IDAGEmb : An Incremental Data Alignment Based on Graph Embedding

Oumaima El Haddadi, Max Chevalier, Bernard Dousset, Ahmad El Allaoui, Anass El Haddadi, Olivier Teste

IDAGEmb : An Incremental Data Alignment Based on Graph Embedding

Oumaima El Haddadi[1,2][0009−0008−7951−2231], Max Chevalier[1][0000−0001−5402−6255], Bernard Dousset[1], Ahmad El Allaoui[2,3][0000−0002−8897−3565], Anass El Haddadi[2,3][0000−0002−3338−2477], and Olivier Teste[1][0000−0003−0338−9886]

[1] IRIT, SIG, Toulouse University, CNRS, France
firstname.lastname@irit.fr
[2] LSA, SDIC, Abdelmalek Essaadi University, Tetouan, Morocco
[3] lastname.firstname@uae.ac.ma

Abstract. In the evolving digital environments, information systems are faced with a myriad of challenges such as data heterogeneity, the dynamic nature of data and integration complexities. These challenges impact on decision-making and data integration processes. We define data alignment as the process of aligning columns from different tabular sources using their schema and instances. Data alignment is emerging as an essential solution, ensuring data consistency between different sources and enabling effective integration and decision-making. However, existing solutions fail to take into account the dynamic nature of data in an incremental way. This study presents an incremental methodology that uses dynamic graph embedding techniques to progressively refine data alignments. Although the use of graph embedding techniques for data alignment is well established, their integration into incremental processing approaches remains less explored. This research attempts to fill this gap by evaluating the potential of incremental graph embedding techniques for data alignment. The adoption of this incremental technique has significantly improved the management of heterogeneous data in dynamic environments, while optimizing resource usage. Likewise, this study brings a new perspective to the field of data alignment at it aims to highlight the usefulness of dynamic embedding techniques for the exploration of dynamic datasets.

Keywords: Heterogenuous Data · Incremental Data Alignment · Dynamic Environment · Graph Embedding.

1 Introduction

The evolution of data (schema and instances) within information systems requires advanced strategies for data alignment, which is vital for the integration and interoperability of heterogeneous datasets. Data alignment is no longer a static process but one that must evolve to reflect the ongoing changes within

data environments. To address this complexity, our research delves into incremental data alignment, thus leveraging the potential of dynamic embedding techniques to facilitate continuous alignment adjustments. Specifically, we detail the selection of source and target tables to ensure a complete understanding of the matching process. The matching status of each source column (1:1 match, partial match or no match) is examined in detail to highlight the complexities involved.

We define data alignment as the process of aligning columns from different tabular sources using their schema and instances. Moreover, current embedding-based methods for data alignment (i.e. schema matching), such as those proposed by Cappuzzo et al.[1] and others [2–4] have shown promising results. However, the integration of an incremental perspective in these methods has not been explored. Our study aims to address this issue by evaluating how incremental embedding techniques can be applied to the incremental data alignment, thereby enhancing their adaptability and relevance.

The ability of embedding techniques to incorporate incremental changes is well recognised in some domains such as representation learning, as shown by models like Online Node2Vec and StreamNode2Vec [5, 6]. Nevertheless, their application to data alignment has not been explored. Our research aims to fill this gap by investigating the adaptation of dynamic embedding methods to the requirements of evolving data alignment.

In this context, our study will focuses on several key questions that aim at clarifying the capabilities and performance of our incremental alignment method. These questions are essential to ensure that the approach is not only theoretically sound, but also practical for managing the dynamism of today's data ecosystems:

- RQ1 : How does the proposed incremental alignment method compared with traditional techniques in terms of precision, recall and other key measures?
- RQ2 : Can the proposed incremental approach based on graph embedding significantly reduce resource usage compared with static data alignment methods?
- RQ3 : Given the potential variations in data models, how does the method guarantee consistent and reliable data alignment?

By addressing these issues, our research can extend the theoretical foundations and practical implementations of incremental data alignment. With the increasing growth of data and the dynamic nature of data, the need for incremental alignment methods becomes more crucial for data management.

This paper is organised as follows: Section 2 provides a background on data alignment and representation learning, highlighting the shortcomings of static alignment approaches and identifying the gaps that our research seeks to fill. In Section 3, we detail the methodology of the incremental approach. To do so, we attempt to explain the concept and the incremental embedding method that we adapted along with the processes steps. Section 4 presents the experimental setup, results, and discussion that follows, designed to address the research questions mentioned above. It includes a description of the datasets, metrics and measures that will be used to evaluate the performance of the incremental

approach. Finally, section 5 concludes with a summary of our contributions and suggestions for future research.

