Tracking Uncertainty in Knowledge Graphs: A Kalman Filtering Approach

Alina Tkachenko, Alex Thomo, and Kevin Stanley

University of Victoria, Canada
{alinatkachenko, thomo, kevinstanley}@uvic.ca

Abstract. We introduce a method for learning knowledge graph embeddings as evolving Gaussian distributions, using Kalman filtering to support online updates and explicit uncertainty tracking. Our approach extends static models by replacing point embeddings (vectors) with probability distributions and applying online Kalman updates to both means and covariances. This allows embeddings to adapt continuously as new data arrives, without retraining. Experiments on standard benchmarks show up to 15% improvement in Mean Reciprocal Rank and 12% in Hit@10 over static baselines. Our method achieves scalable, real-time knowledge graph embedding with interpretable uncertainty.

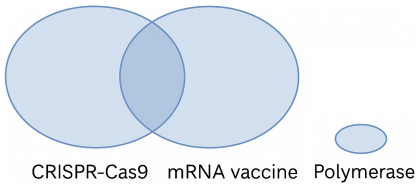
1 Introduction

Knowledge graphs (KGs) represent structured real-world information as triples (*head*, *relation*, *tail*), supporting reasoning and inference across domains. Their effectiveness hinges on Knowledge Graph Embeddings (KGEs), which map entities and relations into continuous vector spaces [1, 2, 13].

Standard embedding models treat all facts as equally reliable and assume fixed, uniform certainty across the graph. But in real-world KGs, uncertainty varies dramatically. Some entities appear in only a handful of triples, while others are richly connected. Some relations are semantically precise, others ambiguous or context-dependent. Ignoring this heterogeneity leads to brittle representations, especially in downstream tasks involving noisy, sparse, or evolving data.

KG2E [1, 7] addressed this by modeling embeddings as Gaussian distributions, with covariance capturing uncertainty. It showed that incorporating uncertainty improves robustness to sparse data and ambiguous relations. However, KG2E operates in a static setting: embeddings are learned in batches and cannot adapt to new triples without retraining.

To illustrate the importance of uncertainty-aware embeddings, consider a scientific knowledge graph where entities such as "CRISPR-Cas9" and "mRNA vaccine" are recent additions. These concepts are still evolving, and their meanings are context-dependent, with sparse and emerging connections in the graph. Representing such entities as fixed point vectors fails to capture their inherent ambiguity. Instead, embedding them as Gaussian distributions, with high covariance in less certain dimensions, better reflects this uncertainty.



The figure on the right illustrates three entities: "CRISPR-Cas9", "mRNA vaccine," and "Polymerase," each represented as a Gaussian. The larger ellipses for CRISPR-Cas9 and mRNA vaccine reflect greater uncertainty and show partial overlap, capturing both their emerging status and shared scientific context. In contrast, "Polymerase" appears as a tightly clustered ellipse, representing an older, well-established concept with low uncertainty. Without uncertainty modeling, these ellipses would collapse to single points at their centers, losing this richer view.

In this work, we extend uncertainty-aware KGEs into the online setting, where new facts continuously arrive. We model each embedding as a Gaussian distribution with evolving mean and covariance, using Kalman filtering [8] to update their estimates in real time. This approach brings three practical advantages over prior work such as KG2E, which models uncertainty but relies on batch gradient updates. First, uncertainty is explicitly tracked: entities or relations that appear rarely retain high variance, while those seen frequently converge to confident estimates. Second, embeddings are updated one triple at a time, making the method naturally suited for streaming or dynamic KGs. Third, we can model non-stationary distributions by tuning the process noise, allowing embeddings to drift as entities or relations evolve. In contrast to KG2E, which treats embeddings as static Gaussians optimized over a fixed training set, our method learns **evolving distributions** that sharpen or shift meaningfully as new data arrives.

