Lyndon Puzon

Dr. Xu Chu

Intro. to Database Systems

27 Apr. 2021

Entity Matching

Code

The code for this solution is at: https://github.com/donpuz/CS4400entitymatching

Solution Outline

Using *Google Colab*, we use *DeepMatcher* and *py_entitymatching* to implement, train, and tune a deep *Hybrid* model for entity matching as described in this paper.

1. Data reading, merging, training sets, pre-processing
    1.1. We first read the right table, left table, and the given labeled data
    1.2. We then merge the right table and the left table on the id's specified in the labeled data yielding a labeled DataFrame of the right table attributes and left table attributes. This format is enforced by *DeepMatcher* for all data.
    1.3. Next, we split the labeled data into a training set, validation set, and test set with the following split: 60%, 20%, and 20% for the training set, validation set, and a test set, respectively.
    1.4. We lowercase and tokenize all attributes, and use *fasttext* to create character-level embeddings, excluding the left and right table id's (as we do not want to learn on the product id's).
2. Model definition and training

2.1. We use the *Hybrid* architecture included in *DeepMatcher*, and tune the hyper-parameters *learning rate annealing factor*, *batch size*, *number of epochs*, and *positive to negative weight ratio* (used in NLLLoss to account for the imbalance in the labeled data). An *Adam* optimizer is selected for gradient descent.

2.2. At each epoch, the F1 score on the validation set is calculated, and the model at the epoch where the F1 score on the validation set is greatest is the one used to make our prediction on the unlabeled data. We utilize this model selection to reduce overfitting on the training set.

3. Blocking and pre-processing unlabeled data (candidate set)

3.1. We block on *brand* by performing single-token overlap blocking on *title* from the left table and *brand* from the right table to obtain our candidate set. For any record where *title* from the left table or *brand* from the right table is empty, we pair said record with every record in the other table. We choose these attributes because we observe that *brand* in the left table is dirty (ie. there are many missing attributes) and that this results in many additional candidate pairs, as the right table has 22074 records (ie. every dirty record in the left table results in 22074 additional candidate pairs). We overlap block on *title* and *brand* because it is likely that a product's brand is included in *title*. The unlabeled data (candidate set) is saved as *unlabeled.csv.*

3.2. We then perform the same merging and pre-processing on the unlabeled data as we did for the training set.

4. Predictions and output

4.1. We perform predictions on our unlabeled data using our trained model, using a threshold of 0.5 to select positive matches, as defined in the paper above.

4.2. The matched pairs which are included in the labeled data are excluded, and the remaining matched pairs are saved in *output.csv*.