On the Semantic-Aware Data Imputation in Dynamic Relational Databases

Gianvincenzo Alfano[1], Sergio Greco[1], Lucio La Cava[1], Tariq Mahmood[1,*] and Irina Trubitsyna[1]

[1]Dept. Computer Engineering, Modeling, Electronics, and Systems Engineering (DIMES), University of Calabria, Italy

Abstract

Missing values undermine the effectiveness of relational databases, and the problem becomes especially acute when databases evolve over time, since most existing data imputation techniques assume a static view of the data and require costly retraining whenever new (possibly incomplete) information arrives. We discuss the recent SENtence Transformer based Imputation approach (SENTI) for *Dynamic Data Imputation* (DDI), that is an incremental, training-free algorithm that exploits pre-trained sentence transformers and a vector database to perform semantic-aware imputation at each update [1]. Across five benchmark datasets and four missingness levels, SENTI consistently outperformed the state-of-the-art static baseline (adapted to the dynamic setting) in both effectiveness and efficiency.

Keywords

Dynamic Data Imputation, Sentence Transformers, Relational Databases

1. Introduction

Database systems are the backbone of modern digital infrastructure, providing the foundation for representing, querying, and efficiently managing information [2]. In real-world deployments, databases are incomplete: a fraction of tuples typically contains *nulls*, i.e., positions in which a value is missing. This issue can arise from various causes, including human or machine errors, malfunction of equipments, non-response in surveys, or data corruption during transmission or storage [3, 4]. The presence of nulls poses several issues regarding both the correctness of answers (recall that relational algebra and calculus do not allow nulls) [5, 6] and the quality and reliability in statistical analyses and machine learning models, introducing bias, reducing statistical power, and compromising the interpretability of results [7, 8, 9]. The need for efficient data imputation techniques is more evident in dynamic databases, where the contents evolve over time and missing values can significantly affect the integrity and reliability of real-time analytics [10, 11]. In such systems, incomplete or inconsistent records can harm decision-making, delay operations, and lead to erroneous interpretations.

Data Imputation (DI) addresses this challenge by estimating and replacing missing values (*nulls*) in databases with reliable constants. The process is foundational for maintaining data integrity and reducing the distortions introduced by incomplete information [12, 13, 14, 15, 16, 17]. Since the accuracy of imputed values propagates to downstream tasks (e.g., training machine-learning models, computing aggregates, performing statistical analyses, etc), DI matters most in fields that depend on reliable data if the imputed values are wrong, decisions and conclusions based on them will be wrong too.

Example 1. Consider the initial complete relation $r_0 = \{t_1, t_2, t_3, t_4\}$ shown at the top of Figure 1, containing personal records from different countries. At time step τ=1, the update $u_1 = \{t_5, t_6\}$ arrives, where t_6 contains nulls in the *Surname* and *City* columns. A semantically-aware imputation procedure should recognise that t_6 most closely resembles t_3 — both refer to people named 'Allisa' born on '10th Jan' in the US — and complete t_6 by borrowing the missing values from t_3, producing t_6^* with

AIxIA 2026 Discussion Papers, 24th International Conference of the Italian Association for Artificial Intelligence, Perugia, Italy, 2026

*Corresponding author.

g.alfano@dimes.unical.it (G. Alfano); greco@dimes.unical.it (S. Greco); lucio.lacava@dimes.unical.it (L. L. Cava); mahmood.tariq@dimes.unical.it (T. Mahmood); i.trubitsyna@dimes.unical.it (I. Trubitsyna)

published 2026-09-22

		Name	Surname	BirthDate	State	City	
r_0:	t_1	David	Rao	12th Feb	France	Paris	(a)
	t_2	Alex	Deere	Second Jan	Un. State	New York	
	t_3	Allisa	Jia	10th Jan	America	Boston	
	t_4	Alex	Deere	2nd Jan	Un. State	NYC	
u_1:	t_5	John	Iser	March 5	Austria	Wien	(b)
	t_6	Allisa	$\perp$	10th Jan	US	$\perp$	
u_1^*:	t_5	John	Iser	March 5	Austria	Wien	(c)
	t_6^*	Allisa	Jia	10th Jan	US	Boston	
u_2:	t_7	$\perp$	$\perp$	5 Mar	Austria	Wien	(d)
u_2^*:	t_7^*	John	Iser	5 Mar	Austria	Wien	(e)

Figure 1: An example of dynamic data imputation process from [1].