Our contributions in this paper are as follows:

- We propose a **Kalman filter-based framework** for knowledge graph embeddings, representing entities and relations as Gaussians updated online.
- Our method tracks **uncertainty explicitly** and adjusts embeddings per triple without revisiting past data.
- It naturally supports **non-stationary embeddings** via process noise, allowing for semantic drift over time.
- We show **competitive performance** on standard benchmarks and improved robustness to sparse and evolving data.

2 Related Work

Knowledge graph embeddings (KGEs) learn vector (point) representations for entities and relations to support tasks such as link prediction and reasoning. Translational models pioneered this area: TransE models each relation as a translation between head and tail embeddings [2], and its variants TransH [14] and TransR [9] handle one-to-many mappings and relation-specific projections.

KG2E builds directly on TransE by embedding entities and relations as Gaussians, where covariance encodes per-dimension uncertainty [7]. This improves

robustness to sparsity but, like prior models, KG2E requires full retraining to incorporate new data.

Frameworks such as PyKEEN [1] provide modular pipelines for training and evaluating static KGE models, including a modern, well-engineered implementation of KG2E. This implementation represents the current state of the art in probabilistic modeling of entities and relations, which we use as our baseline.

Kalman filtering enables recursive state estimation under uncertainty by updating means and covariances incrementally [8]. Despite its wide success in control theory and online neural learning, it remains underexplored in KG embeddings. Building on the uncertainty-aware foundation of KG2E, our approach applies Kalman filtering to perform online updates of embeddings. This enables explicit uncertainty tracking, online adaptation without revisiting data, and support for non-stationary embeddings via process noise, advancing KGEs toward scalable, dynamic, and uncertainty-aware settings.

Finally, other probabilistic methods, such as UKGE [4] and BEUrRE [3], attach uncertainty to individual triples (facts) rather than to entities or relations. While effective for confidence-aware reasoning, they remain static and lack *per-entity and per-relation uncertainty*, and are thus orthogonal to our approach.

3 Background: Knowledge Graphs and Embeddings

Knowledge graphs (KGs) represent information as structured triples (h, r, t), where h is the head entity, r is the relation, and t is the tail entity. For instance, in the triple $(\text{Alice}, \text{parent\_of}, \text{Bob})$, Alice is the head, parent_of is the relation, and Bob is the tail. Relations in KGs can be either *symmetric*, holding in both directions (e.g., married_to), or *asymmetric*, where direction matters (e.g., parent_of). Asymmetric relations are more general since they capture directional dependencies. In this paper, we focus on the asymmetric case.

Knowledge Graph Embeddings (KGEs) map entities and relations in a knowledge graph (KG) to vectors in a continuous vector space, enabling machine learning algorithms to perform tasks such as link prediction, entity classification, and knowledge graph completion. The goal is to learn embeddings that preserve the structure of the KG, facilitating reasoning and inference. Various models achieve this in different ways.

For instance, translational distance models like TransE [2] represent relations as translations on entity embeddings. In TransE, the score function measures the distance between the head-relation vector $h + r$, "translation of head", and the tail vector t for valid triples, and a ranking loss ensures that valid triples are assigned better scores than "corrupted" triples by enforcing a margin between them. To obtain negative triples, corruption is performed by randomly replacing the head or tail entity in a triple to create invalid samples, which helps the model learn to differentiate between true and false triples.

He et al. [7] proposed KG2E to incorporate uncertainty into KG embeddings, addressing challenges associated with incomplete or noisy data. KG2E departs from classical KG embeddings by representing entities and relations as

multivariate Gaussian distributions, where the means capture central values and the variances or covariances reflect uncertainty. It also replaces vector distance with the KL-divergence between distributions. Specifically, for a triplet (h, r, t), KG2E compares the combined distribution of the head and the relation with the distribution of the tail, thus enabling uncertainty-aware modeling.

In the PyKEEN implementation of KG2E [1], optimization is performed with Adam to minimize the loss.

4 Background: Kalman Filter

The Kalman Filter is a mathematical framework for sequentially estimating the state of a system under uncertainty. It is widely used in data assimilation, control and state estimation scenarios to update parameter estimates dynamically as new data arrives. By iteratively balancing prior knowledge with noisy observations, the filter is ideal for applications where systems evolve over time. This adaptability makes it particularly useful for dynamically refining Knowledge Graph Embeddings (KGEs) as new triples are observed.

