Scientific Journal of Impact Factor (SJIF): 4.14

e-ISSN (O): 2348-4470
p-ISSN (P): 2348-6406

International Journal of Advance Engineering and Research Development

Volume 3, Issue 4, April -2016

User Interface to evaluate Fuzzy Matching Results

Chirag Nagpal[1], Neha Malik[2], Ajit Singh[3], Shushma Shirke[4], Shinsmon Varghese[5]

[1-5]*Dept. Of Computer Engineering, Army Institute of Technology*

Abstract —*Entity Resolution is a challenging problem, even more relevant in todays' time of big data. In this paper, we present a User Interface framework in order to perform and evaluate results of the Entity Resolution process. The user interface provides an intuitive experience to an end user to experiment with various parameters, and machine learning algorithms to perform fuzzy clustering based Entity Resolution.*

Keywords*-component; formatting; style; styling; insert (key words) (minimum 5 keyword require) [10pt, Times new roman,Italic, line spacing 1.0]*

I. INTRODUCTION

Entity Resolution is a challenging problem, Entity Resolution is a challenging Computer Science problem. In today's world of Big Data, and the Semantic Web, entity resolution has become an even more important problem to solve. Today, the internet provides extensive amount of data. This data, might be in the form of structured formats like SQL Tables, or maybe completely unstructured with multiple hierarchical structures.

This unstructured data, is a modern feature of today's world of Big Data, and any computational task requiring the exploitation of this data requires it to be arranged, systematically, such that standard algorithms can be easily deployable across a wide range of data sources. The unstructured and noisy nature of this data, also necessitates newer more robust approaches to solve the Entity Resolution problem.

Traditional ER approaches employ extremely naive comparisons between individual features in order to solve the ER problem. Thus a traditional ER approach, might involve the comparison of a combination of certain specific features from two individual databases. In case of a match, these records are grouped together, or in other words resolved into a single entity.

A common extension to such an approach is the 'Merge-and Purge' strategy in which records are compared, pairwise, and if found to be matched, they are merged into a single record. Such a strategy is computationally more tractable as compared to naïve pairwise comparison. [1]

While such an approach might work well, for well structured datasets, however, determining such crisp rules, to resolve unstructured datasets is a challenging task. This challenge stems from the fact that it is extremely difficult to determine a single rule or combination of rules, that can effectively predict a match with perfect confidence, and such a rule can obviously not be determined manually, owning to the possibility of large dimensionality of the data. On the other hand, the dataset under consideration might be incomplete, that is, might be missing in certain features, and hence a specific rule or set of rules may not be applicable. This is especially true in case of data crawled from the internet, like for example microblogs.

A case in example could be microblogs. Let us consider an Entity Resolution problem involving the resolution of all tweets posted at from a particular location. While in some cases, microblogs may contain specific features like the geo-location of posting, in other cases it might contain this particular feature. Thus a simple rule involving the resolution of all tweets posted at a particular area may not be adequate. Here, the match function would require a more sophisticated approach of looking at other features, like if the post is from the same user and the time difference between the various posts is minimal, we can predict that the posting location is the same.

Such complex relations between various features can not be modelled easily with simple feature subsets and require more complicated approaches. A reasonable approach here would be to leverage machine learning approaches in order to learn such complicated features to predict a match. Incase we tried to model the problem above using a standard supervised learning model like decision tree, it would have been able to learn the complicated relations between various individual features in the features, thus able to come up with a representation for the problem. In the most obvious case a decision tree for this task would have looked at the location information, incase not present, it would then have tried to look for if the user is the same and then, the posting difference. In case this corresponded to the previously learnt values for these features, the algorithm would have been able to compute if there is a match.

In such a specialized, specific application, we require to model this problem in an unsupervised or semi supervised setting, in order to carry out the ER process, without the need of labelled training data. [2] In this paper, we present a framework to carry out the semi supervised pipeline to perform entity resolution. Our semi supervised pipeline depends

on a specific feature or combination of features in order to generate positive samples for training a supervised classifier. Our pipeline utilizes multithreaded programming along with Cython[3], which helps speed up the processing by compiling directly to machine code

Since we require a fuzzy, probabilistic output for predicting a match, our framework allows the user to experiment with various supervised classifiers like Logistic Regression and Random Forests, which, along with the binary prediction can also present a score, of confidence for the given prediction.

