Entity Resolution with PyTorch Geometric

Francesco Pugnaloni

May 2023

1 Introduction

Cappuzzo et al. [1] propose EmbDI, a framework that uses a graph as an intermediate representation for relational tables to generate embeddings of schema elements such as tuples, tokens, and columns. Their tests proved that this approach causes a lower loss of information compared to the frameworks that consider tuples as sentences.

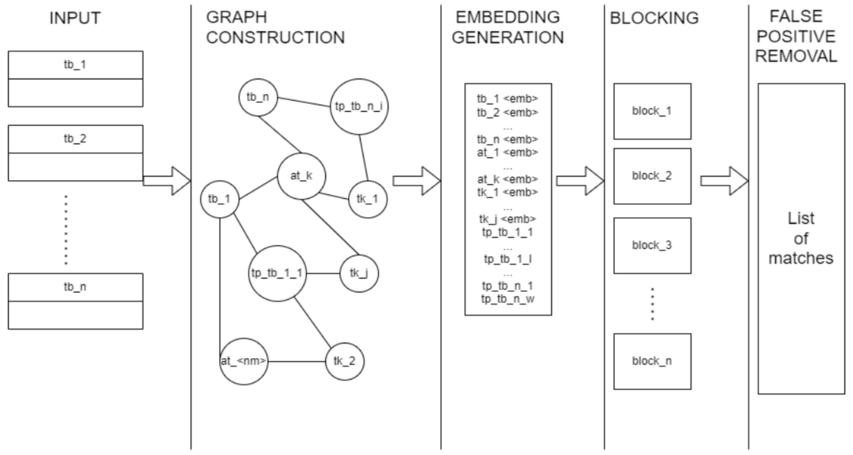
In my project, I re-implemented from scratch EmbDI using PyTorch geometric to generate the embeddings. My objective was to test the performances of this deep learning library based on Graph Neural Networks on the entity resolution task.

In the following sections are explained the architecture of the framework and are listed the experimental results obtained.

2 Architecture

The framework is composed of a pipeline of operations that given a set of tables provides in output a list of "candidate matches" after a blocking phase.

The pipeline is the following:



- *Input*: a list of paths to CSV files containing the tables to resolve.
- *Graph Construction*: the tables are processed to generate a unique graph. It is not guaranteed to be connected, but there are no isolated nodes.
- *Embedding Generation*: every node in the graph is embedded using the node2vec embedding algorithm [2]
- *Blocking*: in this phase an algorithm that makes use of the "faiss" library is used.
- *False Positive Removal*: this block is not implemented in the library and it is up to the user to validate the results provided.

2.1 Graph Construction

The graph used has the same structure as the one from EmbDI, it is a tripartite graph with three colours of nodes and two categories of undirected and unweighted edges.

The nodes are:

1. *Column nodes*: one for every distinct column name in all the input tables.
2. *Tuple nodes*: a different one for every tuple in all the input tables.
3. *Token nodes*: one for every distinct token present in the cells of all the input tables, we consider as a token every element contained in a cell. Note that if a cell contains a sentence this will be split and every word will be a separate token.

The edges are:

1. Token_Column: every token is linked to all the columns where it appears.
2. Token_Tuple: every tuple is linked to all the tokens that appear inside it.

2.2 Embedding Generation

The framework generates an embedding for every node in the graph using the Node2Vec implementation provided by PyTorch Geometric. In essence, this model generates for every node a user-provided number of "random walks" building a sentences corpus. This corpus is fed into a word2vec model that generates an embedding for every word inside it. The words represent the nodes in the graph, so this trick will provide the desired embeddings.

2.3 Blocking

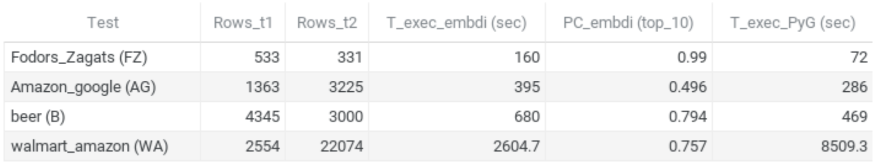
This operation is divided into two steps:

1. The framework generates for every tuple its "closest_list", i.e., the list of the n closest tuples in the embedding space where n is a user-defined number.
2. The framework generates the list of matches: two tuples are considered a match if they co-occur in each other's "closest list".

