FTRLIM Results for OAEI 2020 ⋆ ⋆⋆

Xiaowen Wang[1], Yizhi Jiang[1], Hongfei Fan[1],
Hongming Zhu[1]⋆⋆⋆, and Qin Liu[1]

School of Software Engineering, Tongji University, Shanghai, China
{1931533,1931566,fanhongfei, zhu_hongming,qin.liu}@tongji.edu.cn

Abstract. FTRLIM is a distributed framework that is designed for large-scale instance matching. The FTRLIM framework leverages the blocking algorithm to generate candidate instance pairs, and applies the follow-the-regularized-leader model to determine whether candidate instance pairs are matched. FTRLIM participated in the SPIMBENCH Track of OAEI 2020, and achieved the fastest matching efficiency both in SANDBOX and MAINBOX, as well as the competitive matching quality.

1 Presentation of the system

1.1 State, purpose, general statement

The instance-based matching has gradually become a promising topic recently[1]. Many methods have been proposed to complete the instance matching task. Several state-of-the-art instance matching methods evolve from ontology matching methods such as LogMap[2], AML[3], RiMOM-IM[4], and Lily[5]. As the scale of the data increases, the efficiency and cost requirements of instance matching methods become more stringent.

FTRLIM is a distributed instance matching framework that focus more on the matching efficiency. When matching instances, it first generates indexes for instances based on their attributes. Instances with the same index are divided into the same instance block, and instances from different sources under the same block form the candidate instance pairs. Then FTRLIM figures out the matched instance pairs leveraging the online-learning model, follow-the-regularized-leader (FTRL). This is the second time that FTRLIM has participated in the OAEI evaluation. To participate in the SPIMBENCH Track, FTRLIM is rebuilt using JAVA with core functionalities as the submitted version. The complete version of FTRLIM has been developed and deployed on a Spark cluster, which provides the FTRLIM framework with ability to deal with large-scale data. Compared

⋆ Copyright © 2020 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

⋆⋆ This research has been supported by the National Key R&D Program of China (No. 2018YFB0505000), the National Natural Science Foundation of China (No. 61702374), and the Fundamental Research Funds for the Central Universities.

⋆⋆⋆ Corresponding author, email: zhu_hongming@tongji.edu.cn

with last year's version, this year's FTRLIM has been slightly changed, which will be introduced later.

1.2 Specific techniques used

This section introduces the refined working flow of FTRLIM. FTRLIM consists of four major components: *Blocker*, *Comparator*, *Trainer*, and *Matcher*. The framework accepts input instances in the OWL format, which are stored in source dataset and target dataset, respectively. FTRLIM finds matched instances between the two datasets. The overview of the FTRLIM's work flow is presented in Fig. 1.

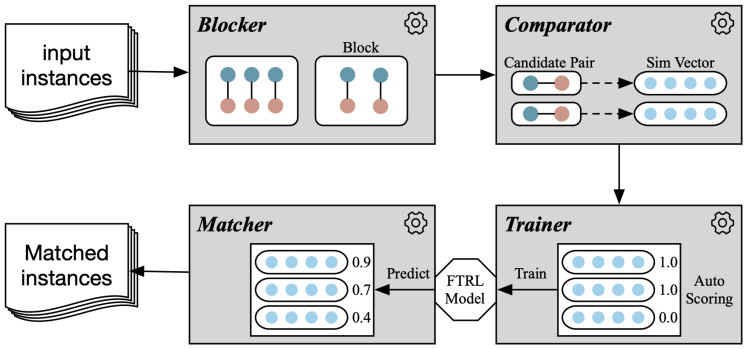

Fig. 1. Work Flow of FTRLIM OAEI 2020

Blocker Since the scale of instances that need to be matched is usually very large, it is very time-consuming and space-consuming to compare all the instances with each other to find matched instance pairs. *Blocker* extracts features of textual attributes related to instances to generate indexes for them. The interactions among different textual information are taken into consideration, which allows instances to be fine-grained divided. It also has the ability to infer indexes for instances whose textural attributes are in-completed or missing. FTRLIM supports users to generate indexes for instances via more than one attribute. Instances with the same index are divided into the same instance block, and instances from different sources under the same block will form candidate instance pairs. Only when a pair of instances is a candidate pair can it be matched 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[4], which will be regarded as an matched instance pair directly.