We also provide the UI tools to perform hyper parameter tuning for our classifier, in order to ensure that the resulting model has the correct required specifications in terms of recall, precision, TPR, FPR etc. We also provide a literature review on existing entity resolution approaches, and machine learning models.

II. UI TOOLS

Most datasets do not come with labelled training data. However, we can utilize certain features, to train a classifier as a match function.

2.1. Proxy Feature

The Proxy Feature is defined as the dataset dimension, to be used to generate labelled data for a match function. Thus, the framework first performs ER using matching using the strong proxy label, and then learns the match function.

2.2. Feature Extractor

It allows, us to define, and write a macro or a shell snippet to perform extraction of information from a corresponding input table. This allows us to have multiple features extracted from an underlying physical table

2.3. Classifier

This option allows us to learn the appropriate classifier given our proxy feature. It is basically a wrapper around Scikit Learn [4]. Choices include several popular ML algorithms like Decision Trees, Naïve Bayes, Logistical Regression. It also includes options to compare and evaluate the performance of a given classifier with respect to others on metrics like

- FPR: The false positive rate. The number of entries incorrectly identified.
- ROC Space: A plot between the True Positive Rate and The False Positive Rate.
- PR Space etc: A Plot between the Precision and Recall.
- TPR: The true positive rate of the classifier. This is the number of entries that are correctly identified.

2.4. Query Server

This is one of the most powerful features of our User Interface. The software also provides a SQL like query option, wherein a user, with little or no prior data science experience can also perform ER and query the server using simple SQL like syntax. The query server, converts the SQL command and runs the requisite code in the background. Thus model loading, and fitting is abstracted away from the user.

It also determines if the size of the data is enough to carry out the operation in memory or saving it to disk intermediately. It thus provides a clean, easy to use syntax without the need of the end user to write unclean code segments or patches. For example let us take the query below

```
resolve * from table1 where strong_feature=phone_number model=log_reg and model_par_a=0.3 and model_par_b=0.5
```

Here resolve keyword is analogous to the select statement in SQL. table1 identifies the name of the table on which ER is to be performed. `strong_feature` is to identify the column to use as strong feature and `model_par_x` defines various parameters require to fit the required model.

2.5. Plotting Server

This module allows the end user to seamlessly utilize certain external plotting tools like Matplotlib [5] and Gephi [6], inorder to plot the resulting output of the Entity Resolution process, as the output clusters can be visualized in terms of a Graph, with edges representing the matched entitities.

III. SOFTWARE DESIGN

We provide certain formal declaration to our software design as under.

A. Use Case

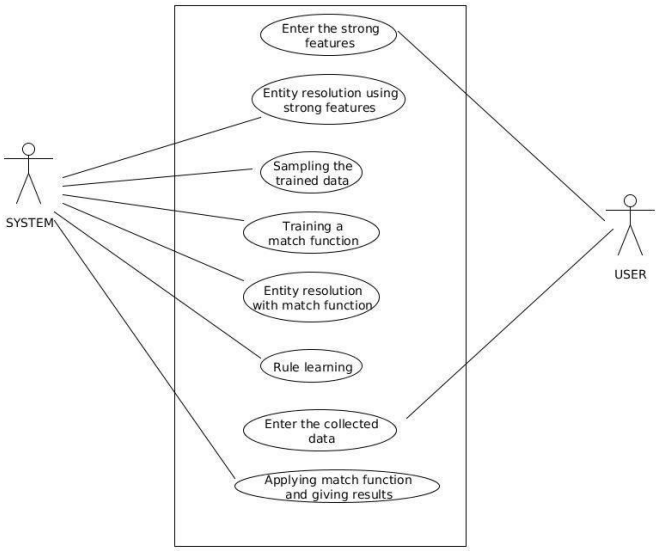

Fig. 1 Use Case Diagram for the System

The end user is the primary actor in this system. The most common actions to be performed are the data addition and management tasks. Other actions that involve the internal working of the systemin include Training the required ER model, and performing the Entity Resolution.