2 Background

The landscape of data alignment (i.e. schema matching) methodologies is rich and varied. It addressess the critical need for effective management of data and schema heterogeneity. This section explores a range of existing approaches, while highlighting the absence of incremental data alignment approaches in current literature, as outlined in Section 2.1. and Section 2.2 introduces dynamic graph embedding techniques, which we consider potential candidates for the development of incremental data alignment strategies. Finally, Section 2.3 provides a discussion of these topics.

2.1 Existing Data Alignment Approaches

Data alignment is a process of matching different data element, schema and instances that address the challenge of data heterogeneity. This process has traditionally been met with non-incremental alignment methods, broadly classified into schema-based, instance-based, and hybrid approaches [7]. Bernstein et al. [8] provide a comprehensive classification of these methods, highlighting their application in various contexts, from requirements-focused storage solutions like data integration and schema mapping [9] to broader applications such as data lakes [10] and ontology matching [11]. A particularly promising avenue in this field is the use of embeddings for alignment, which offers a robust framework for representing and comparing data from disparate sources [1, 2]. Through the use of graphs, these methods do well in building complex connections between different data parts, greatly improving the data alignment process.

The advent of machine learning and natural language processing technologies has brought about significant advancements in data alignment methodologies. Incorporating representation-based learning, especially embedding, these modern approaches have redefined alignment strategies. Embeddings, essentially numerical vector representations of schema attributes or instances, leverage distance-based methods to compute alignments. Their applications extend beyond traditional databases to graph representations, offering a nuanced approach to data alignment [4, 12].

2.2 Graph Embedding in Representation Learning

Exploring the domain of graph embedding has revealed several methods tailored for incremental embedding in temporal graphs. Each method addresses different aspects of dynamic graph analysis [13]. For instance, Online-Node2Vec [5] innovatively updates dynamic network representations in real-time. Liu's approach [6] involves generating embeddings for new nodes and revising the embeddings of nodes influenced by these additions. The FLDNE Framework [14] focuses on

evolving networks, employing a combination function and alignment mechanism to adapt standard embedding techniques across various time steps.

These methods typically begin with an initialization phase using conventional embedding techniques. One of the main challenges they face is updating dynamic node representations (e.g. when new nodes are added). Most of these methods give priority to node additions, leaving a significant gap in the ability to handle node deletions and modifications in temporal networks. This gap highlights the need for more comprehensive solutions capable of handling fully dynamic sources (adding, deleting or modifying a schema element or instance in the source).

2.3 Discussion

Despite the absence of incremental data alignment to manage data evolution, dynamic embedding methods have shown potential. Even though they were not initially designed for data alignment, their ability to adapt to dynamic environments makes them promising candidates for application in this field. Furthermore, incremental graph embedding and representation learning techniques have demonstrated their efficacy in capturing the evolution of relationships and structural changes within complex data constructs [5, 13].

Adapting these methodologies for data alignment offers an interesting way of developing solutions that can accommodate real-time schema modifications, thereby improving the flexibility and efficiency of data alignment processes in dynamic environment.

3 Methodology

This section describes the methodology of the incremental approach, IDAGEmb. Section 3.1 introduces the concept, while Section 3.2 details the preliminaries and the algorithms implemented.

3.1 Research Design

Incremental data alignment is emerging as an essential solution for managing heterogeneous and dynamic data, as it avoids the need to recompute alignments from scratch. This approach is designed to manage data changes, ensuring that the alignment process remains both efficient and adaptive.

The changes taken into account fall into three main categories : the addition, modification, and deletion of 1) schema elements, 2) the entire schema itself, and 3) the instances within the data sources. For example, modifications have been planned to cover both minor changes (e.g. the removal of vowels from attribute names) and major changes (e.g. the complete encoding of attribute names). By considering these diverse evolutions, we aim to ensure that the proposed approach adequately addresses the dynamic nature of data sources and captures any changes that may impact the alignments. It is essential to account for these evolutions to maintain the efficiency and effectiveness of the alignment

process over time. Moreover, such incremental dimension aims at reducing the computational cost of identifying the matching in data alignment (only matching concerned by data evolution should be updated).

Building on advances in representation learning, in particular studies of dynamic graph embedding, we have developed an incremental data alignment framework inspired by the [6] method, focusing on both additions of new nodes and modifications to existing nodes. Unlike previous approaches that focused solely on adding data, our goal is broader, targeting a full range of data source evolution to ensure up-to-date alignments. This strategy is essential for maintaining accurate data alignments in changing environments, which is crucial for sectors requiring real-time data analysis, such as healthcare IT and dynamic database management, improving system efficiency and reducing redundant computations.