Surname = 'Jia' and *City* = 'Boston'. Similarly, at step τ=2, the update $u_2 = \{t_7\}$ arrives, with nulls in *Name* and *Surname*; the procedure now identifies t_5 as the closest existing tuple (both refer to people in "Wien, Austria, March 5") and fills the missing values accordingly, producing $t_7^* =$ ("John, Iser, 5 Mar, Austria, Wien"). Note also that semantically equivalent birthdates such as 'Second Jan' and '2nd Jan' (in r_0) must be recognised as equivalent — a step that requires reasoning over the tuple's semantic content rather than its surface form. □

The example highlights two facts. First, accurate imputation requires *semantic* reasoning over the entire tuple, not just column-wise statistics. Second, applying this process naïvely at every update step and re-training the full database from scratch is prohibitively expensive for real-world dynamic databases. Recently, both issues have been addressed by formalizing *Dynamic Data Imputation* (DDI), and by introducing the first incremental (evolving over time), training-free DDI procedure based on pre-trained sentence transformers [1]. It is worth noting that the dynamic nature of the problem and the training-free nature of SENTI are logically independent: a dynamic database could in principle be handled by periodic retraining of a learning-based imputation algorithm, but SENTI additionally avoids retraining by relying on frozen sentence-transformer embeddings indexed in a vector database. This decoupling is what enables a constant per-update cost w.r.t. the size of the database. This discussion paper summarizes the formulation, the algorithm, and the main empirical findings in [1].

2. Overview

Before recalling the *dynamic* data imputation problem, we introduce some notations.

Let $r = \{t_1, \ldots, t_n\}$ be a relation built over schema $R(A_1, \ldots, A_m)$. we denote with $r^* = \{t_1, \ldots, t_n\}$ the (complete) relation obtained from r by replacing null values $t_i[A_j]$ with elements of the active domain (i.e., $dom(A_j)$). Given a sentence transformer $\mathcal{S}$ and relations r and r^*, the *semantic similarity* among relations r and r^* is obtained as: $\text{sim}(r, r^*) = \frac{1}{n}\sum_{i=1}^{n} \frac{\mathbf{h}_i \cdot \mathbf{h}_i^*}{\|\mathbf{h}_i\| \, \|\mathbf{h}_i^*\|}$, with $\mathbf{h}_i = \mathcal{S}(t_i)$ and $\mathbf{h}_i^* = \mathcal{S}(t_i^*)$ the *sentence embeddings* of t_i and t_i^*. Finally, for any relation r and $k \in [0, 100]$, r^k denotes the relation obtained from r by replacing $k\%$ of its $n \times m$ values with $\perp$ uniformly at random.

(Dynamic) data imputation. A *data imputation function* is any $\gamma : [1, n] \times [1, m] \to \bigcup_{j=1}^{m} dom(A_j)$ with $\gamma(i, j) = t_i[A_j]$ when $t_i[A_j] \neq \perp$, and $\gamma(i, j) \in dom(A_j)$ otherwise. In the static setting (i.e., when the database is fixed and does not evolve over time), *Data Imputation* is the problem of finding, given a complete relation r, $k \in [0, 100]$, $\epsilon \in [0, 1]$, similarity function δ among tuples, and incomplete r^k derived from r, a function γ_θ such that $\delta\big(r, \gamma_\theta(r^k)\big) \geq \epsilon$.

In learning-based approaches, θ is fit from data and ϵ is used only as a feasibility threshold. Recently [1], the *Semantic-Aware Data Imputation problem* has been proposed as a generalization of the static data imputation problem in which δ is replaced by semantic similarity sim. Formally put, let r_0 is initially complete relation, $\langle u_1, \ldots, u_\ell \rangle$ a sequence of ℓ complete updates, $\langle u_1^k, \ldots, u_\ell^k \rangle$ the corresponding incomplete updates obtained by injecting nulls at rate $k \in [0, 100]$, δ a similarity function, and $\epsilon \in [0, 1]$. The *Dynamic Data Imputation* problem is to determine a function γ_θ such that $\delta\big(u_\tau,\ \gamma_\theta(r_\tau^k)\big) \geq \epsilon$ for every $\tau \in \{1, \ldots, \ell\}$, where $u_\tau^* = \gamma_\theta(r_\tau^k)$ is the imputed update at step τ.

