The School of Mathematics

THE UNIVERSITY
of EDINBURGH

Project 2 - Gauging the Performance of Deep Learning for De-Duplication and Record Linkage in High and Low Resource Environments

by

Lennart Hoheisel, s1744523

Dissertation Presented for the Degree of
MSc in Statistics with Data Science

August 2021

Supervised by
Dr Daniel Paulin, Dr Grant Galloway, Dr Victor Elvira and Alaa Amri

Executive Summary

While data storage is continually becoming cheaper, computational cost increases with growing input datasets, and analytics suffer from unclean data. For this reason, it is important to maintain clean, coherent and deduplicated data sets. To achieve this, while moving locally stored data to cloud storage can be difficult, and various Machine Learning and Deep Learning models have been proposed to achieve record linkage - combining two different datasets and filtering out the overlap - and de-duplication, which is the process of finding duplicates in a dataset. This report aims to gauge the performance of a number of Machine Learning solutions and neural network based Deep Learning solutions on these two tasks, in low and high resource scenarios. Extensive analysis showed that both types of methods could cope well with the complexity, achieving F1 scores in the medium 0.9s on medium sized datasets, while achieving scores in the higher 0.9s on high resource datasets. Usually the more complex neural-network-based solutions could outperform the simpler models, but the increased computational cost might not make the relatively small gains very feasible. Furthermore, a complex Deep Learning model could achieve very high F1 scores in the low to mid 0.9s with training based on as little as 400 labelled samples. These results confirmed the findings of similar studies in the literature, and raises some new questions which are stated in the conclusion of this report.

Acknowledgments

I would like to thank Dr Daniel Paulin and Dr Grant Galloway for their assisstance in creating this report, and their help throughout my studies. Further, I would like to thank everyone responsible for the creation and guidance of this project at Amazon Research Scotland for making this project possible.

University of Edinburgh – Own Work Declaration

Name: Lennart Hoheisel
Matriculation Number: s1744523
Title of work: Gauging the Performance of Deep Learning for De-Duplication and Record Linkage in High, Mid and Low Resource Environments

I confirm that all this work is my own except where indicated, and that I have:

- Clearly referenced/listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc)
- Given the sources of all pictures, data etc. that are not my own
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Not sought or used the help of any external professional academic agencies for the work
- Acknowledged in appropriate places any help that I have received from others (e.g. fellow students, technicians, statisticians, external sources)
- Complied with any other plagiarism criteria specified in the Course handbook

I understand that any false claim for this work will be penalised in accordance with the University regulations (`https://teaching.maths.ed.ac.uk/main/msc-students/msc-programmes/statistics/data-science/assessment/academic-misconduct`).

Signature .....

Date: August 6th, 2021

Contents

1 Introduction **1**

2 Data and Methodology **1**
2.1 Description of Survey Data 1
2.2 Methodology 1
2.2.1 Deep Learning Concepts 2
2.2.2 Data Processing 2
2.2.3 Models 4

3 Results **6**
3.1 Structured Record Linkage with Google Scholar and DBLP data 6
3.2 Dirty De-Duplication with Cora data 7
3.3 Transfer Learning 9
3.4 Active and Transfer-Active Learning 9

4 Conclusion **11**

Appendices **13**

A Appendix A- Model Information **13**

List of Tables

1 Number of missing values in Cora data by feature 13
2 Deep Learning Model Specifications on DBLP-Scholar 13
3 Deep Learning Model Specifications on Cora-Cora 14
4 Novel Learning Algorithm in Low-Resource Scenarios - Model Specifications 14
5 More Novel Learning Algorithm in Low-Resource Scenarios - Model Specifications . . 15

List of Figures

1 Model Attributes Processing Framework 3
2 Results of ML models on DS task 6
3 Results of DL models on DS task 7
4 Precision-Recall of DL models on DS task 7
5 Results of ML models on CC task 8
6 Results of DL models on CC task 8
7 Precision-Recall of selected DL models on CC task 8
8 Performance of Hybrid model in Transfer Learning 9
9 Performance of Hybrid model in Active Learning Task 10
10 Performance of Hybrid model in Active Learning Task 10

1 Introduction

A 2009 news article by the Financial Times [13] identifies deduplication, the process of removing duplicates in a single dataset, and entity resolution, the process of removing overlap when consolidating multiple datasets, as one of the most effective ways for businesses to cut cost as they make the transition from storing data on tapes to hard disks. Today, as another transition is made to cloud-based data storage, the problem of entity-resolution remains just as important, not simply for cost efficiency in storage, but also to sustain clean, harmonized records that might then be used for analysis. This report aims to assess the predictive power of various Machine Learning (ML), Deep Learning (DL), and novel algorithmic solutions in simple and complex settings and in low and high-resource environments. Extensive tests performed on tasks with data availability between 400 and 250,000 labelled entity mention pairs showed that DL algorithms can outperform the ML models on most tasks, however the high computational cost does not seem to be set off by the gain in performance that they produce given the complexity of the data sets used here. Specifically, tree-based ML models could generally match F1 scores of the more complex DL models in both tasks. Of the DL methods, Attention and Hybrid performed best on medium resource record linkage (RecL) tasks with around 12,000 samples, achieving F1 scores outperforming the ML models by 4.6-15.7%. On the high resource de-duplication (DeDup) task with around 220,000 samples, RNN and Hybrid performed best, achieving very similar scores to the ML models. In the low-resource scenario, novel transfer, active, and transfer-active learning methods could predict (and differentiate between matches and non-matches) almost as well as the DL methods in the high-resource setting.