B. Activity Diagram

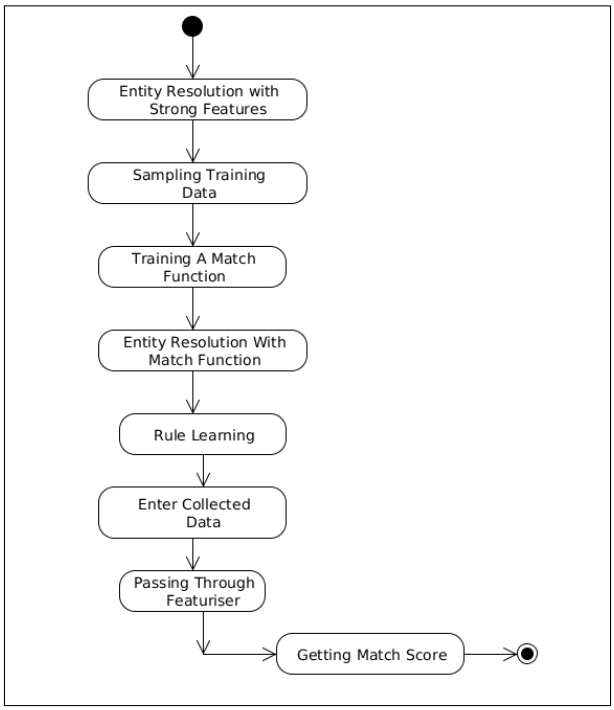

Fig. 2 Activity Diagram for the System

The activity diagram presents the most common approach for the user to interact with the system. The user first specifies the strong features to use, and ER is performed on the strong features using pairwise matching. Based on the clusters obtained we then generate adequate training data for a classifier to be trained. The appropriate classifier is selected, with the required parameters. The size of the test and training dataset is set based on the accuracy requirement, amount of computational resources.

The evaluation metric like Holdout number of Folds in Cross Validation can be selected. Results are obtained using the dynamic GUI, in terms of ROC plot and PR curves, which can be used to compare with the historic data that is available. Incase the end user is not satisfied with the classifier results, the classifier can be retrained, or some other classifier can be selected.

The system also provides the user the option to perform Grid Search over the number of parameters in order to find the appropriate value of the parameters, that reduce overfitting, while keeping a high accuracy on both the training as well as the test set. Based on the results obtained, an appropriate value of the match function is learnt which is then used to re perform Entity Resolution on the dataset.

The GUI also provides the ability to featurise the extracted connected components and perform rule learning on the extracted components, which helps identify characteristics that determine how the obtained clusters vary.

IV. CHALLENGES

Scaling the system is a tremendous challenge. Since we use pairwise comparisons, the system is effectively quadratic in time complexity. This nature can be significantly improved by using some intelligent notion of 'bucketing' that is dividing the data into partitions, and performing ER on the individual partitions and then merging the results. Such a strategy, if efficiently implemented can give near optimal results with a tremendous improve in time complexity.

Most steps in the pipeline of the current system are embarrassingly parallel. The current system, perform these operations using Multiple threads in parallel. Another good strategy to consider, is to carry out these steps over a distributed system using Map Reduce (Hadoop). The possibility of GPU computing also provides evidence for use of NVIDIA CUDA for these massively parallel operations.

Another significant challenge is the learning of a confidence threshold to perform ER. Low confidence thresholds lead to a breakdown while high confidence thresholds, lead to extremely granular resolved clusters. The creeping up of false positives, results in the breakdown at even very high thresholds. Further research needs to be carried out in order to ensure the prevention of such a breakdown.

REFERENCES

[1] Benjelloun, Omar, et al. "Swoosh: a generic approach to entity resolution."The VLDB Journal—The International Journal on Very Large Data Bases18.1 (2009): 255-276.

[2] Veeramachaneni, Sriharsha, and Ravi Kumar Kondadadi. "Surrogate learning: from feature independence to semi-supervised classification."Proceedings of the NAACL HLT 2009 Workshop on Semi-Supervised Learning for Natural Language Processing. Association for Computational Linguistics, 2009.

[3] Behnel, Stefan, et al. "Cython: The best of both worlds." Computing in Science & Engineering 13.2 (2011): 31-39.

[4] Pedregosa, Fabian, et al. "Scikit-learn: Machine learning in Python." The Journal of Machine Learning Research 12 (2011): 2825-2830.

[5] Hunter, John D. "Matplotlib: A 2D graphics environment." Computing in science and engineering 9.3 (2007): 90-95.

[6] Bastian, Mathieu, Sebastien Heymann, and Mathieu Jacomy. "Gephi: an open source software for exploring and manipulating networks." ICWSM 8 (2009): 361-362.

[7] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