The state of a system at time n is modeled as a vector $\mathbf{x}_n \in \mathbb{R}^d$, evolving according to:

$$\mathbf{x}_n = F\mathbf{x}_{n-1} + \mathbf{w}_{n-1},$$

where $F \in \mathbb{R}^{d\times d}$ is the state transition matrix, and $\mathbf{w}_{n-1} \sim \mathcal{N}(0, Q)$ represents process noise with covariance $Q \in \mathbb{R}^{d\times d}$.

At each step, noisy observations $\mathbf{z}_n \in \mathbb{R}^m$ of the state are received:

$$\mathbf{z}_n = H\mathbf{x}_n + \mathbf{v}_n,$$

where $H \in \mathbb{R}^{m\times d}$ is the observation matrix, and $\mathbf{v}_n \sim \mathcal{N}(0, R)$ is the measurement noise with covariance $R \in \mathbb{R}^{m\times m}$.

The Kalman Filter operates in two phases:

- **Prediction Step:** The state $\mathbf{x}_n$ and its covariance P_n are predicted using the state transition model:

$$\hat{\mathbf{x}}_{n,\text{prior}} = F\hat{\mathbf{x}}_{n-1,\text{update}}, \quad P_{n,\text{prior}} = FP_{n-1,\text{update}}F^\top + Q.$$

- **Update Step:** The prediction is refined using the new observation $\mathbf{z}_n$:

$$K_n = P_{n,\text{prior}}H^\top(HP_{n,\text{prior}}H^\top + R)^{-1},$$

$$\hat{\mathbf{x}}_{n,\text{update}} = \hat{\mathbf{x}}_{n,\text{prior}} + K_n(\mathbf{z}_n - H\hat{\mathbf{x}}_{n,\text{prior}}),$$

$$P_{n,\text{update}} = (I - K_nH)P_{n,\text{prior}},$$

where K_n, the Kalman gain, determines the relative weight of the prior prediction and the new observation.

In the context of KGEs, we will present in the next section a principled way to dynamically update embeddings using the Kalman Filter as new triples arrive. Each entity and relation is modeled as a state vector $\mathbf{x}_n$, with uncertainty captured by a diagonal covariance matrix P_n, a simplification that improves efficiency while capturing per-dimension variance. Positive triples provide direct measurements that adjust embeddings to satisfy relational constraints, while negative triples act as contrastive signals to refine relationships.

5 Kalman Filter for KG Embeddings (KalmanKG2E)

In our proposed approach, we use Kalman filtering to learn embeddings for entities and relations in a Knowledge Graph as each triple arrives. The goal is to iteratively update both embeddings and associated covariance matrices for each entity and relation, maintaining a diagonal covariance structure for efficiency. We incorporate both positive and negative triples to learn from contrastive information.

Initialization

Each entity e_i and relation r_j in the Knowledge Graph has:

- an initial embedding vector, which can be initialized randomly or based on prior information, and
- a diagonal covariance matrix $P = \sigma^2 I$, where I is the identity matrix and σ^2 represents initial uncertainty.

Kalman Filter Update for Each Triple

For each triple (h, r, t), we update the embeddings of the head h, relation r, and tail t based on the type of triple (positive or negative).

Prediction Step In the prediction step, each embedding vector is predicted based on the previous state:

$$\hat{x}_{n,\text{prior}} = F\hat{x}_{n-1,\text{update}}$$

where F is the identity matrix, meaning that $\hat{x}_{n,\text{prior}} = \hat{x}_{n-1,\text{update}}$.

The covariance is updated to reflect process noise:

$$P_{n,\text{prior}} = P_{n-1,\text{update}} + Q$$

where Q is a diagonal matrix with small values representing process noise.