2 Data and Methodology

2.1 Description of Survey Data

One of the use cases, among many others (e.g. Computer Vision or Natural Language Processing) of neural de-duplication and record linkage is the alignment of textual databases by filtering out duplicates, either across multiple or within datasets. For this report, the DBLP and Scholar records of academic works were used to assess RecL. Each dataset was contains information on author, venue of release, title, and year of publication, although the number of records in each differed. The DBLP data was made up of 2,616 observations, of which about 200 were missing data on the authors and venue of release. In comparison, the Scholar dataset consists of 64,263 records, missing values on venue of release are experienced for almost 15,000 of these and 35,000 lacked information on the year of release. Since both figures make up a large proportion of the total number of data the Scholar data might be considered as 'messy' with regards to the completeness of observations, under the definition of Mudgal et al. [2018] [8] however it should still be considered as structured EM[1]. For the second task of DeDup, the 1,879 observations constituting the Cora dataset were used. The data recorded on each observation features the same as the two above and in addition the address, book title (if released in or as a book) the editor, institution, month of release, a supplementary note, the number of pages, publisher, technology, type of issue and volume. For most of these features, there are many missing values: Only for about half the records there is information on address, book title, publisher, and volume, for type, month, note and institution there are almost no observations at all (refer to Table 1). Due to problems of filling attribute values under the wrong feature column, this data set can also be considered dirty in the sense proposed by Mudgal et al. [2018] [8]. This dataset is processed by combining the `date` and `year` columns, as both contain this information, and the combination has a lot fewer missing values than the individual features. Further, special characters are removed from all datasets, and all alphabetical characters are cast as lower case.

2.2 Methodology

In this section, I describe the underlying ML, DL, transfer and active learning concepts, preliminary to introducing the models themselves and their applicability to the two tasks.

[1]Structured EM is made up of short and atomic feature values like authors, title year and others.

2.2.1 Deep Learning Concepts

The general idea behind DL systems is to receive data in a certain representation and process it to produce a different representation, often into a numeric form. In the current task, the goal is to create a mapping that will identify duplicates, which is achieved by finding similar patterns in the input data and produce numeric values that are in proximity of each other. Since transforming entire sentences into a meaningful output would require a too flexible model, the task is split onto various layers, each of which produces a slightly reduced representation of the input to find the aforementioned similarities, with subsequent layers utilising the estimated weights of the last layer (Stevens et al. [2020] [12]). Models that combine new inputs with the output of previous models in the next iteration of modelling, also known as Recurrent Neural Networks (RNNs), are very popular in the field of NLP and have been used with great success according to Stevens et al. [2020] [12] and Mudgal et al. [2018] [8], and hence they suggest themselves in our scenario. This algorithm takes the current input and produces a numerical representation through a hidden layer, the output of which is used on the next data sample, producing a final representation that is based on all samples to some extent, so RNNs can deal with sequential data. RNNs are implemented, alongside all the models used in this report, in the Python `deepmatcher` library, which is built as a wrapper on PyTorch for the specific task of RecL and DeDup. The attention algorithm extends the RNN structure by allowing the representation to place more weight on some input tokens given a certain context c (Mudgal et al. [2018] [8]). The layers have an inherent focus on those tokens associated with the context, and these will be represented higher than non-contextual tokens, which are often assumed to be noise.

2.2.2 Data Processing

Each of the models outlined above processes and uses data in a similar pattern: Textual input attributes are transformed into numerical representations and stored in vectors, which are summarised and transformed into equal lenght representations, based on which a similarity is estimated. Finally, this information is used in the classification module. Each step is introduced briefly below. All explanations are based on the basic problem of pairing two observations or entities from one or more datasets and comparing the word or sequence of words in their attributes (this can be seen in Figure 1, taken from Mudgal et al. [2018])[8]). These attributes are assumed to be the same across datasets, but no restriction is placed on the token of each entity mention[2].

[2]An entity is a unique item, and entity mentions are references to it. In this specific case, each row in the data is a mention of one entity (a certain academic work) and the columns are representing some of the attributes of that entity.