An algorithm for solving DDI. The incremental algorithm SENTI has been proposed in [1] to solve the imputation problem in the dynamic setting. That is, at each time step, SENTI incrementally imputes the null values contained in the update relation u using: i) a pre-trained sentence transformer $\mathcal{S}$, ii) the current (complete) relation r, and iii) the vector database $E_r = \mathcal{S}(r)$ containing the embedding of r.

At each step, it maintains, alongside the current (imputed) relation r, a vector database $E_r = \{\mathcal{S}(t) \mid t \in r\}$ in which every tuple is represented by the dense embedding produced by a pre-trained sentence transformer $\mathcal{S}$. When a new (possibly incomplete) update u arrives, the algorithm (i) extends the vector database with the embeddings of tuples in u, obtaining $E_{r,u} = E_r \cup E_u$; (ii) for every incomplete tuple $t \in u$, retrieves the tuples of $r \cup u$ in increasing order of distance from $\mathcal{S}(t)$ (cosine similarity); and (iii) fills each null position of t with the corresponding value from the closest tuple that has a constant in that position. Referring to Example 1, the closest tuple to $\mathcal{S}(t_6)$ is t_3, so t_6's missing *Surname* and *City* are filled with 'Jia' and 'Boston' from t_3; at step 2, the closest tuple to $\mathcal{S}(t_7)$ is t_5, so t_7's missing *Name* and *Surname* are filled with 'John' and 'Iser' from t_5. The procedure runs in $O(|u| \cdot |r \cup u| \cdot \log |r \cup u|)$ time and uses the embedding model only at inference with no fine-tuning, no per-step retraining.

SENTI's donor-based mechanism assumes that, for any incomplete tuple t with $t[A] = \perp$, the value $t'[A]$ of its nearest neighbor t' in the embedding space is a plausible substitute for $t[A]$. In the extreme case — as in Example 1, where t_6 and t_3 arguably refer to the same real-world entity — the problem is similar to entity resolution/duplicate detection. In weaker but still legitimate cases, nearest neighbors denote distinct entities that nonetheless share the missing attribute (e.g., different individuals living in the same city agreeing on *State*).

Example 2. Consider the situation of Example 1, where the initial (complete) relation $r_0 = \{t_1, t_2, t_3, t_4\}$ is updated with $u_1 = \{t_5, t_6\}$ at time step $\tau = 1$. Algorithm is called with input $(r = r_0, E_r, u = u_1)$, where E_r represents the (already computed) vector database derived from r by means of a given pre-trained sentence-transformer $\mathcal{S}$. The algorithm first updates the vector DB E_r with the embeddings of each tuple in u, i.e., $E_{r,u} = E_r \cup (E_u = \{\mathcal{S}(t) \mid t \in u\})$ where e.g. $\mathcal{S}(t_6) : [-0,06,\ -0.003, \ldots]$. The first tuple of u containing a missing value is $t_6 = [$ "Allisa, $\perp$, 10 Jan, US, $\perp''$], as $t_6[2] = \perp$ and $t_6[5] = \perp$. The list of tuples $\langle t_3, t_4, t_5, t_2, t_1 \rangle$ in $r \cup u$ ordered w.r.t. the distance of $\mathcal{S}(t_i)$ from $\mathcal{S}(t_6)$ (for $i \in [1, 5]$); is computed. Then, the Algorithm replaces $t_6[1] = \perp$ with $t_3[1]$, and $t_6[5] = \perp$ with $t_3[5]$, As t_6 is the only incomplete tuple in u, the algorithm ends returning the complete relation u^*.