3.2 Preliminaries

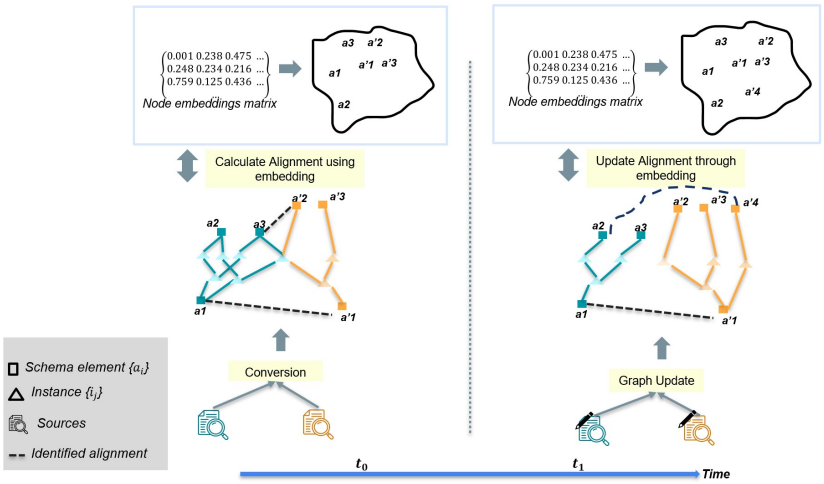

Fig. 1: Outline for the principle of Incremental Data Alignment.

To provide an overview of the process shown in Figure 1, we begin by transforming the two data sources into a graphical representation. This allows the schema elements as well as the instances to establish a basic structure for later analysis. The process progresses with the generation of embedding for each node via graph embedding techniques, a critical step in capturing the nuanced characteristics of each node in a high-dimensional space.

Next, the similarity between nodes is determined using a vector distance measure, such as cosine similarity. This process facilitates the identification of data

source columns in different graphs by quantifying the similarity of their embeddings. When changes are detected in the data sources, the strategy switches from recomputing the entire alignment to a more efficient approach. Specifically, we update the embeddings of modified nodes according to the algorithm proposed by [6], thus avoiding the process of recomputing from scratch. The foundational equation used within this algorithm is presented as follows:

$$Z_t^{(i+1)} = Z_t^{(i)} + \alpha_i \Gamma(i)[I_{k\times k} + \alpha_i^2(\Gamma(i))^T\Gamma(i)]^{-1/2} \tag{1}$$

where:

- $Z_t^{(i)}$ is the embedding matrix at iteration i.
- α_i is the step size at iteration i.
- $\Gamma(i)$ is the search direction in the tangent space of the Stiefel manifold at iteration i.
- $I_{k\times k}$ is the identity matrix of size $k \times k$.

Each matrix embedding, as described by the set of n nodes vectors, is given by: $Z^{t_i} = [(z_{v_1})^T, z_{v_2})^T, ..., z_{v_n})^T]$.

By adopting incremental embedding methods, we can reduce computational overhead and ensure efficient updates to the alignments without the necessity to recompute the entire set of alignments. This process is depicted in Figure 1 and elaborated through Steps 1-6 below, highlighting the seamless transition from data transformation to alignment updates.

1. Transform each data source (DS_1, DS_2) into a graph representation, denoted as $G_{t_i} = (V, E)$.
   Where t_i represents the state of the graph at time i, $V = v_1, v_2, ..., v_n$ is the a set of n nodes (vertices) and $E = \{e_{ij} = (v_i, v_j)|(v_i, v_j) \in (VXV)\}$ is a set of edges e_{ij} connecting pairs of nodes. This process involves transforming the data available at t_i in DS_1 and DS_2 into a graph that represents different types of nodes (schema nodes and instance nodes) and adding prefixes for each type to define the source node. At t_{i+1} , the graph is updated with the new state of the data sources.
2. Generate embeddings for the j^{th} nodes (v_j) using the graph embedding method StreamNode2Vec (SN2V) [15]. Let Z_{t_i} represent the set of nodes embeddings for G_{t_i}. Where $Z_{t_i} = \{z(v_j)\}$.
3. Compute the similarity between pair of nodes' embedding by the Cosine Similarity at t_i, selected for its dimentionality independence and ability to effectively identify semantic relationships[16]:

$$\text{Similarity}\,(z(v_i), z(v_j)) = \frac{z(v_i) \cdot z(v_j)}{\|z(v_i)\|\|z(v_j)\|} \tag{2}$$

4. Compute data alignment set A_{t_i}.
5. If there are changes ΔG_i in the data sources (DS_1, DS_2), update graph, then update the embedding matrix using equation (1).
6. Update data alignments set $A_{t_{i+1}}$.

3.3 Adopted Algorithm for IDAGEmb

The process of the algorithm proposed by Liu et al.[6] for real-time streaming graph embedding can be described as follows:

1. **Identify Influenced Vertices:** Identify the set of vertices V_{infl} that are most influenced by the arrival of new vertices, as detailed in Algorithm 1 in [6].
2. **Generate Embeddings for each New Vertex:** For a new vertex v, generate its embedding $z(v)$ based on the linear summation of original embeddings of other vertices based on equation 1.
3. **Adjust Embeddings of Influenced Vertices:** Update the embeddings of vertices in V_{infl} considering the influence of the new vertex.