Comparator All candidate pairs will be sent to the comparator to calculate similarity. The comparator compares two instances from user-specified aspects.

The edit distance similarity is calculated for textual instance attributes, while the Jaccard similarity is calculated for instance relationships. The calculation results will be arranged in order to form the similarity vector. Formally, let the list of predicates adopted by *Comparator* be $\langle p_1, p_2, \ldots, p_n \rangle$, then the similarity vector of the two instance is

$$\langle s_1, s_2, \ldots, s_n \rangle, s_i \in [0, 1], (i = 1, 2, \ldots, n)$$

where s_i is the similarity of the two instances under the i-th predicate.

Trainer FTRLIM treats the instance matching as a regression problem, where the similarity score between two instances can be regarded as the probability that the two instances are matched. We innovatively introduce the FTRL model[6] to solve the problem. FTRL is a widely-used online logistic regression model with high precision, excellent sparsity, fast training speed and satisfactory streaming data processing ability. *Trainer* is designed to train the FTRL model for instance matching. It first generates train set for the FTRL model. After the preparation of train set is completed, the FTRL model will be trained with hyperparameters in configuration files. Benefiting from the FTRL model's feature, the training process won't cost a long time. The *Trainer* component plays a greater role in the complete version. It can be used to accept the feedback of users and adjust the parameters of the 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 scores in this component. Since FTRLIM produces the similarity scores in the interval [0,1], candidate pairs whose scores are greater than 0.5 will be regarded as matched pairs. The matching score s is calculated as follows:

$$s = \frac{1}{1 + e^{-\boldsymbol{x}^T \boldsymbol{w}}} \tag{1}$$

where $\boldsymbol{x}$ is the similarity vector, $\boldsymbol{w}$ is the weight of the FTRL model. In this year's submission, all elements of similarity vectors accepted by the FTRL model are unified from [0, 1] to [-1, 1] to satisfy the symmetry of the equation.

Configurations FTRLIM is easily to be tailored according to user's requirements. We expect that all matching procedures are under user's control, thus 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 the 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 FTRLIM and replaced some manual operations with automatic strategies.

The train set for training the FTRL model is automatically generated in the submitted version, while it needs manual scoring in the completed version. The train set is composed of instance pairs' similarity vectors as well as their similarity scores. The *Trainer* regards all unique pairs as matched pairs. Therefore, it selects all similarity vectors of unique pairs as positive samples, and assigns them with similarity score 1.0. The mismatched pairs are built by replacing one instance of each unique pair randomly. These pairs are assigned with similarity score 0.0 and treated as negative samples in the train set. In the completed version, however, FTRLIM does not regard all unique pairs as matched pairs directly. It 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. Besides, we excluded the non-core functionalities of FTRLIM such as the user-feedback and the load balance mechanism. The ways of input and output is adapted for the evaluation as well.

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.[1]

2 Result

In this section, we present the results obtained by FTRLIM in the OAEI 2020 competition. FTRLIM participated in the SPIMBENCH Track, which aims at determining whether two OWL instances describe the same Creative Work. The datasets are generated and transformed using SPIMBENCH[7]. Our competitors includes LogMap[2], AML[3], Lily[5] and REMinder. The first three systems have participated in this track for many years, while REMinder is a new contestants in this year. The results are published in this OAEI 2020 result page[2].

2.1 SPIMBENCH

The SPIMBENCH task is executed in two datasets, the SANDBOX and the MAINBOX, of different size. The SANDBOX has about 380 instances and 10000 triplets, while the MAINBOX has about 1800 instances and 50000 triplets. We summarized the results of the SPIMBENCH Track in Table 1 and Table 2, where the best results are indicated in bold.

Table 1. The Results of SANDBOX

	LogMap	AML	Lily	FTRLIM	REMiner
Fmeasure	0.8413	0.8645	0.9917	0.9214	**0.9983**
Precision	0.9383	0.8349	0.9836	0.8542	**1**
Recall	0.7625	0.8963	**1**	**1**	0.9967
Time performance	7483	6446	2050	**1525**	7284

Table 2. The Results of MAINBOX

	LogMap	AML	Lily	FTRLIM	REMiner
Fmeasure	0.7856	0.8605	0.0.9954	0.9215	**0.9977**
Precision	0.8801	0.8385	0.9908	0.8558	**0.9987**
Recall	0.7095	0.8835	**1**	0.9980	0.9967
Time performance	26782	38772	3899	**2247**	33966