The identity matrix F is used in the prediction step because, in the context of learning embeddings, there is no external force or factor causing the embeddings to change independently between updates. The embeddings are only adjusted based on new triple information rather than an intrinsic evolution over time.

This allows the prior estimate $\hat{x}_{n,\text{prior}}$ to directly inherit the previous updated state $\hat{x}_{n-1,\text{update}}$.

Additionally, adding the process noise matrix Q to the covariance $P_{n,\text{prior}}$ introduces a modest level of uncertainty in each embedding dimension, enabling the model to remain flexible and responsive as subsequent measurements (triples) arrive, without rigidly fixing the embeddings in place.

Measurement Model for Positive and Negative Triples For each triple (h, r, t):

- **Positive Triple:** For a positive triple (e.g., (h, r, t)), the measurement reflects the expected relationship:

$$z_{h,\text{measure}} = t - r, \quad z_{t,\text{measure}} = h + r, \quad z_{r,\text{measure}} = t - h$$

  and the measurement noise covariance R_{positive} is used.
- **Negative Triple:** For a corrupted or negative triple (e.g., (h, r, t') where $t' \neq t$), the measurement encourages the model to avoid $h + r \approx t'$. We do this by setting:

$$z_{t',\text{measure}} = t' + \text{offset}$$

  where *offset* is a small vector that pushes t' away from $h + r$. The measurement noise covariance is set higher for negative triples as $R_{\text{negative}} = \alpha \cdot R_{\text{positive}}$, where $\alpha > 1$.

The measurement model differentiates between positive and negative triples to reinforce both valid and invalid relationships in the Knowledge Graph. For a positive triple (h, r, t), the measurements $z_{h,\text{measure}} = t - r$, $z_{t,\text{measure}} = h + r$, and $z_{r,\text{measure}} = t - h$ directly align with the TransE translation principle, so that the embeddings adjust to satisfy $h + r \approx t$. We also apply a noise covariance R_{positive} to allow some flexibility but still place high trust in these measurements.

Conversely, for a negative triple (h, r, t') (where $t' \neq t$), a shifted measurement $z_{t',\text{measure}} = t' + \text{offset}$ is used to create separation between $h + r$ and t', pushing the embeddings to avoid this incorrect association.

For negative triples, only $z_{t',\text{measure}}$ (e.g., $z_{t',\text{measure}} = t' + \text{offset}$) is used, without defining $z_{h,\text{measure}}$ or $z_{r,\text{measure}}$. This way we emphasize the role of negative triples as "soft" constraints, simply indicating that $h + r \approx t'$ is incorrect. Focusing only on $z_{t',\text{measure}}$, the model is encouraged to push $h + r$ away from t' without enforcing specific adjustments for h and r. This approach keeps the negative triples' influence simple and contrastive, guiding the embeddings to avoid incorrect associations without heavily altering h and r based on each negative example.

The noise covariance R_{negative} is set higher, with $\alpha > 1$, to downweight the influence of negative triples compared to positive ones. This balance prevents negative triples from overly distorting the embeddings while still providing contrastive information that sharpens the model's understanding of true relationships.

The offset in the measurement model for negative triples introduces a controlled separation between $h+r$ and the corrupted tail t', helping the model distinguish valid triples from incorrect ones. The choice of offset size is important–it should be large enough to discourage alignment between $h + r$ and t' without forcing an unrealistic distance that would interfere with the general embedding structure. We set the offset as a random vector drawn from a normal distribution with a small standard deviation relative to the embedding size. This adds variation to each corrupted example without making the model overly sensitive to negative triples.

Kalman Gain and Update Equations For each entity or relation embedding, the Kalman filter applies the following update steps:

Kalman Gain:

$$K_n = P_{n,\text{prior}}(P_{n,\text{prior}} + R)^{-1}$$

where $R = R_{\text{positive}}$ for positive triples and $R = R_{\text{negative}}$ for negative triples.