Algorithm 1 TRANSFORM(DS_1, DS_2, t_0)

```
G_0 ← transformToGraph(DS_1, DS_2)                    ▷ Step 1
Initialise Z_{t_0} as an empty set                    ▷ Initialise embedding matrix
for all v ∈ V(G_0) do                                 ▷ V(G_0) is the set of nodes
    z^{t_0}(v) ← Embedding(v)                         ▷ Step 2
    Z_{t_0} ← Z_{t_0} ∪ [(z^{t_0}(v))^T]^T
end for
A_{t_0} ← CalculateSimilarity(z^{t_i}(v), z^{t_i}(v'))   ▷ Equation (2)
return A_{t_0}, Z_{t_0}, G_0
```

Inspired by this algorithm and the implementation presented in [15], our development incorporates mechanisms for creating and updating graphs, as well as their respective alignments. Algorithm 1 encompasses the first three steps of our methodology, while Algorithm 3 focuses on steps 4 to 6.

4 Experiments and Results

To evaluate the approach, we designed an experimental protocol that addresses the three research questions described in the introduction. This involves detailing the set-up implemented and the dataset used. For each research question, we develop the main objective as well as the results and discussion that follows. Before discussing the procedure and results associated with each research question, we incorporate an experiment (Experiment #1) devoted to optimizing the algorithm's hyperparameters. Section 4.1 describes the set-up and data sets used, followed by section 4.2 to section 4.4 which outline the objectives and present the relevant results for each experiment.

4.1 Experiment Configuration

For the three experiments we will follow the process described in Figure 2, using the materials, datasets, and metrics/measures described below:

Algorithm 3 UPDATE(DS_1, DS_2, t, A_{t_0}, Z_{t_0}, G_0)

Initialise the set of alignment A_t
for $i = 1$ **to** $|t| - 1$ **do** ▷ Iterate over subsequent timestamps
  $G_i \leftarrow$ UPDATEGRAPH($G_{i-1}, \Delta G_i$) ▷ Step 5
  $Inf_{t_i}(u_j) \leftarrow$ DETECTINFLUENCEDNODES(G_i) ▷ Algorithm 1 [6]
  Compute embedding for new nodes
  $z^{(t_i)}(u_j) \leftarrow \frac{1}{|Inf^{(t_i)}(u_j)|} \sum_{v \in Inf^{(t_i)}(u_j)} z^{t_{i-1}}(v)$ ▷ Generated from Equation (2)
  Update $z(u_j)$ based on $Inf_{t_i}(u_j)$
  $Z_{t_i} \leftarrow Z_{ti-1}$ ▷ Initialise $Z_{t[i]}$ with previous embeddings
  for all $v_j \in$ Influenced(V) **do**
    $z(v_j) \leftarrow$ UPDATEEMBEDDING($z(v_j), Inf_{t_i}(u_j)$) ▷ Algorithm 2 [6]
    $Z_{t_i} \leftarrow Z_{t_i} + z(v_j)$ ▷ Update Z_{t_i} with new embeddings
  end for
  $A_t \leftarrow$ UPDATEALIGNMENT(A_{t_0}, Z_{t_i}) ▷ Step 6
end for
return A_{t_i} ▷ Return the alignments for the last timestamp

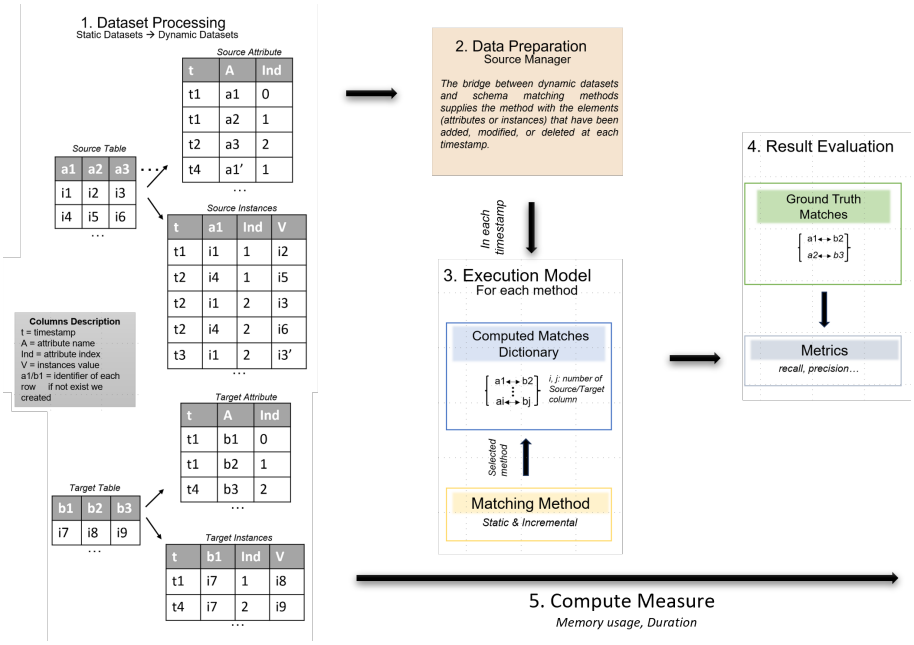

Fig. 2: Experiment Process.