3 Testing

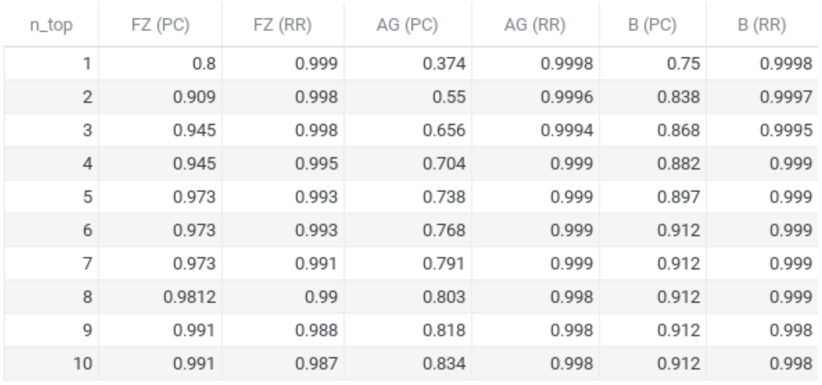
The test performed is the same as Cappuzzo et al. [1] : the research of tuples that represent the same real-world entity provided two input tables assumed to be "clean", i.e., tables without matches inside them.

Here are compared the execution times of EmbDI and PyTorch geometric and the "pairs completeness" (PC) of EmbDI's matches:



Test	Rows_t1	Rows_t2	T_exec_embdi (sec)	PC_embdi (top_10)	T_exec_PyG (sec)
Fodors_Zagats (FZ)	533	331	160	0.99	72
Amazon_google (AG)	1363	3225	395	0.496	286
beer (B)	4345	3000	680	0.794	469
walmart_amazon (WA)	2554	22074	2604.7	0.757	8509.3

Here is shown how "Pair completeness" (PC) and "Reduction Ratio" (RR) vary in PyTorch geometric with different choices of n (length of the "closest_list" generated during the blocking phase) in the various datasets:



n_top	FZ (PC)	FZ (RR)	AG (PC)	AG (RR)	B (PC)	B (RR)
1	0.8	0.999	0.374	0.9998	0.75	0.9998
2	0.909	0.998	0.55	0.9996	0.838	0.9997
3	0.945	0.998	0.656	0.9994	0.868	0.9995
4	0.945	0.995	0.704	0.999	0.882	0.999
5	0.973	0.993	0.738	0.999	0.897	0.999
6	0.973	0.993	0.768	0.999	0.912	0.999
7	0.973	0.991	0.791	0.999	0.912	0.999
8	0.9812	0.99	0.803	0.998	0.912	0.999
9	0.991	0.988	0.818	0.998	0.912	0.998
10	0.991	0.987	0.834	0.998	0.912	0.998

The results show that PyTorch geometric, in general, has lower execution times than EmbDI, and it suggests that the usage of GNNs speeds up the embedding generation. There is only one exception to this: the test performed in the "Walmart-Amazon" dataset is significantly faster in EmbDI. The reason for this performance gap is that EmbDI applies optimizations to the entity matching between unbalanced relations such as WA_A (2554 rows) and WA_B (22074 rows). Unfortunately, these optimizations are in the random walk generation phase and can not be implemented in PyTorch geometric because this operation is transparent to the user.

4 Conclusion

The usage of graphs to embed schema elements appears to be a promising research field and the GNNs proved to be a good tool to work with data structures of this kind. During my master's thesis I plan to continue researching on this topic extending this approach to the "table matching" task, i.e., the research of similar tables according to the "table overlap" similarity measure.

References

[1] Riccardo Cappuzzo, Paolo Papotti, and Saravanan Thirumuruganathan. Creating embeddings of heterogeneous relational datasets for data integration tasks. In David Maier, Rachel Pottinger, AnHai Doan, Wang-Chiew Tan, Abdussalam Alawini, and Hung Q. Ngo, editors, *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*, pages 1335–1349. ACM, 2020.

[2] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In Balaji Krishnapuram, Mohak Shah, Alexander J. Smola, Charu C. Aggarwal, Dou Shen, and Rajeev Rastogi, editors, *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 855–864. ACM, 2016.