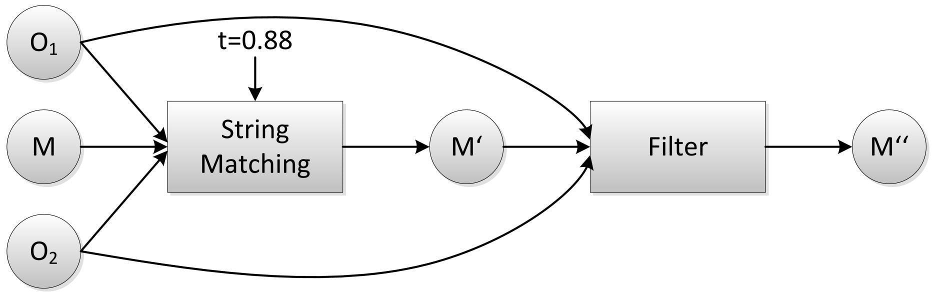

Fig. 2. Composition of matching algorithms of Hertuda. The string based approach and the filter are sequential composed.

2 Results

2.1 Benchmark

The implemented approach is only string based and works on the element level, whereby missing labels or comments or replaced terms by random strings has a high effect on the matching algorithm.

2.2 Anatomy

Hertuda has a higher recall than the *StringEquiv* from OEAI 2011.5 (0.673 to 0.622). Through the tokenization and also the string distance the precision is much lower (0.69 to 0.997). This yield in worse F-Measure for Hertuda (68.1 to 0.766).

2.3 Conference

The first version of Hertuda only compares the tokens for equality, whereas the new version computes a string similarity. Though the recall is a little bit higher than the first version, but the precision is lower. All in all, the F-Measure has increased by 0.01. This approach can find a mapping between *has_the_first_name* and *hasFirstName* with an similarity of 1.0.

2.4 Multifarm

Hertuda is not designed for multiligual matching. Nevertheless, some simple alignments are returned like $person(en) \equiv person(de)$.

2.5 Library

In the Library Track a relatively high recall has been achieved (0.925). Through splitting the words a very low precision value (0.465) was the result.

2.6 Large Biomedical Ontologies

Hertuda was only capable to match the small task for FMA-NCI and FMA-SNOMED. The large ones are not finished in time. The reason can be, that the complexity is to high trough the cross product of all concepts.

3 General comments

3.1 Comments on the results

The approach shows, that also simple string based algorithms can yield in good results. The improvement of version 1 is not much, but the recall was higher in many tracks. The precision was therefore lower, but it ends often in better F-Measure values.

3.2 Discussions on the way to improve the proposed system

To improve Hertuda it is possible to add more stop words in different languages. This helps by comparing two ontologies that have the same language, but this differs from English.

Another point is to set the threshold more precise and not one for all. It is also imaginable to set the threshold based on the matching ontologies. This will help to reduce the low precision in some tracks.

4 Conclusion

The results show that an string based algorithm can also produce good alignments. The recall of this version is in many cases much higher that the first version. Thus it is possible to use this matcher as a previous step of structural matchers.

References

1. Damerau, F.: A technique for computer detection and correction of spelling errors. Communications of the ACM **7**(3) (1964) 171–176
2. Levenshtein, V.: Binary codes capable of correcting deletions, insertions and reversals. In: Soviet Physics Doklady. Volume 10. (1966) 707