Assume now at time step $\tau = 2$ the update $u_2 = \{t_7\}$ is performed. Algorithm is called with input $(r = r_0 \cup u_1^*, E_r, u_2)$. The algorithm first updates the vector DB E_r with the embeddings of tuple $t_7 = [\perp, \perp,$ '5 Mar', 'Austria', 'Wien'], as shown above. Then, the tuple containing missing values is $t_7 = [\perp, \perp,$ 5 Mar, Austria, Wien], as $t_7[1] = \perp$ and $t_7[2] = \perp$. The list of tuples $\langle t_5, t_3, t_4, t_2, t_1, t_6 \rangle$ in r ordered w.r.t. the distance of $\mathcal{S}(t_i)$ from $\mathcal{S}(t_7)$ (for $i \in [1, 6]$). Next, the algorithm replaces $t_7[1] = \perp$ with $t_5[1]$, and $t_7[2] = \perp$ with $t_5[5]$. The complete update u_2^* is hence obtained. Clearly, after each imputation made at step τ, the underlying database is $r_\tau^* = r_0 \cup \bigcup_{i \in [1,\tau]} u_i^*$. □

3. Final Discussion

To ensure a fair comparison with the current SoTA data imputation approaches, SENTI has been evaluated on benchmark DI data [16] (see Table 1, column (Dataset)).

Table 1
An overview of the results presented in [1]. Imputation accuracy (%) and runtime (seconds), averaged over the 20 (#) update steps and 30 seeds. *Size* is the average number of tuples per update.

Dataset	Shape	Updates		Accuracy (%) w.r.t. % of nulls				Runtime (sec) w.r.t. % of nulls			
	(rows × cols)	#	Size	5%	10%	20%	40%	5%	10%	20%	40%
Adultsample	3016×14	20	101	95.19	94.90	94.15	92.93	1.13	1.84	2.84	3.97
Australian	690×15	20	23	92.53	92.24	91.46	89.80	0.29	0.43	0.63	0.97
Contraceptive	1473×10	20	49	96.06	95.76	94.92	93.78	0.49	0.76	1.25	1.83
Credit	653×16	20	22	91.99	91.83	91.12	89.80	0.29	0.42	0.61	0.88
IMDB	4529×11	20	151	89.10	88.95	88.59	88.39	1.67	2.79	4.15	5.79

In particular, each (complete) relation r has been corrupted by injecting in each column containing categorical values four different percentages $k\%$ of nulls with $k \in \{5, 10, 20, 40\}$, obtaining four different incomplete datasets r^k. The injected nulls follow the Missing Completely At Random (MCAR) mechanism [18], a general approach that does not assume any specific pattern in the missingness. The process has been validated on 30 runs, obtaining, for each complete relation r, a number of $4 \times 30 = 120$ distinct incomplete datasets.

Results. Table 1 reports SENTI's accuracy and runtime across five benchmarks at four null rates. Accuracy stays in 88–96% and *degrades gracefully*: from k=5% to k=40% the drop never exceeds 3 points. IMDB is the most stable (0.71 points), Australian the most sensitive (2.73 points), the smallest dataset, where a missing donor tuple has the largest relative impact. Contraceptive reaches the highest accuracy (96%) and IMDB the lowest, indicating that benchmarks with semantically homogeneous tuples benefit most from sentence-transformer embeddings. The running time is modest — under six seconds per update on IMDB at 40% nulls, under one second on smaller benchmarks — grows linearly with k, and stays constant across update steps, in stark contrast with the linearly growing cost of retraining-based baselines. The full comparison against IPM and its frozen variant IPM$_f$ is reported in [1]. Considering all datasets and seeds, SENTI consistently outperforms both IPM and IPM$_f$ in accuracy, with an average gain of 6.1% (resp., 6.9%) over IPM (resp., IPM$_f$). Regarding efficiency, across all datasets and seeds, the running time of IPM is 93.3 times that required by SENTI. Furthermore, SENTI is also 1.7 times faster than IPM$_f$.

Future Work. We have discussed SENTI, a transformer-based approach for imputing missing values in evolving relational databases. By embedding tuples into a high-dimensional semantic space via pre-trained sentence transformers and indexing them in a vector database, SENTI achieves high accuracy, stable imputation, and orders of magnitude faster than retraining-based imputers. Future directions include integrating SENTI with entity resolution pipelines on incomplete data, characterising downstream fairness implications of imputation choices on evolving relational databases, and quantifying the energy footprint of the procedure under a Green-AI evaluation framework.

Acknowledgements

We acknowledge support by the Italian Ministry of University and Research (MUR) PRIN 2022 grant 2022XERWK9 "S-PIC4CHU - Semantics-based Provenance, Integrity, and Curation for Consistent, High-quality, and Unbiased data science" – CUP: H53C24000990006.

