FTRLIM Results for OAEI 2019 *

Xiaowen Wang, Yizhi Jiang, Qin Liu, Hongming Zhu**, Hongfei Fan, Yi Luo, and Hua Jiang
School of Software Engineering, Tongji University, Shanghai, China
{1931533,1931566,zhu_hongming,qin.liu, fanhongfei,1731530,1653545}@tongji.edu.cn

Abstract. FTRLIM is a new instance matching system, which adopts the Follow the Regularized Leader(FTRL) algorithm to determine whether two instances refer to the same entity in the real world. We first introduce the major components of FTRLIM system briefly in this paper. Then we present the results of FTRLIM's participation in OAEI 2019 and discuss the strength and weakness according to the results.

1 Presentation of the system

1.1 State, purpose, general statement

Researchers have worked a lot on ontology alignment. Early methods focused on matching ontologies based on the schema. Recently, however, the instance-based matching has gradually become a hot topic.[1] There are many ontology matching systems that support to solve the instance matching problem, such as LogMap[2], AML[3], Lily[4], RiMOM-IM[5] and so on. But with the rapid growth of data scale, it has become a practical requirement to complete the task of instance matching among large-scale knowledge graphs.

FTRLIM system is designed to provide a effective and efficient solution of matching instances among large-scale datasets, whose core functionalities are listed as follows:

1. Build indexes for instance based on labels and textual attributes. Only instances with the same index have possibility to be aligned.
2. Calculate similarity between two instances on certain attributes and relationships. Different methods are used to calculate similarity according to the data type of attributes or relationships.
3. Generate train dataset for FTRL model [6] from the given data automatically. Specific instance pairs are selected as train set during the matching process without any user operations.

* This research has been supported by the National Key R&D Program of China (No. 2018YFB0505000), the Science and Technology Commission of Shanghai Municipality (No. 17511107303, No. 17511110202), the National Natural Science Foundation of China (No. 61702374), the Shanghai Sailing Program (No. 17YF1420500) and the Fundamental Research Funds for the Central Universities.

** Correspondence author, email: zhu_hongming@tongji.edu.cn
4. Aggregate similarities of different attributes and relationships into similarity score with FTRL model, which is trained after the generation of train set.
5. Select aligned instances according to similarity scores between each instance pairs.
6. Customize all procedures using configuration files.

FTRLIM is a newly developed system and it’s the first time that we have participated in OAEI evaluation. We expect to check the feasibility and effectiveness of our system, so that we rebuilt our system using Java with core functionalities. The complete version of FTRLIM system is developed and deployed on Spark cluster, which provides the system with ability to deal with gigabyte-level data. User feedback mechanism is integrated into the system as well. The system will correct matching result on the basis of feedback. Last but not least, merging aligned instances' attributes and relationships is also supported.

1.2 Specific techniques used

FTRLIM system consists of five major components: Index Generator, Comparator, Train set Generator, Model Trainer and Matcher. The system accepts input instances in OWL format, which are stored in source dataset and target dataset respectively. FTRLIM will find aligned instances between the two datasets. The structure of our system is presented in Fig.1.

![Fig. 1. FTRLIM System OAEI 2019](image)

**Index Generator** This component plays an important role in FTRLIM system. It builds indexes for all input instances based on their attributes. The system first extracts values of a specified instance attribute, then regards each of the values as a document, all of which will constitute a document set. The measurement TF-IDF is used to find keywords for each document. Finally the indexes of a instance
are generated from the combination of its keywords. FTRLIM system supports
users to generate indexes for instances via more than one attribute. In this
scenario, different indexes of a instance created referring to different attributes
will be concatenated together as the final index. Instances with the same indexes
are divide into the same instance block, and instances from different sources
under the same block will form candidate instance pairs. Only when a pair of
instance is candidate pair can they be aligned in the following procedures. When
there are only two instances from different data sources in the same block, these
two instances will form a unique instance pair, which will be regarded as aligned
instance pair directly. Missing value of attributes is taken into consideration to
avoid losing candidate instances as far as possible.