Materials The experiments will be conducted on a high-performance computing node to ensure efficient processing of complex data manipulations and model training. The node specifications are as follows: - Processor: AMD EPYC 7402 2.8 GHz dual processor. - Architecture: 48 processors. - Memory: 512 GB RAM. - Operating System: Linux Centos7.

Datasets We have selected a set of datasets referenced in the literature [1, 17] as described in Table 1. Three datasets, TPC-DI, Open Data and ChEMBL, were prepared according to the methodologies described by [17]. For each dataset, we identified four pairs of source and target tables. These datasets were selected because of their complexity and heterogeneity, characterised by differences when specific attributes in the source tables differ textually from their counterparts in the target tables. In addition, we encountered cases where attributes in the source tables matched textually those in the target tables, but differed semantically. Moreover, in some cases, only one column in each table could be accurately mapped. These tables range from 11 to 43 columns and from 7491 to 23254 rows.

Additionally, we use two raw datasets from [17] — Magellan and Wikidata. Magellan comprises 7 pairs, while Wikidata contains 4 pairs, with variations ranging from 331 to 10845 rows and 4 to 20 columns. The IMDB-Movielens dataset, sourced from [1], features a source table with 4529 rows and 11 columns, and a target table with 45346 rows and 13 columns. The later three dataset present simple heterogeneity with a significant number of rows. These tabular datasets comprise numerical, textual and noisy data.

Table 1: Datasets Characteristics.

Dataset	**#Pairs**	**#Columns**	**#Rows**
TPC-DI	4	11 to 22	7491 to 14982
Open Data	4	26 to 43	11627 to 23254
ChEMBL	4	14 to 20	7500
Magellan	7	4 to 9	331 to 64263
Wikidata	4	13 to 20	5422 to 10845
IMDB-Movielens	1	11 to 12	4529 to 45346

All datasets include files representing a list of true matches, as determined by researchers, which we use to compare the matches obtained from our model.

Given that these datasets are 'static', we processed them to simulate dynamic data. This was achieved through a splitting method that applied timestamps to distinguish schema (i.e. column attributes) from instances (i.e. rows) in each table, in order to simulate the addition, modification and deletion of data. For example, the addition, modification or deletion of schema was simulated at three splits, while the addition, modification or deletion of instances occurred at four

splits. As a result, this process produced seven timestamps, assuming that the changes were not simultaneous; if they were, the number of timestamps could be lower. For each split, we selected part of the dataset. It was found that for large datasets, the starting point had to be 70% of the initial data, whereas for small datasets, it was sufficient to start with 30%. The purpose of this step (step 1 in Figure 2) is to facilitate the detection of changes in both instances and schema.

Metrics and Measure To evaluate the performance of the embedding methods and the overall model, the following metrics and measure will be used, focusing in particular on the F1 Score:

- For effectiveness:
  - F1 Score: To understand the balance between precision and recall in the model's performance.
  - Precision: Assessing the model's ability to correctly identify relevant matches.
  - Recall: Evaluate the model's capability to find all relevant instances.
- For efficiency:
  - Resource Usage: Evaluates the computational resources required by each method, including consumption and transformation duration and memory usage. This metric could compare the resource used in the static and incremental method.

Baseline methods We used static schema matching methods as a reference to compare the obtained alignments. Specifically, we selected the six methods used by [17], which fall into three categories:

- **Schema-based matching**
  - Similarity Flooding: A method that relies on graphical similarity flooding algorithms to find matches between schema elements
  - Cupid: Uses linguistic and structural analysis of schemas to match elements, taking into account both names and data types.
  - Coma_SHM: This technique, part of the COMA (Combined Approach) suite, combines several matching tools to improve the accuracy of schema matching through structural analysis.
- **Instance-based matching**
  - Coma_INS: Another variant of the COMA suite, which focuses on instances to determine matches between schemas.
  - Jaccard Levenstein: Uses a combination of Jaccard similarity and Levenshtein distance to match schema elements based on their instances
- **Hybrid matching**
  - EmbDi: An approach that integrates both schema and instance information, using embedding techniques to generate a complete match.

Our method, IDAGEmb, falls into the category of hybrid matching and is the only one that incorporates an incremental aspect.