State Update:

$$\begin{aligned}
\hat{x}_{n,h,\text{update}} &= \hat{x}_{n,h,\text{prior}} + K_{n,h}(z_{h,\text{measure}} - \hat{x}_{n,h,\text{prior}}) \\
\hat{x}_{n,t,\text{update}} &= \hat{x}_{n,t,\text{prior}} + K_{n,t}(z_{t,\text{measure}} - \hat{x}_{n,t,\text{prior}}) \\
\hat{x}_{n,r,\text{update}} &= \hat{x}_{n,r,\text{prior}} + K_{n,r}(z_{r,\text{measure}} - \hat{x}_{n,r,\text{prior}})
\end{aligned}$$

Covariance Update:

$$\begin{aligned}
P_{n,h,\text{update}} &= (I - K_{n,h})P_{n,h,\text{prior}} \\
P_{n,t,\text{update}} &= (I - K_{n,t})P_{n,t,\text{prior}} \\
P_{n,r,\text{update}} &= (I - K_{n,r})P_{n,r,\text{prior}}
\end{aligned}$$

The Kalman gain K_n dynamically adjusts the weight given to the prior estimate versus the new measurement, based on their respective uncertainties. Since we assume an identity mapping from state to observation (i.e., $H = I$), the measurements directly inform the embeddings without transformation.

5.1 Example

Assume a Knowledge Graph with a positive triple $(h, r, t) = (\text{Paris}, \text{capital\_of}, \text{France})$ and a negative triple $(h, r, t') = (\text{Paris}, \text{capital\_of}, \text{Germany})$. We initialize the embeddings for each entity and relation randomly and define their initial covariance matrices as follows.

– **Embeddings:**

$$\hat{x}_{\text{Paris}} = \begin{bmatrix} 0.5 \\ 0.3 \end{bmatrix}, \quad \hat{x}_{\text{capital\_of}} = \begin{bmatrix} 0.2 \\ 0.1 \end{bmatrix},$$

$$\hat{x}_{\text{France}} = \begin{bmatrix} 1.0 \\ 0.8 \end{bmatrix}, \quad \hat{x}_{\text{Germany}} = \begin{bmatrix} 0.7 \\ 0.9 \end{bmatrix}$$

– **Initial Covariance Matrices:** Each embedding is associated with an initial diagonal covariance matrix to represent uncertainty in each dimension.

$$P_{\text{Paris}} = \begin{bmatrix} 0.05 & 0 \\ 0 & 0.05 \end{bmatrix}, \quad P_{\text{capital\_of}} = \begin{bmatrix} 0.05 & 0 \\ 0 & 0.05 \end{bmatrix},$$
$$P_{\text{France}} = \begin{bmatrix} 0.05 & 0 \\ 0 & 0.05 \end{bmatrix}, \quad P_{\text{Germany}} = \begin{bmatrix} 0.05 & 0 \\ 0 & 0.05 \end{bmatrix}$$

Kalman Filter Update for Positive Triple

For positive triple $(\text{Paris}, \text{capital\_of}, \text{France})$, we compute the measurements as:

$$z_{\text{Paris, measure}} = \hat{x}_{\text{France}} - \hat{x}_{\text{capital\_of}} = \begin{bmatrix} 0.8 \\ 0.7 \end{bmatrix},$$
$$z_{\text{France, measure}} = \hat{x}_{\text{Paris}} + \hat{x}_{\text{capital\_of}} = \begin{bmatrix} 0.7 \\ 0.4 \end{bmatrix}$$

and use a measurement noise covariance

$$R_{\text{positive}} = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$$

for both Paris and France.