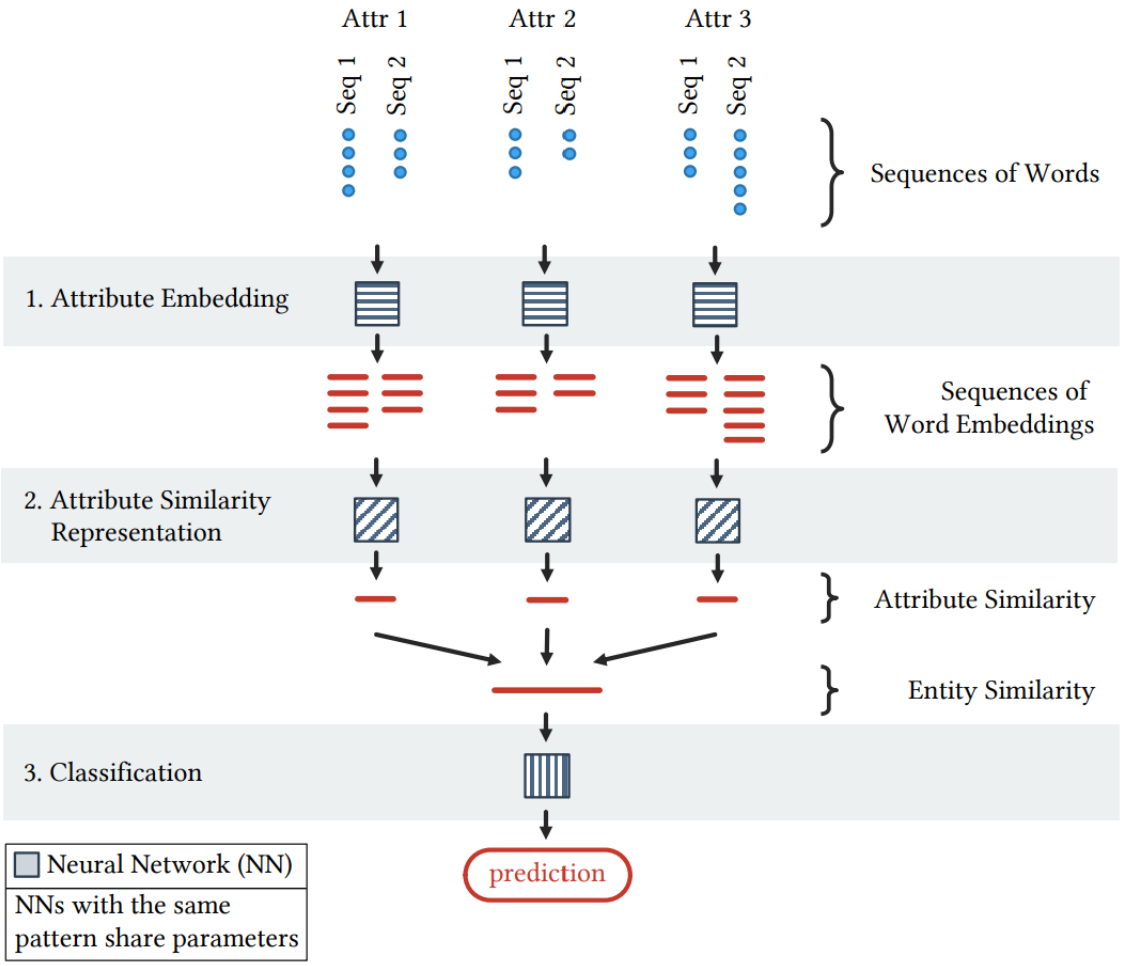

Figure 1: Model Attributes Processing Framework

Blocking

The process of blocking is used to slim the data prior to model training. The goal is to filter out obvious non-matches by comparing the tokens in a highly expressive attribute (Papadakis et al. [2016])[9]). In this case, a good candidate might be the title of the scientific work: Assuming that most of the observations in the data experience only little input error[3] in this feature, comparing the similarity of the titles allows for filtering out pairs where titles are vastly different. The trade-off here is that some true matches might be lost due to input error. For the tasks at hand, grid search cross-validation suggested a threshold of 0.41 for Jaccard similarity in the RecL (DBLP-Scholar (DS)) task and 0.35 on the DeDup task (Cora-Cora (CC)) on the title column. For the latter, this yielded about 230,000 potential pairs, in which almost all true matches were retained. Positive and negative matches were relatively balanced therefore. For the other task, about 99 percent of all true matches could be retained, and the resulting 10,500 observations were well balanced too. Further processing is required for transfer and transfer-active learning, information on which is provided in the relevant sections below.

Text Embedding

Text embeddings are naturally performed on two levels, the higher tier word embedding, treating each word as an individual entity, and character-level embeddings, which are finer but lack the ability to convey meaning in the way that entire words do (Mudgal et al. [2018][8]). The character-level encoding offers a significant advantage when dealing with spelling mistakes, rarely occurring words or impurities of other kinds and hence it will be used in this report. The idea behind the finer grained method is to train a model that will produce word embeddings for a word with certain characters

[3]filling attribute values under another (wrong) feature column, which Mudgal et al. [2018] [8] refer to as dirty EM.

present. In practice, the characters in a word are used to find a numeric representation, returned as a d-dimensional vector (Stevens et al. [2020][12]). This means, that a k word sequence in a certain attribute of a specific entity mention in the data will result in an output with $d * k$ dimensionality. To reduce the complexity of this task, the special characters (aside of numeric characters) are removed, and all remaining characters are cast as lowercase where possible (Stevens et al. [2020][12]). Further, one must choose between ready-to-use embeddings that are trained without knowledge of the actual task and data or learned embeddings that are initialised with the data at hand. The pre-trained version means savings in computing cost and it is more robust to new data, since pretrained models are often trained on quite large datasets (e.g. Wikipedia, Gutenberg Project or similar (Stevens et al. [2020][12])), whereas the learned embeddings are useful because they are trained target specific, which is beneficial when the vocabulary is highly specialised, but fail when unknown attribute tokens like out-of-vocabulary (OOV) words are used in training or prediction. Since the vocabulary of the data at hand is quite standard, the fastText [[1]] embedding is used.

Attribute Similarity

The embedding of two entity mentions is encoded as a value representing their similarity on an attribute-by-attribute basis. This is done in two steps: Firstly, since the sequences in each attribute may differ in length, the representations are summarized, which is a form of normalization to improve comparability. The summarised attributes are then passed to the comparator, which produces a similarity figure on an attribute-by-attribute basis, returning one similarity for each of j attributes, so that the output is a vector of length j. The `deepmatcher` library offers a range of possible summarizers for each of the models outlined above, and features vast customisation capabilities, which can be accessed from the `deepmatcher` project pages[4] but for the purpose of this report, working with the defaults for each model suffices. A more theoretical explanation of the summarization methods is given in Chapter 3.3 of Mudgal et al. [2018] [8].

Classification

Finally, the similarity estimates are used to determine whether the two mentions are a match on the same entity or not, by examining the distance between the similarity measures. The input vectors are assessed by cosine or Euclidean fixed distance comparator function, but functions could also be learned with neural network classification. The latter means better on-target performance, but higher computational cost.

2.2.3 Models

ML Algorithms

On both tasks, a range of ML solutions served as baseline models, to gauge the effect of using the more costly deep learning models in comparison. These models are decision tree (DT), random forest (RF), support vector machine (SVM), logistic regression (LogReg) and linear regression (LinReg). As these models are relatively basic and the literature on them vast, I will not go into detail explaining them. The computational implementation was performed with the `py_entitymatching` library [7].

DL Solutions

Settles [2009][11] summarises a broad range of possible deep learning model specification, with varying strong suits and shortcomings, some of which are more suitable than others to the tasks at hand. For this reason, four representative models have been selected to assess their efficiency on the problem. The choice largely follows the models used by Mudgal et al. [2018][8], which is beneficial as it makes the performance comparable across datasets. Beyond these models, following the work of Kasai et al. [2019][3], Settles [2009][11] and some implementation guides [5], the most highly performing model was used in an alternative algorithm in a simulated low resource environment.

Smooth Inverse Frequency (SIF)

The simplest, and therefore baseline, DL model uses weighted average for attribute summarization, which is based on the frequency of individual words, while the default absolute difference is used as the word comparator. The model is known to perform relatively well on text comparison tasks (Mudgal et al. [2018][8]) while saving computing power, as will be shown in the results section.

[4]The link to an introductory article: [6]

[5]https://www.kaggle.com/rajmehra03/a-comprehensive-guide-to-transfer-learning [Accessed 15/07/2021]

Recurrent Neural Networks (RNN)
Since this model was explained in some detail above, it should suffice to mention that a bidirectional GRU is used for attribute summarization, which is especially suitable for relational inputs, performing a forward and backward pass of the sequence through the network and concatenating the output for further processing. The attribute similarity representation is then transformed by the default method, which in this case is just concatenation, to be passed to the comparator. Of all the DL models, this model is of medium complexity.