4.2 Experiment #1: Embedding Method Selection

Objective The main objective of this study is to identify the most efficient graph embedding technique among three distinct methodologies and to choose the best hyperparameters to generate node embedding at the first iteration. The first method considered is the node2vec (N2V) algorithm [18], which is readily available in the Python library. The second is the StreamNode2Vec (SN2V) method detailed by [15], and the third is EMBDI, a graph embedding method presented in [1]. The evaluation critically examines various hyperparameters: embedding dimension, walk length, window size, number of walks and min count. This evaluation is essential to achieve an optimal balance between the consumption of computing resources and the quality of the embedding results.

Description and Results For this experiment, we focused on the following five hyperparameters for the three embedding graph methods:

- **Embedding dimension** = [64, 100, 128, 300]: Specifies the size of the vector space for embedding nodes, which is crucial for encapsulating the essential features of the graph.
- **Walk length** = [20, 40, 60, 80, 200]: Specifies the number of walks taken in each random walk, a parameter that influences the scope of local neighborhood exploration.
- **Window size** = [3, 5, 7, 10]: Defines the contextual window for the inclusion of neighboring nodes in the embedding process, influencing the amount of graph contextual information included in the embedding of each node.
- **Min count** = [0, 1, 2]: Defines the minimum frequency a node must have to be included in the embedding process, allowing us to filter out nodes with few occurrences.
- **Number of walk** = [10, 20, 30, 40, 100]: Indicates the number of random walks launched from each node, which plays an important role in the completeness of graph exploration.

Initially, we explored all possible combinations of hyperparameters for the three embedding methods, focusing on a relatively small, non-heterogeneous dataset to simplify the analysis. However, the EMBDI method proved particularly time-consuming, especially with higher dimensions. Therefore, we excluded the EMBDI method from further evaluations.

Subsequently, we conducted full experiments with the two remaining methods, N2V and SN2V, on three datasets of varying complexity: a simple dataset, an intermediate dataset characterized by its larger size, and a complex dataset with more heterogeneous columns. This strategic approach allowed us to evaluate the performance of both methods under diverse conditions, providing valuable insights into their applicability and effectiveness across datasets with varying characteristics and complexities. As a result, the F1 Score of SN2V is generally higher in most configurations than the F1 Score of N2V for the different datasets.

Based on these analyses, we determined that the optimal set of hyperparameters for different datasets is as follows: embedding dimension = 128, walk length = 40, window size = 10, minimum count = 0, and number of walks = 20.

4.3 Experiment #2: Comparison with Static Methods (effectiveness and efficiency)

Objective The main objective of this experiment is to evaluate the performance of the incremental alignment method, IDAGEmb, compared to traditional static methods. This evaluation, conducted on dynamic datasets, is designed to answer the two research questions, RQ1 and RQ2, by determining the effectiveness and efficiency of the embedding method in dynamical data environments, respectively.

Results and Evaluation In this experiment, we aim to identify columns that are similar in different dynamic data sources. The data processing step was structured in three main phases. Firstly, we exclusively added, modified and deleted schema element (i.e. attributes column). Secondly, we simulated the addition, modification and deletion of instances (i.e. value cells). Finally, we simulated modifications encompassing both schema element and instances (e.g. modifying the numerical ages to a categorical value, modifying schema element values).

In all stages, the baseline methods relying on schema-based failed to match the correct columns when the attribute value or the instances values are modified, despite their faster performance. On the other hand, methods based on instance-based matching and those that match textually also failed to detect the correct columns when the instances were modified. However, the method that match semantically can match the columns correctly after the re-execution from scratch. Similarly, the baseline based on hybrid method, Embdi, was able to correctly match columns after re-execution of the whole process.

Effectiveness: The average F1 Score for all matchers across all datasets, as plotted in Figure 3(a, b), reveals that our method, IDAGEmb, achieved the highest average F1 Score of 0.513, with a standard deviation of 0.283. In comparison, the Similarity Flooding method, with the second-highest mean, achieved an average F1 Score of 0.507, with a standard deviation of 0.352, while the EmbdI method recorded mean F1 Score of 0.416 with a standard deviation of 0.337. These results highlight the variability in performance between the matching techniques. Although the average effectiveness of Similarity Flooding is slightly lower than that of IDAGEmb, it shows considerable variability in the results, as suggested by its higher standard deviation. In contrast, IDAGEmb shows more consistent performance across the different scenarios, as indicated by its comparatively lower standard deviation. This consistency indicates that IDAGEmb may provide more stable performance under varying data conditions, offering a reliable, if slightly less accurate, matching solution compared to static methods. The high values of standard deviation and the variability observed in the results can be attributed to the different sizes, heterogeneity, and complexity of the datasets used. Improving and optimizing IDAGEmb could increase its accuracy and make it a competitive choice for data alignment in environments characterised by dynamic data.