**Comparator** All candidate pairs will be sent to comparator to calculate similarity. The comparator compares two instances from different aspects. Edit distance
similarity is calculated for textual instance attributes, while Jaccard similarity
is calculated for instance relationships. The calculation result will be arranged
in order to form the similarity vector. For example, if we compare a candidate
pair \((x_1, x_2)\) under two attributes \((a_1, a_2)\) and relationship \(r_1\), the similarities of
\((x_1, x_2)\) from each aspect are 0.3, 1 and 0.8, respectively, the similarity vector
should be \((0.3, 1, 0.8)\). All the pairs are compared from identical aspects to ensure
that the same dimension of different similarity vectors has the same meaning.

**Train set Generator** This component will generate train set for FTRL model.
The input of FTRL model is the similarity vector, and the output is the similarity
score of the instance pair. All unique pairs are assigned with similarity score 1.0.
Unaligned instance pairs are built by randomly replacing one instance of each
unique pair. These pairs are assigned with similarity score 0.0. This component
is different from the complete version of FTRLIM, which will be introduced in
Section 1.3.

**Model Trainer** FTRL model is trained in this component with hyperparameters in configuration files. Benefit from FTRL model’s feature, the training pro-
cess won’t cost a long time. The trainer plays a greater role in the complete
version as well: it can be used to accept the feedback of user and adjust the
parameters of FTRL model. Users are allowed to choose a batch of candidate
instance pairs and correct the similarity score, or pick up a certain pair to correct.

**Matcher** All candidate pairs will obtain their final similarity score in this com-
ponent. The trained FTRL model accepts all the similarity vectors and predict
the matching scores of them. Instance pairs with score larger than 0.5 will be
regarded as aligned pairs. They will form the final output of aligned instances
together with unique pairs.
Configurations FTRLIM system is easily to be tailored according to user’s requirements. We expect that all matching procedures are under user’s control, so we allow users to customize their own FTRLIM system using configuration files. Users are able to set the attributes for index generation, the attributes and relationships for comparison, the hyperparameters for FTRL model and many other detailed parameters to get a better result.

1.3 Adaptions made for the evaluation

To participate in the evaluation, we rebuilt the FTRLIM system and replaced some manual operations with automatic strategy. In the origin version, FTRLIM system will compute the mean value of similarity vectors’ elements as the raw score for each instance pairs. Then it will select a batch of instance pairs that have raw scores higher than a threshold as positive samples, as well as the same amount of instance pairs whose raw scores are lower than the threshold as negative samples. Users will determine the similarity score by themselves to generate the train set. In the version developed for OAEI, this procedure is changed as we mentioned in 1.2. We excluded the non-core functionalities of the system, and made the ways of input and output suitable for the evaluation.

1.4 Link to the system and parameters file

The implementation of FTRLIM and relevant System adapter for HOBBIT platform can be found at this [FTRLIM-HOBBIT's gitlab page](https://gitlab.com/team-ftrlim/ftrlim-hobbit).

2 Result

In this section, we present the results obtained by FTRLIM in the OAEI 2019 competition. FTRLIM system participated in the SPIMBENCH track, which aims at determining when two OWL instances describe the same Creative Work. The datasets are generated and transformed using SPIMBENCH. The result is published in this [OAEI 2019 result page](https://www.aifb.kit.edu/oaei2019/).

2.1 SPIMBENCH

The SPIMBENCH task is composed of two datasets, the sandbox and the mainbox, with different scales. The sandbox has about 380 instances and 10000 triplets, while the mainbox has about 1800 Create Works and 50000 triplets.

Evaluation results of sandbox are summarised in Table 1. The optimal result is presented in bold text. Compared with AML, Lily and LogMap, FTRLIM system obtained the highest F-measure, highest recall and best time performance, while the precision is 0.08 lower than LogMap that has the best precision.