Attention
The attention model does not come with an implemented word contextualisation scheme as the previous methods did, but with the more high-level comparator of decomposable attention, which augments the standard normalised comparison between a word in one sequence and all words in another sequence by passing this comparison together with the embedding vector to another neural net, creating aggregate comparison vectors. In classification, the vectors are then summarised (Parikh et al. [2016][10]). The model is inherently contextual, with one sequence being the context for the other. This also is a medium complexity model.

Hybrid
The hybrid model is a mixture of the prior two: It combines a bi-directional GRU RNN for contextualisation with the decomposable attention mechanism for word-by-word comparison. A second layer of attention is used for the aggregation and the result it is passed to the classifier module by form of concatenation. While this model comes with the most representational power, it also faces increased cost of high complexity, and this is confirmed by the results.

Transfer Learning
Transfer learning is the first of the models used in low resource environments, meaning that little or no labelled data is available for the target data. The concept is based on the idea that a similar dataset, call this the base, might be used for training, while prediction is done on the target data set. In this specific case, the three datasets were previously split into two tasks (CC for DeDup, DS for RecL). Since their domain - scientific publications - is the same, using data from either task as the base and training on the other is straightforward and intuitively suggests itself. In terms of pre-processing, the features of each dataset where concatenated to one feature for ease of implementation. While reducing the expressiveness of the models to some extent, due to the lack of features, it allows for direct comparison even though in this case more features are available in the Cora data than in the other two datasets. Blocking is then performed on the manipulated data, and the entire blocked data is used for training.

Active Learning
Another possibility when no labelled data exists is to iteratively train a model on small subsets of data that is labelled by a subject matter expert. If successful, a well performing model might be built with little to no effort. Since this is a form of supervised learning, some form of manual labour is required. In this scenario, the model is initialised on randomly sampled entity pairs from the previously blocked data, which are labelled and then used for training. On completion, prediction is performed, and matching scores are obtained by way of entropy, which assigns high values to uncertain (where probability of a match is close to 0.5) and low values to certain matches. The k most uncertain matches are then labelled and added to the existing training data, while the predicted label of the k most certain predictions is used as the label, and added to the training data also. The last model state with its current trainable parameters is then loaded and training progresses on this state of the model. This process is repeated for a number of i iterations. The choice of i and k depends on availability of human resources for labelling.

Transfer-Active Learning
As the name indicates, this is a combination of the previous two implementations. Specifically, the model obtained through transfer learning on the base dataset is used to get the initial predictions on the target data, upon which the active learning algorithm is performed on the target. This replaces the relatively arbitrary random selection of initial training pairs (Kasai et al. [2019][3]).

3 Results

The purpose of this report is to assess the relative performance of various ML, DL, and augmented DL solutions on the two tasks of structured entity matching and dirty entity matching. To this end, the models were prepared, and the data processed in the ways mentioned above and in the Appendix A. In accordance with the results found by Mudgal et al. [2018][8] and Kasai et al. [2019][3], this report finds that ML matchers perform competitively on structured record linkage and saving computing complexity but are outperformed by DL methods in most tasks.

3.1 Structured Record Linkage with Google Scholar and DBLP data

Figures 2 and 3 highlight the performance of ML and DL matchers respectively. All models were trained with 15 epochs and a batch size of 16. More information can be taken from the table 2. The best performing ML model is a Random Forest, achieving precision, recall and F1 scores[6] of about 0.9 and thereby outperforming the worst model, the Support Vector Machine, by 0.05, 0.15 and 0.1 respectively on each of these scores. In comparison, the ML models performed reasonably well on the structured data as could be expected given the findings of Mudgal et al. [2018][8]: The best DL model was hybrid, achieving about 0.95 precision and F1 on the task, and an AUC score of about 0.99. The former figures are a gain of 0.05 on the best performing ML model, and similar for the worst performing DL model, which is the SIF. However, it should be acknowledged that the hybrid was by far the most time intensive model, taking over 80 minutes to compute, while all other models reduced this by about an order of magnitude with base 2 in the order in which they appear in Figure 3, the red line indicates the execution time on the second y-axis. These results confirm those found by Mudgal et al. [2018][8] on the same task and suggest that all models perform very strongly on structured EM tasks with comparatively small datasets available. For this reason, the execution time might be the foremost concern. Following this, the RNN model might be the best choice, as it requires only 20 minutes of training, yet scores competitively (F1 = 0.944, Precision = 0.941, AUC = 0.979, Recall = 0.947). Further, the figures indicate that the model is very strong at differentiating positive and negative cases (i.e. matches and non-matches) and this is supported by the precision-recall curve in Figure 4, which has been used here due to the slight imbalance in the datasets (particularly in CC) and to keep the two problems comparable.

Figure 2: Results of ML models on DS task

[6]The F1 score is used as the dominant metric, as it is more expressive combination of precision and recall. The score ranges between [0, 1] where the lower level corresponds to no prediction power and the high perfect prediction power. It is preferred to other combinations as it creates convex trade-off curves (Stevens et al. [2020] [12]).