Efficiency: The analysis of the consumption and transformation duration for all the matching methods, as shown in Figure 3(c, d), shows that our method,

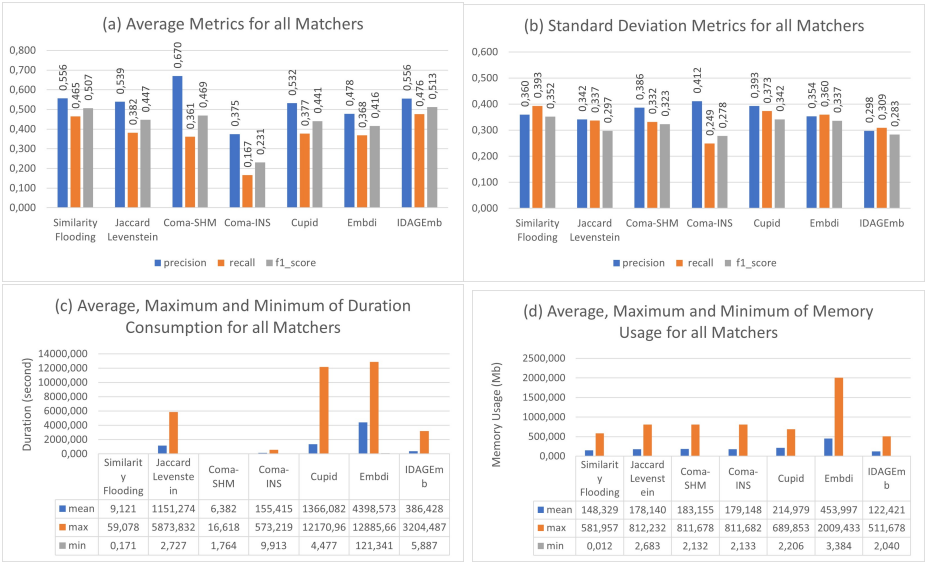

Fig. 3: Standard deviation and Mean of performance metrics (a,b) and resource usage (c,d).

IDAGEmb, requires ≈386 seconds on average, significantly less than the ≈4399 seconds required by the EmbDI method. EmbDI, a hybrid approach, also relies on graph embedding techniques similar to those used by IDAGEmb. In addition, the memory usage of IDAGEmb is significantly lower than that of EmbDI. These results indicate that IDAGEmb outperforms EmbDI in terms of processing speed and memory efficiency. In summary, IDAGEmb is more efficient than instance-based and graph-based methods such as EmbDI and Cupid. As a result, IDAGEmb appears to be a more appropriate option for data alignment tasks, particularly in dynamic data environments where resource optimization and fast processing are paramount.

4.4 Experiment #3: Model Sensitivity to Data Order Variation

Objective The objective is to study the impact of changing the order of data entries. Specifically, we examine whether the order in which certain matched columns are presented at the beginning, middle or end of the alignment process affects model performance. The aim of this test is to understand the sensitivity of the model to the order in which the data is entered and to check whether the model's performance is consistent regardless of the order of the data.

Results and Evaluation By analysing the results presented in the Figure 4, we observed that the final F1 Scores at time #5, which represent the result of the alignment after data additions and modifications, retain a notable consistency.

This consistency persists even if the datasets are randomly changed (e.g., an attribute that appears at the initial timestamp in one version of the dataset preparation may appear at the final timestamp in another). For the 'musician' dataset , the standard deviation of the final F1 Score is 0.02, implying very low variability and indicating the robustness of the alignment method. In contrast, the 'assays' dataset has a slightly higher standard deviation of 0.04. Although this indicates greater variability than the musician dataset, it still denotes a relatively stable final alignment accuracy across different dataset preparations. These observations suggest that the timing of data arrival has a negligible impact on the final alignment result.

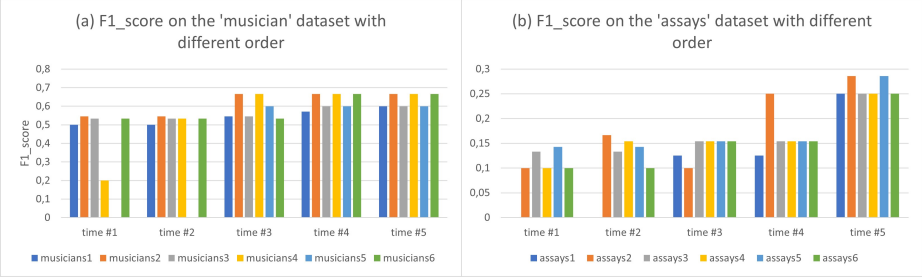

Fig. 4: F1 Score on two dataset with different order.