Kalman Gain Calculation

For Paris, the Kalman gain K_{Paris} is calculated as:

$$K_{\text{Paris}} = P_{\text{Paris, prior}}(P_{\text{Paris, prior}} + R_{\text{positive}})^{-1}$$
$$= \begin{bmatrix} 0.05 & 0 \\ 0 & 0.05 \end{bmatrix} \begin{bmatrix} 0.15 & 0 \\ 0 & 0.15 \end{bmatrix}^{-1} = \begin{bmatrix} 0.333 & 0 \\ 0 & 0.333 \end{bmatrix}$$

State Update for Paris

The updated embedding for Paris is:

$$\hat{x}_{\text{Paris, update}} = \hat{x}_{\text{Paris, prior}} + K_{\text{Paris}}(z_{\text{Paris, measure}} - \hat{x}_{\text{Paris, prior}}) =$$
$$\begin{bmatrix} 0.5 \\ 0.3 \end{bmatrix} + \begin{bmatrix} 0.333 & 0 \\ 0 & 0.333 \end{bmatrix} \left(\begin{bmatrix} 0.8 \\ 0.7 \end{bmatrix} - \begin{bmatrix} 0.5 \\ 0.3 \end{bmatrix} \right) = \begin{bmatrix} 0.6 \\ 0.433 \end{bmatrix}$$

Covariance Update for Paris

The updated covariance for Paris is:

$$P_{\text{Paris, update}} = (I - K_{\text{Paris}})P_{\text{Paris, prior}} =$$
$$\left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 0.333 & 0 \\ 0 & 0.333 \end{bmatrix} \right) \begin{bmatrix} 0.05 & 0 \\ 0 & 0.05 \end{bmatrix} = \begin{bmatrix} 0.033 & 0 \\ 0 & 0.033 \end{bmatrix}$$

This completes the Kalman filter update for Paris. A similar process would apply for France and capital_of.

Kalman Filter Update for Negative Triple

For the negative triple $(\text{Paris}, \text{capital\_of}, \text{Germany})$, we aim to discourage the alignment $h + r \approx t'$. To achieve this, we introduce an offset to the measurement for Germany, pushing it away from the predicted position of Paris + capital_of. We set:

$$z_{\text{Germany, measure}} = \hat{x}_{\text{Germany}} + \text{offset} = \begin{bmatrix} 0.7 \\ 0.9 \end{bmatrix} + \begin{bmatrix} 0.1 \\ -0.1 \end{bmatrix} = \begin{bmatrix} 0.8 \\ 0.8 \end{bmatrix}$$

where the offset is chosen to shift Germany's embedding slightly in a different direction. For negative triples, we use a higher measurement noise covariance $R_{\text{negative}} = \alpha \cdot R_{\text{positive}} = \begin{bmatrix} 0.3 & 0 \\ 0 & 0.3 \end{bmatrix}$, with $\alpha = 3$.

Kalman Gain Calculation for Germany

For Germany, the Kalman gain K_{Germany} is calculated as:

$$K_{\text{Germany}} = P_{\text{Germany, prior}}(P_{\text{Germany, prior}} + R_{\text{negative}})^{-1} =$$

$$\begin{bmatrix} 0.05 & 0 \\ 0 & 0.05 \end{bmatrix} \begin{bmatrix} 0.35 & 0 \\ 0 & 0.35 \end{bmatrix}^{-1} = \begin{bmatrix} 0.143 & 0 \\ 0 & 0.143 \end{bmatrix}$$

State Update for Germany

The updated embedding for Germany is:

$$\hat{x}_{\text{Germany, update}} =$$

$$\hat{x}_{\text{Germany, prior}} + K_{\text{Germany}}(z_{\text{Germany, measure}} - \hat{x}_{\text{Germany, prior}}) =$$

$$\begin{bmatrix} 0.7 \\ 0.9 \end{bmatrix} + \begin{bmatrix} 0.143 & 0 \\ 0 & 0.143 \end{bmatrix} \left(\begin{bmatrix} 0.8 \\ 0.8 \end{bmatrix} - \begin{bmatrix} 0.7 \\ 0.9 \end{bmatrix} \right) = \begin{bmatrix} 0.714 \\ 0.886 \end{bmatrix}$$

Covariance Update for Germany

The updated covariance for Germany is:

$$P_{\text{Germany, update}} = (I - K_{\text{Germany}})P_{\text{Germany, prior}} =$$

$$\left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 0.143 & 0 \\ 0 & 0.143 \end{bmatrix} \right) \begin{bmatrix} 0.05 & 0 \\ 0 & 0.05 \end{bmatrix} = \begin{bmatrix} 0.043 & 0 \\ 0 & 0.043 \end{bmatrix}$$

6 Evaluation

We experiment on six widely used benchmarks,WN18RR [5], FB15k237 [12], CoDExMedium [11], DB100K [6], CoDExLarge [11], and YAGO310 [10], which vary in domain, scale, and complexity. This diversity ensures a thorough evaluation of performance and generalizability. These datasets and their statistics are shown in Table 1.