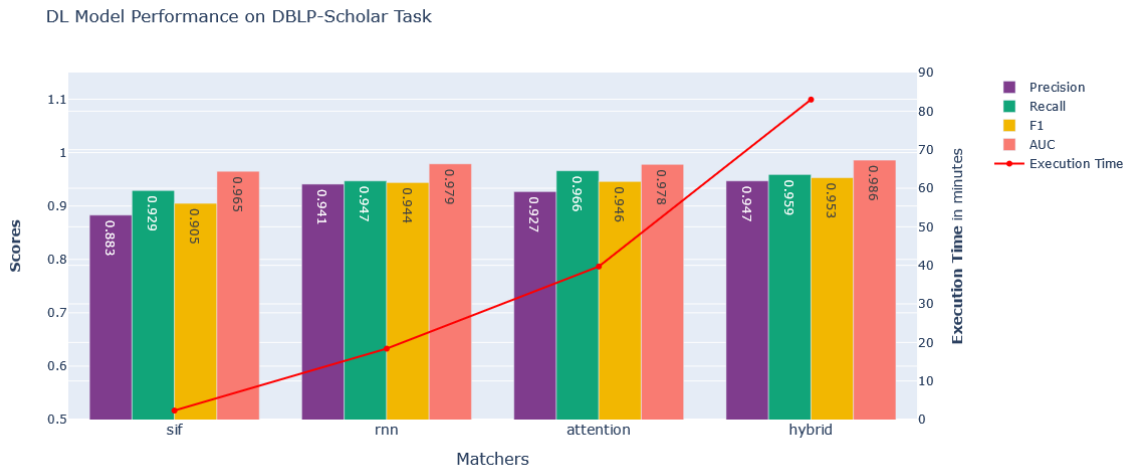

Figure 3: Results of DL models on DS task

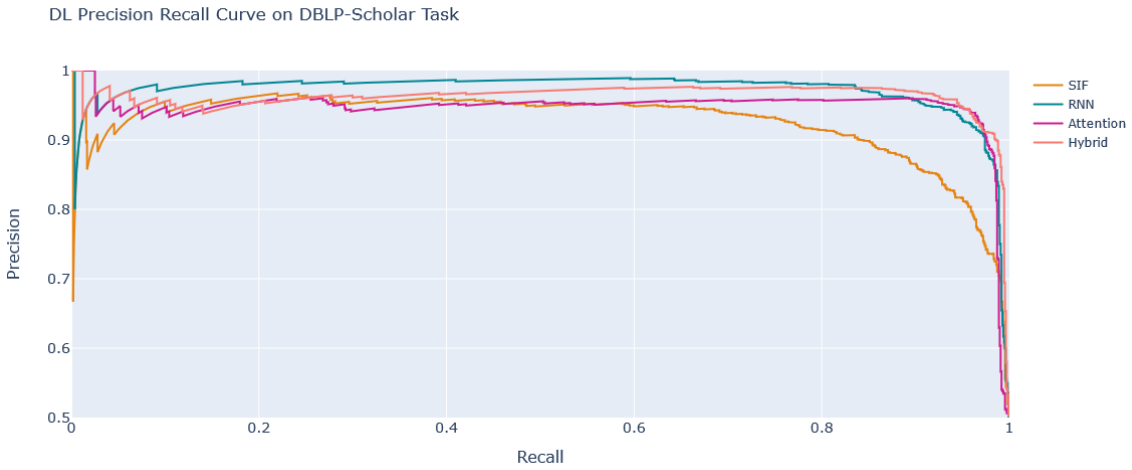

Figure 4: Precision-Recall of DL models on DS task

3.2 Dirty De-Duplication with Cora data

According to Mudgal et al. [2018][8], the DL models should exhibit improved performance as the availability of data increases, and indeed, that could be observed in this task: Some of the ML and DL models achieved F1, Precision and Recall scores of about 0.99 and higher. Figure 5 shows the ML performance, and it can be seen immediately that tree-based solutions performed best, with both the random forest and decision tree algorithm achieving scores of about 0.999, while being relatively time efficient. More precisely, these scores are a gain of 0.05, 0.08 and 0.02 in F1, precision and recall respectively over the baseline linear regression model. Both logistic regression and the SVM algorithm performed similarly to the baseline model. In comparison, the highest scoring DL models (specifications can be seen in Table 3, results in Figure 6), namely RNN and Hybrid, were trained for about 2 hours each. The simpler SIF model could not compete with the other DL models and the same goes for Attention. For this reason, while the former is further useful as a baseline, the latter might be disregarded. In theory, these models should be more flexible, but no indication of that could be found here, and therefore, the ML models might be preferable. Despite being of little use - given these results - in the current setting, as the complexity of the data increases, the DL models become more and more performant in comparison and should thus not be disregarded (Mudgal et al. [2018][8]). To sum up, the ML tree-based methods and the DL methods of Hybrid and RNN performed

extremely well and were efficient at differentiating between matches and non-matches, despite there being a slight imbalance in the data.[7], as can be seen from the precision recall curve in Figure 7.