5 Conclusion and outlook

This research tackles data alignment in evolving information systems, characterised by data heterogeneity, the dynamic data and complex integration processes. By introducing an incremental methodology based on dynamic graph embedding techniques, this study aims to improve data alignment progressively. It addresses the gap in applying incremental graph embedding techniques to incremental data alignement by evaluating their effectiveness (RQ1), efficiency (RQ2) and senstivity (RQ3). The refined technique enhance the management of heterogeneous data in dynamic environments and optimizes resource consumption, offering a new perspective on data alignment through the integration of dynamicity for exploring constantly evolving data.

Outlook In future work, we plan to enhance the process to align more complex data by incorporating external dictionaries and leveraging models that are pre-trained on other datasets, such as BERT found in Large Language Models (LLMs). In addition, we wish to evaluate the integration of incremental data alignment into broader data management processes, in order to improve the efficiency and consistency of data processing. This exploration will contribute to more effective management of complex and dynamic data environments.

References

1. Cappuzzo, R., Papotti, P., Thirumuruganathan, S. : Local Embeddings for Relational Data Integration , in Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data, juin 2020, p. 1335-1349. https://doi.org/10. 1145/3318464. 3389742.
2. Koutras, C., Fragkoulis, M., Katsifodimos, A., Lofi, C. : REMA: Graph embedding-based Relational Schema Matching. In: EDBT/ICDT Workshops, (2020)
3. Rodrigues, D., da Silva, A. : A study on machine learning techniques for the schema matching network problem , J Braz Comput Soc, vol. 27, no 1, p. 14, déc. 2021, doi: 10. 1186/s13173-021-00119-5.
4. Hättasch, B., Truong-Ngoc, M., Schmidt, A., Binnig, C. : It's AI Match: A Two-Step Approach for Schema Matching Using Embeddings. In: 2nd International Workshop on Applied AI for Database Systems and Applications (AIDB'20) (2020).
5. Béres, F., Kelen, D. M., Pálovics, R., Benczúr, A. A. : Node embeddings in dynamic graphs. Applied Network Science **2**(4), 64 (2019). https://doi.org/10. 1007/s41109-019-0169-5
6. Liu, X., Hsieh, P. -C., Duffield, N., Chen, R., Xie, M., Wen, X. : Real-Time Streaming Graph Embedding Through Local Actions , in Companion Proceedings of The 2019 World Wide Web Conference, San Francisco USA: ACM, mai 2019, p. 285-293. doi: 10. 1145/3308560. 3316585.
7. Sutanta, E., Wardoyo, R., Mustofa, K., Winarko, E..: Survey: Models and Prototypes of Schema Matching. IJECE, ;6(3):1011, (2016).
8. Bernstein, P., Jayant, M., Rahm, E. : Generic Schema Matching, Ten Years Later. In: PVLDB, vol. 4, pp. 695–701 (2011). https://doi.org/10. 14778/3402707
9. Miller, RJ., Haas, LM., Hernandez, MA. : Schema Mapping as Query Discovery. In: Very Large DataBase conference (VLDB), pp. 77—88. (2000)
10. Alserafi, A., Abelló, A., Romero, O., Calders T. : Keeping the Data Lake in Form: Proximity Mining for Pre-Filtering Schema Matching. ACM Trans. Inf. Syst. **2**(38), 3 (2020)
11. Aumueller, D., Do, H-H., Massmann, S., Rahm, E. : Schema and ontology matching with COMA++. In: ACM international conference on Management of data (SIGMOD '05). Association for Computing Machinery, New York, NY, USA, pp. 906–908 (2005)
12. Zhao, Z., Castro Fernandez, R. : Leva: Boosting Machine Learning Performance with Relational Embedding Data Augmentation. In: Proceedings of the 2022 International Conference on Management of Data. ACM, Philadelphia PA USA, pp. 1504–1517 , (2022).
13. Barros, C.D., Mendonça, M.R., Vieira, A.B., Ziviani, A. : A Survey on Embedding Dynamic Graphs. ACM Computing Surveys (CSUR), 55, 1 - 37. (2021).
14. Bielak, P., Tagowski, K., Falkiewicz, M., Kajdanowicz, T., Chawla, N. V.: FILDNE: A Framework for Incremental Learning of Dynamic Networks Embeddings. Knowledge-Based Systems, 236, (2020). https://doi.org/10. 1016/j. knosys. 2021. 107453
15. StreamNode2Vec,https://github. com/husterzxh/StreamNode2Vec.
16. Wang, L., Luo, J., Deng, S., Guo, X.: RoCS: Knowledge Graph Embedding Based on Joint Cosine Similarity. Electronics 2024, 13, 147.
17. Koutras, C., Siachamis, G., Ionescu, A.,Psarakis, K., et al.: Valentine: Evaluating Matching Techniques for Dataset Discovery. In: IEEE 37th International Conference on Data Engineering (ICDE),(2021). doi: 10.1109/ICDE51399.2021.00047
18. Node2vec python https://github. com/eliorc/node2vec. Last accessed 10-03-2024