Evaluation results of mainbox are presented in Table 2 with the optimal result bolded. Our system is about 36% faster than Lily and about 17 times faster than
Table 1. The result of SANDBOX

<table>
<thead>
<tr>
<th></th>
<th>FTRL-IM</th>
<th>AML</th>
<th>Lily</th>
<th>LogMap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fmeasure</td>
<td><strong>0.9214175655</strong></td>
<td>0.864516129</td>
<td>0.9185867896</td>
<td>0.8413284133</td>
</tr>
<tr>
<td>Precision</td>
<td>0.8542857143</td>
<td>0.8348909657</td>
<td>0.8494318182</td>
<td><strong>0.9382716049</strong></td>
</tr>
<tr>
<td>Recall</td>
<td>0.8963210702</td>
<td>0.8963210702</td>
<td>1</td>
<td>0.762541806</td>
</tr>
<tr>
<td>Time performance</td>
<td><strong>1474</strong></td>
<td>6223</td>
<td>2032</td>
<td>6919</td>
</tr>
</tbody>
</table>

the slowest one, while the fmeasure is only 0.0015 lower than the best one. We obtained the nearly full mark on recall and the second highest precision as well.

Table 2. The result of MAINBOX

<table>
<thead>
<tr>
<th></th>
<th>FTRL-IM</th>
<th>AML</th>
<th>Lily</th>
<th>LogMap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fmeasure</td>
<td>0.9214787657</td>
<td>0.8604576217</td>
<td><strong>0.9216224459</strong></td>
<td>0.790560472</td>
</tr>
<tr>
<td>Precision</td>
<td>0.85584563</td>
<td>0.8385678392</td>
<td><strong>0.854638009</strong></td>
<td>0.8925895087</td>
</tr>
<tr>
<td>Recall</td>
<td>0.9980145599</td>
<td>0.8835208471</td>
<td>1</td>
<td>0.7094639312</td>
</tr>
<tr>
<td>Time performance</td>
<td><strong>2155</strong></td>
<td>39515</td>
<td>3667</td>
<td>26920</td>
</tr>
</tbody>
</table>

3 General comments

3.1 Comments on the result

FTRLIM system has satisfactory performance in both datasets of SPIMBENCH, especially in the sandbox. Our system benefits a lot from the significant component, Index Generator. It helps the system filter out instance pairs with a high possibility to be aligned effectively and efficiently. The comparator only needs to compare instances with same indexes rather than every instance pairs. The datasets of SPIMBENCH contains a wealth of textual information, and there are many attributes that can be used to build indexes or to compare the similarity among instances. The FTRL model trained by the Model Trainer component is as smart as we expect to learn a weight for attributes or relationships and distinguish pairs of instances pointing to the same entity in real world.

The major weakness compared with the other systems is the precision. We found that the system reports some false positives when using the sandbox dataset and its reference alignments. There are a few instance pairs with a slight difference regarded as the aligned pairs by our system while not by SPIMBENCH. Our system still does not have enough capacity to deal with this situation.
4 Conclusion

In this paper, we briefly present our instance matching system FTRLIM. The core functionalities and components of the system are introduced, and the evaluation results of FTRLIM system are displayed and analyzed. FTRLIM system got the significantly better time performance than other systems in both datasets of SPIMBENCH, and got the highest fmeasure in sandbox and almost the same fmeasure as the best one in mainbox. The results proved the effectiveness and highly efficiency of our matching strategy, which is important for matching instances among large-scale datasets.

There are still many aspects to be improved in FTRLIM system. We'll continue to optimize the algorithm of generating indexes for instances and the matching strategy in following work. More comparison methods and supporting data types should be attached to our system as well. And we are committed to building the GUI for our system. Although FTRLIM system is specially designed to solve the instance matching problem, it’s also expected to produce meaningful results in other similar tracks in the future.

References