Figure 5: Results of ML models on CC task

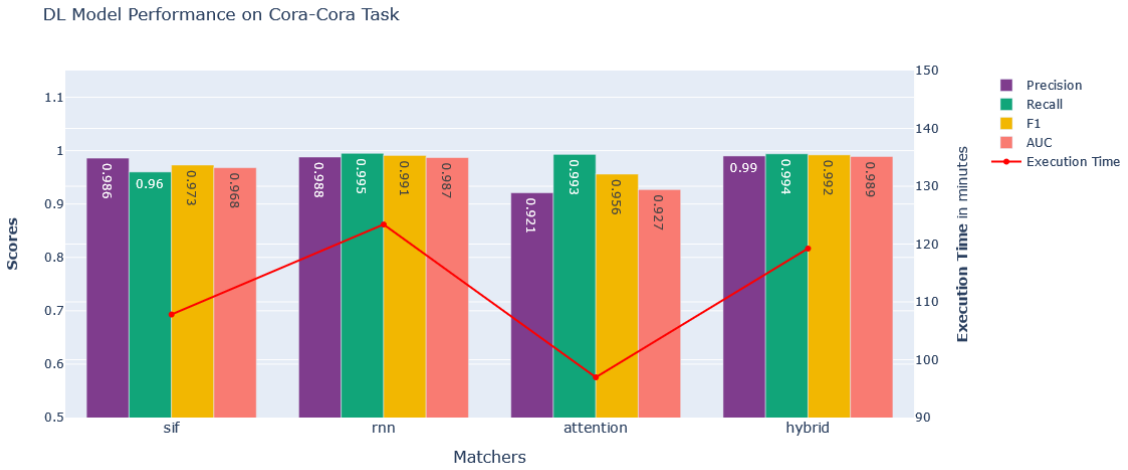

Figure 6: Results of DL models on CC task

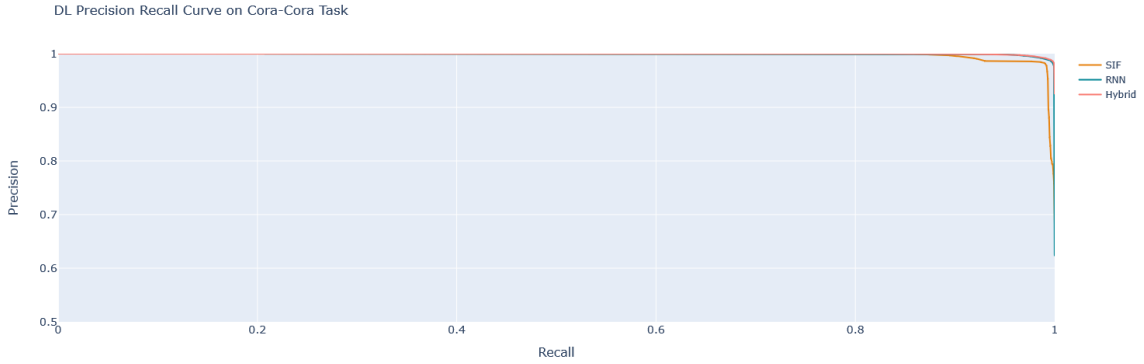

Figure 7: Precision-Recall of selected DL models on CC task

[7]This might come as a surprise, but it is due to a design choice: Blocking of the same two datasets results in a dataset in which the entity mentions containing the observation at index one from the 'left' data and index two at the 'right' data, as well as that of index two in 'left' and index one at 'right' are present. These are obviously the same pair, the only difference being their ordering. Despite not adding much information, the duplicates are kept for reinforcement of the positive case.

3.3 Transfer Learning

Following the model description, a single feature vector is created in each dataset from all input features. Based on this, blocking is performed on the DS and CC data, yielding 11,322 (out of which 5,240 are true matches) and 233,183 entity mention pairs (out of which 129,109 are true matches) and two hybrid models are trained on these base datasets, with batch size 16 and 128, respectively. After 15 epochs of training, both models perform similar to the normal DL algorithms when evaluated on base, and predictions are performed on the relative target The model specifications can be seen in Table 4. The performance recorded on these predictions can be seen in Figure 8: The model with CC base and DS target performs reasonably well, achieving around 0.8 F1, 0.74 precision, 0.9 recall and 0.86 AUC scores. The volatility of the scores is probably due to the relatively small prediction set. Similarly, the model based on DS and used for prediction on CC achieves relatively high performance, with all scores around 0.84. This indicates that despite the base data differing in the type and number of input features, the models are flexible enough to achieve relatively high accuracy on predicting both positive and negative cases.

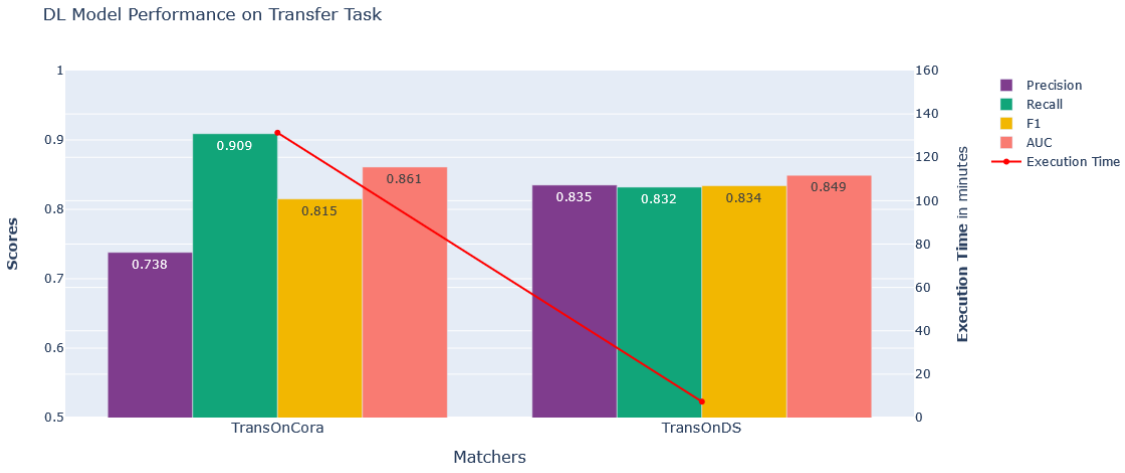

Figure 8: Performance of Hybrid model in Transfer Learning

3.4 Active and Transfer-Active Learning

The active algorithm performs an initialisation on 200 randomly selected entity pairs, which are labelled and used for training a hybrid model. On every following iteration, 20 certain and uncertain predictions are selected, and the algorithm progresses in the way described above. On DS, the hybrid model is defined with 15 epochs, and trained for 20 iterations, meaning that on the final stage of training, 1,000 labelled samples have been accumulated. The specifications for all active models can be seen in Tables 4 and 5. The model is validated on a 1,000 samples [8] and tested on the same set that the training samples stem from. This means the set for testing reduces over time by a small amount, which should generally be avoided, but given the size of the full set, it appeared necessary to retain both a relatively large test set and a large pool from which to obtain training samples, to ensure that no especially important observations for training are filtered out with a normal train, validation, and test split. These manipulations resulted in a stable algorithm, achieving a performance on the same level as the standard DL models, with AUC at 0.98, F1 at 0.95 and Precision at 0.92. Each of these scores was significantly higher than the baseline model, where the active selection by the prediction match score was replaced with a random selection of 40 samples on each iteration of the algorithm. This can be seen in Figure 8. Additionally, it can be seen that the transfer-active algorithm achieves high F1 scores after very little labelled samples, and matches the standard active learning in

[8] This follows the idea that the number of labelled data available is limited. Hence, a reduced validation set is constructed.

most scores. However, since the gains are minimal, it is questionable whether the gain is worth the additional computational complexity.