Compared with all competitors, FTRLIM achieves the best time performance on both two datasets. The time cost of our framework is reduced by 25.6% than the second fastest one, Lily, on SANDBOX, while it is reduced by 42.4% than Lily on MAINBOX. The results on time performance indicate the efficiency of FTRLIM, which is more essential for large-scale instance matching. The FTRLIM also achieves the highest recall on SANDBOX and almost the highest recall on MAINBOX. The precision of FTRLIM is relatively low on both dataset. There are two reasons that account for this situation. One reason is that the automatic strategy we adopted for generating train set is flawed. In the generated train set, there is almost no similarity between the the sample instance pairs with low score. Although this kind of samples helps the FTRL model learn to distinguish similar instance pairs from dissimilar instance pairs, it does not help the model distinguish matched instance pairs from similar instance pairs. Then the model prefers to predict high similarity scores for similar instance pairs, which improves the recall but reduces the precision. Another reason is that there may be problems with the way unique pairs are treated. Regarding the unique pairs as matched pairs directly will also affect the precision of the prediction. But the overall matching quality of FTRLIM is still competitive.

3 General comments

3.1 Comments on the result

FTRLIM has achieved time performance in both datasets of SPIMBENCH. The *Blocker* component makes a significant contribution to achieving the results. It helps the framework filter out instance pairs with a high possibility to be

[1] https://git.project-hobbit.eu/937522035/ftrlimhobbit
[2] http://oaei.ontologymatching.org/2020/results

matched effectively and efficiently. The *Comparator* component only needs to compare instances with the same indexes rather than every instance pairs. The datasets of SPIMBENCH contain 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 *Trainer* is able to learn a weight for attributes or relationships and distinguish instance pairs that points to the same entity in the real world. Compared with other systems, the precision of FTRLIM is unsatisfactory, which should be improved in feature works.

3.2 Improvements

There are still many aspects to be improved in FTRLIM. The submitted version of FTRLIM generates flawed train set for training the FTRL model, and considers unique pairs as matched instances unconditionally. The automatic strategy adopted by *Trainer* and *Matcher* should be optimized to address the problems. More comparison methods for various data types should be attached to our frameworks as well. Although FTRLIM is specially designed to solve the instance matching problem, it is also expected to produce meaningful results in other similar tracks in the future.

4 Conclusion

In this paper, we briefly presented our instance matching framework FTRLIM. The core functionalities and components of FTRLIM were introduced, and the evaluation results of FTRLIM were presented and analyzed. FTRLIM achieved significantly better time performance than other systems on both two datasets of SPIMBENCH, as well as the competitive matching quality. The results indicated the effectiveness and high efficiency of our matching strategy, which is important for matching instances on large-scale datasets.

References

1. Otero-Cerdeira, L., Rodríguez-Martínez, F.J., Gómez-Rodríguez, A.: Ontology matching: A literature review. Expert Systems with Applications **42**(2), 949–971 (2015)
2. Jiménez-Ruiz, E., Grau, B.C.: Logmap: Logic-based and scalable ontology matching. In: International Semantic Web Conference. pp. 273–288. Springer (2011)
3. Faria, D., Pesquita, C., Santos, E., Cruz, I.F., Couto, F.M.: Agreementmakerlight results for oaei 2013. In: OM. pp. 101–108 (2013)
4. Shao, C., Hu, L., Li, J.Z., Wang, Z., Chung, T.L., Xia, J.B.: Rimom-im: A novel iterative framework for instance matching. Journal of Computer Science and Technology **31**, 185–197 (2016)
5. Wu, J., Pan, Z., Zhang, C., Wang, P.: Lily results for oaei 2019. In: OM@ ISWC. pp. 153–159 (2019)

6. McMahan, H.B., Holt, G., Sculley, D., Young, M., Ebner, D., Grady, J., Nie, L., Phillips, T., Davydov, E., Golovin, D., et al.: Ad click prediction: a view from the trenches. In: Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 1222–1230 (2013)
7. Saveta, T., Daskalaki, E., Flouris, G., Fundulaki, I., Herschel, M., Ngomo, A.C.N.: Spimbench : A scalable , schema-aware instance matching benchmark for the semantic publishing domain (2014)