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] G. Alfano, S. Greco, L. La Cava, T. Mahmood, I. Trubitsyna, Semantic-aware data imputation in dynamic relational databases via pre-trained language models, Information Fusion (2026) 104291. doi:10.1016/j.inffus.2026.104291.

[2] S. Abiteboul, R. Hull, V. Vianu, Foundations of Databases: The Logical Level, Addison-Wesley, 1995. URL: https://dl.acm.org/doi/10.5555/551350.

[3] T. Dasu, Data glitches: Monsters in your data, in: Handbook of data quality: Research and practice, Springer, 2013, pp. 163–178. doi:https://doi.org/10.1007/978-3-642-36257-6_8.

[4] D. A. Newman, Missing data: Five practical guidelines, Organizational research methods 17 (2014) 372–411. doi:https://doi.org/10.1177/1094428114548590.

[5] S. Greco, C. Molinaro, I. Trubitsyna, Approximation algorithms for querying incomplete databases, Inf. Syst. 86 (2019) 28–45. doi:https://doi.org/10.1016/j.is.2019.03.010.

[6] L. Libkin, L. Peterfreund, SQL nulls and two-valued logic, in: Proceedings of the PODS Conference, 2023, 2023, pp. 11–20. doi:https://doi.org/10.1145/3584372.3588661.

[7] E. G. Armitage, J. Godzien, V. Alonso-Herranz, Á. López-Gonzálvez, C. Barbas, Missing value imputation strategies for metabolomics data, Electrophoresis 36 (2015) 3050–3060. doi:https://doi.org/10.1002/elps.201500352.

[8] S. Jäger, A. Allhorn, F. Bießmann, A benchmark for data imputation methods, Frontiers in big Data 4 (2021) 693674. doi:10.3389/fdata.2021.693674.

[9] X. Zhen, M. Yu, X. He, S. Li, Multi-target regression via robust low-rank learning, IEEE transactions on pattern analysis and machine intelligence 40 (2017) 497–504. doi:10.1109/TPAMI.2017.2688363.

[10] H. Li, S. Han, M. Yang, J. Liu, J. Zhou, T. Zhang, C. L. P. Chen, Dynamic spatial-temporal imputation network with missing features for traffic data imputation, IEEE Internet of Things Journal (2025). doi:10.1109/JIOT.2025.3567697.

[11] D. Gkoulis, A. Tsadimas, G. Kousiouris, C. Bardaki, M. Nikolaidou, Exploring the performance of real-time data imputation to enhance fault tolerance on the edge: A study on environmental data, Simulation Modelling Practice and Theory 144 (2025) 103178. doi:https://doi.org/10.1016/j.simpat.2025.103178.

[12] N. Fouladgar, K. Främling, A novel lstm for multivariate time series with massive missingness, Sensors 20 (2020) 2832. doi:https://doi.org/10.3390/s20102832.

[13] M. Śmieja, M. Kołomycki, Ł. Struski, M. Juda, M. A. Figueiredo, Iterative imputation of missing data using auto-encoder dynamics, in: International Conference on Neural Information Processing, Springer, 2020, pp. 258–269. doi:10.1007/978-3-030-63836-8_22.

[14] L. Cappelletti, T. Fontana, G. W. Di Donato, L. Di Tucci, E. Casiraghi, G. Valentini, Complex data imputation by auto-encoders and convolutional neural networks—a case study on genome gap-filling, Computers 9 (2020) 37. doi:https://doi.org/10.3390/computers9020037.

[15] R. Shahbazian, S. Greco, Generative adversarial networks assist missing data imputation: A comprehensive survey & evaluation, IEEE Access (2023). doi:10.1109/ACCESS.2023.3306721.

[16] R. Cappuzzo, S. Thirumuruganathan, P. Papotti, Relational data imputation with graph neural networks, in: Proceedings 27th International Conference on Extending Database Technology, EDBT, Paestum, Italy, 2024. doi:DOI:10.48786/EDBT.2024.20.

[17] Y. Mei, S. Song, C. Fang, H. Yang, J. Fang, J. Long, Capturing semantics for imputation with pre-trained language models, in: Proc. of IEEE International Conference on Data Engineering (ICDE), 2021, pp. 61–72. doi:10.1109/ICDE51399.2021.00013.

[18] D. B. Rubin, Inference and missing data, Biometrika 63 (1976) 581–592. doi:https://doi.org/10.1093/biomet/63.3.581.