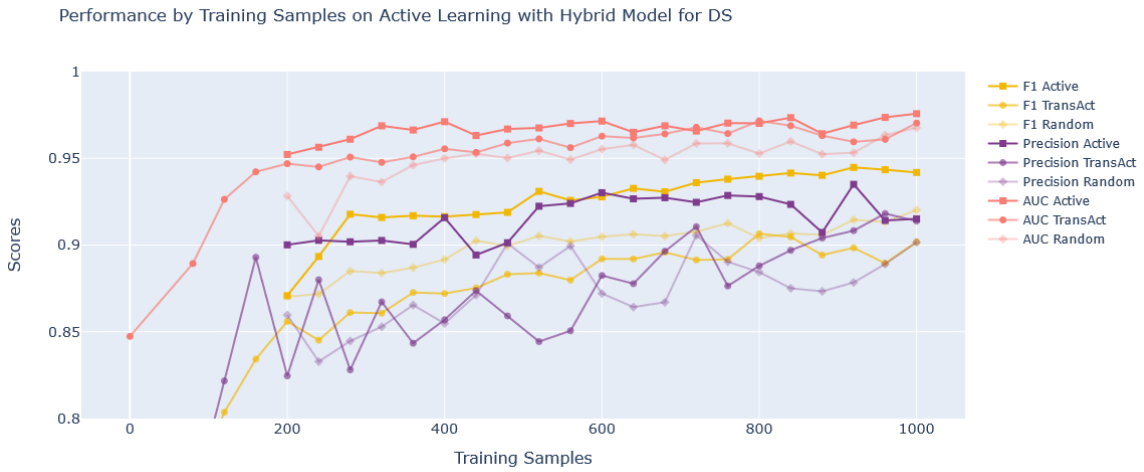

Figure 9: Performance of Hybrid model in Active Learning Task

For the DeDup task on CC, the general algorithm was carried out in the same way, with two important exceptions: Firstly, due to the large amount of data available, a standard split into training, validation (1,000 samples) and testing data was carried out, so that the model was scored on the independent test set and the selection of training observations was performed on a candidate set. This candidate set was artificially reduced to 80,000 observations by sub-sampling, to reduce the time to prediction. Secondly, the hybrid model was trained with batch size of 16 and 10 epochs for only 5 iterations. These attributes were chosen to reduce the computational cost. A similar pattern, albeit not as strong, could be observed here: The active learning performed better than the randomly initialised model, and competitively to the DL model when all the data was available, and no resource scarcity existed. Similar to the DS task, the transfer-active model achieved competitive scores after 400 observations, but it did not seem to provide gains overall. Additionally, the same consideration of training time should be applied here, and hence the active learning model might be preferred, if manual labour is more readily available than computing power.

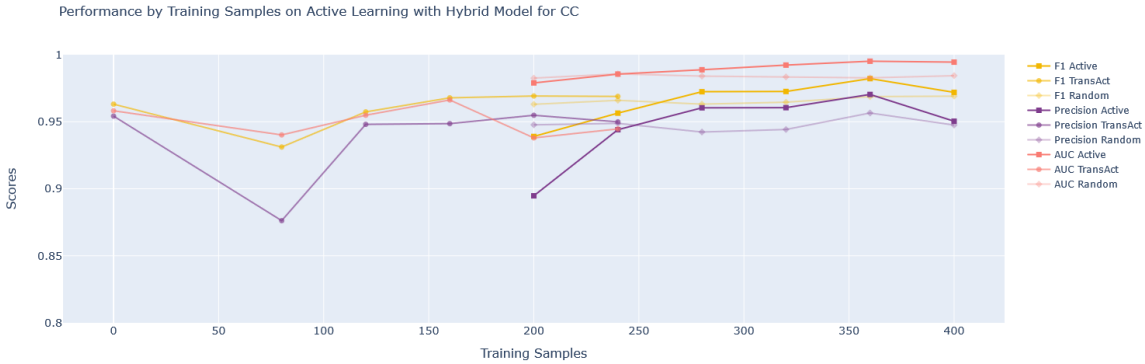

Figure 10: Performance of Hybrid model in Active Learning Task

These results confirm the findings of Kasai et al. [2019][3] and raise some questions that might be interesting to examine in future research, posted in the next section.

4 Conclusion

This report aimed to assess the predictive power of various ML, DL, and other algorithmic solutions on both structured and dirty EM tasks, in low and high resource environments. Extensive tests showed that while advanced DL algorithms can outperform the ML models on most tasks, the high computational cost does not seem to be set off by the gain in performance that they produce on the data sets used here. Specifically, the tree-based ML models could generally achieve the highest scores in both tasks. Of the DL methods, Attention and Hybrid performed best on the DS RecL task, while RNN and Hybrid performed best on the CC-DeDup task. In the low-resource scenario, novel active, and transfer-active learning methods could predict (and differentiate between matches and non-matches) almost as well as the DL methods in the high-resource setting. Further, the scores obtained confirm those found by Mudgal et al. [2018][8] and Kasai et al. [2019][3]: All models perform very well in all metrics used. However, the transfer learning by itself did not achieve very high results, and only became highly effective in combination with active learning, but since the active learning method performed as well on its own, the choice should be based upon the availability of both manual labour and compuatitonal power Finally, a few interesting problems remain for future work. It might be interesting to gauge the performance of other DL and ML models than the hybrid model chosen in this report. As the goal of these novel methods is to save manual labour in low-resource scenarios, performing transfer learning on a base set with an entirely different domain could lead to new possibilities in research if effective. The flexibility of the models used certainly suggests that this might be possible if embeddings and trainable components of the model are specified carefully, however this might be due to the similarity of the underlying data.

References

[1] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[2] P. Christen. *The Data Matching Process*, pages 23–35. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[3] J. Kasai, K. Qian, S. Gurajada, Y. Li, and L. Popa. Low-resource deep entity resolution with transfer and active learning. *CoRR*, abs/1906.08042, 2019.

[4] P. V. Konda. *Magellan: Toward building entity matching management systems.* The University of Wisconsin-Madison, 2018.

[5] R. Mehrotra. A comprehensive guide to transfer learning. `https://www.kaggle.com/rajmehra03/a-comprehensive-guide-to-transfer-learning`. [Accessed: 2021-07-08].

[6] S. Mudgal. Matching models. `https://nbviewer.jupyter.org/github/sidharthms/deepmatcher/blob/master/examples/matching_models.ipynb`. [Accessed: 2021-07-08].

[7] S. Mudgal. Matching models. `http://anhaidgroup.github.io/py_entitymatching/v0.1.x/user_manual/api/supported_matchers.html`. [Accessed: 2021-07-08].

[8] S. Mudgal, H. Li, T. Rekatsinas, A. Doan, Y. Park, G. Krishnan, R. Deep, E. Arcaute, and V. Raghavendra. Deep learning for entity matching: A design space exploration. In *Proceedings of the 2018 International Conference on Management of Data*, SIGMOD '18, page 19–34, New York, NY, USA, 2018. Association for Computing Machinery.

[9] G. Papadakis, J. Svirsky, A. Gal, and T. Palpanas. Comparative analysis of approximate blocking techniques for entity resolution. *Proc. VLDB Endow.*, 9(9):684–695, May 2016.

[10] A. P. Parikh, O. Täckström, D. Das, and J. Uszkoreit. A decomposable attention model for natural language inference. *CoRR*, abs/1606.01933, 2016.

[11] B. Settles. Active learning literature survey. 2009.

[12] E. Stevens, L. Antiga, and T. Viehmann. *Deep learning with PyTorch.* Manning Publications Company, 2020.

[13] F. Times. Battle for data domain. `https://www.ft.com/content/29af45f0-5048-11de-9530-00144feabdc0`. [Accessed: 2021-08-01].

Appendices

A Appendix A- Model Information

Table 1: Number of missing values in Cora data by feature

Attributes	*Missing out of 1,879*
Address	1139
Authors	3
Booktitle	974
Date	1374
Editor	1422
ID	0
Institution	1631
Journal	1244
Month	1849
Note	1807
Pages	643
Publisher	1173
Tech	1729
Title	41
Type	1766
Volume	1044
Year	654

Table 2: Deep Learning Model Specifications on DBLP-Scholar

	Models			
Attributes	SIF	RNN	Attention	Hybrid
Contextualiser	None*	GRU*	None*	GRU*
Comparator	None*	None*	decomp-attention*	decomp-attention*
Aggregator	weighted-avg*	biRNN-last-pool	divsqrt-pool*	attention-with-rnn*
Classifier	2-layer-highway*	2-layer-highway*	2-layer-highway*	2-layer-highway*
Epochs	15	15	15	15
Batch Size	16	16	16	16
Trainable Parameters	542,402	2,169,602	4,332,002	9,210,006

The DL model specification on the structured EM task on DS data. Characteristics highlighted with an asterisk (*) indicate that this is the default option. If not otherwise specified, all other optional arguments are left to their default. More information can be obtained from deepmatcher documentation: https://anhaidgroup.github.io/deepmatcher/html/deepmatcher.html.

Table 3: Deep Learning Model Specifications on Cora-Cora

	Models			
Attributes	SIF	RNN	Attention	Hybrid
Contextualiser	None*	GRU*	None*	GRU*
Comparator	None*	None*	decomp-attention*	decomp-attention*
Aggregator	weighted-avg*	biRNN-last-pool*	divsqrt-pool*	attention-with-rnn*
Classifier	2-layer-highway*	2-layer-highway*	2-layer-highway*	2-layer-highway*
Epochs	10	10	10	10
Batch Size	128	128	128	128
Trainable Parameters	1,083,302	7,185,302	4,332,002	9,210,006

The DL model specification on the structured EM task on DS data. Characteristics highlighted with an asterisk (*) indicate that this is the default option. If not otherwise specified, all other optional arguments are left to their default. More information can be obtained from deepmatcher documentation: https://anhaidgroup.github.io/deepmatcher/html/deepmatcher.html.

Table 4: Novel Learning Algorithm in Low-Resource Scenarios - Model Specifications

	Models				
Learn Type	*Attributes*	Active - DS	Active - CC	Transfer - DS	Transfer - CC
Transfer	Contextualiser	None	None	GRU*	GRU*
	Comparator	None	None	decomp-att*	decomp-att*
	Aggregator	None	None	divsqrt-pool*	att-with-rnn*
	Classifier	None	None	2-layer-hwy*	2-layer-hwy*
	Epochs	None	None	15	15
	Batch Size	None	None	16	128
	Trainable Parameters	None	None	2,798,703	2,798,703
Active	Contextualiser	GRU*	GRU*	None	None
	Comparator	decomp-att*	decomp-att*	None	None
	Aggregator	att-with-rnn*	att-with-rnn*	None	None
	Classifier	2-layer-hwy*	2-layer-hwy*	None	None
	Epochs	15	15	None	None
	Batch Size	16	16	None	None
	Iterations	20	5	None	None
	Labelled Pairs	1,000	400	None	None
	Trainable Parameters	9,210,006	33,136,817	None	None

Model specification in the low-resource task on DS and on CC data. Characteristics highlighted with an asterisk (*) indicate that this is the default option. If not otherwise specified, all other optional arguments are left to their default. More information can be obtained from deepmatcher documentation: https://anhaidgroup.github.io/deepmatcher/html/deepmatcher.html.

Table 5: More Novel Learning Algorithm in Low-Resource Scenarios - Model Specifications

	Models		
Learn Type	*Attributes*	TransAct - DS	TransAct - CC
Transfer	Contextualiser	GRU*	GRU*
	Comparator	decomp-attention*	decomp-attention*
	Aggregator	divsqrt-pool*	attention-with-rnn*
	Classifier	2-layer-highway*	2-layer-highway*
	Epochs	15	10
	Batch Size	16	128
	Trainable Parameters	2,798,703	2,798,703
Active	Contextualiser	GRU*	GRU*
	Comparator	decomp-attention*	decomp-attention*
	Aggregator	attention-with-rnn*	attention-with-rnn*
	Classifier	2-layer-highway*	2-layer-highway*
	Epochs	15	15
	Batch Size	16	16
	Iterations	25	10
	Labelled Pairs	1,000	400
	Trainable Parameters	9,210,006	33,136,817

Model specification in the low-resource task on DS and on CC data. Characteristics highlighted with an asterisk (*) indicate that this is the default option. If not otherwise specified, all other optional arguments are left to their default. More information can be obtained from deepmatcher documentation: https://anhaidgroup.github.io/deepmatcher/html/deepmatcher.html.