We evaluate performance using two standard metrics: **Hit@10** and **Mean Reciprocal Rank (MRR)**. In the link prediction task, each test query corresponds to predicting a missing entity in a triple (e.g., $(h, r, ?)$). To answer such queries in our probabilistic framework, we construct a Gaussian for the missing side by combining the observed entity and relation distributions (e.g., $h + r$ for tail prediction) and compare it with each candidate entity distribution using KL-divergence. Candidates are ranked so that entities with lower KL divergence to the predicted Gaussian receive higher scores. Hit@10 measures the proportion of queries where the true entity is among the top 10 ranked candidates, reflecting retrieval effectiveness. MRR captures the average inverse rank of the correct entity across all queries, rewarding cases where the true answer is ranked near the top. Results for these metrics are shown in Figures 1 and 2.

Dataset	Entities	Relations	Train	Test
WN18RR	40,943	11	86,835	3,134
FB15k237	14,541	237	272,115	20,466
CoDExMedium	17,050	51	185,584	10,311
DB100K	99,604	470	597,572	50,000
CoDExLarge	77,951	69	551,193	30,622
YAGO310	123,182	37	1,073,112	4,076

Table 1. Dataset Statistics.

6.1 Discussion

The results across all datasets consistently underscore the effectiveness of introducing Kalman filtering into knowledge graph embeddings. **KalmanKG2E** achieves clear performance gains over the PyKeen SOTA implementation of KG2E in both Hit@10 and MRR across all benchmarks (Figures 1 and 2). These improvements appear early in training, persist throughout, and reach up to 15% absolute in Hit@10 and 12% in MRR on challenging datasets such as CoDExLarge and YAGO310. Beyond raw scores, this demonstrates that Kalman updates allow the model to assimilate new facts rapidly while maintaining coherent, uncertainty-aware representations, a key advantage in sparse or evolving graphs. Overall, the experiments show that Kalman filtering provides a flexible framework for online embedding refinement. It strengthens KG2E by enabling explicit uncertainty tracking, faster convergence, and resilience to non-stationary graph structure, all without retraining, making it a practical choice for real-time knowledge graph applications.

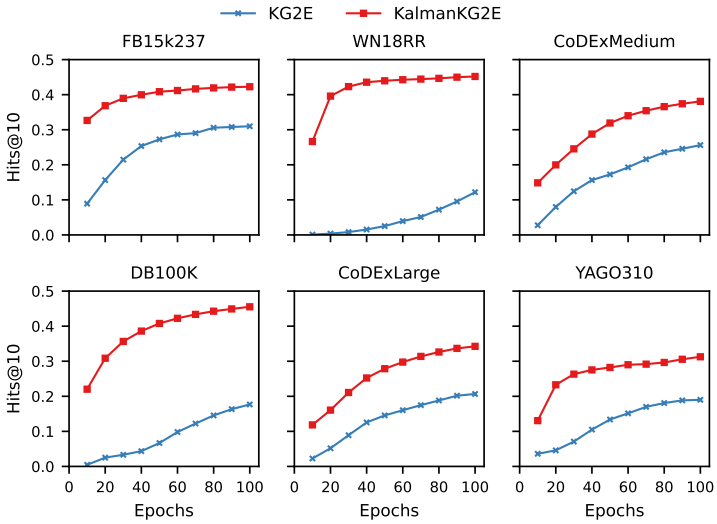


Fig. 1. Hit@10 over training epochs for KalmanKG2E (KG2E + Kalman Filter).

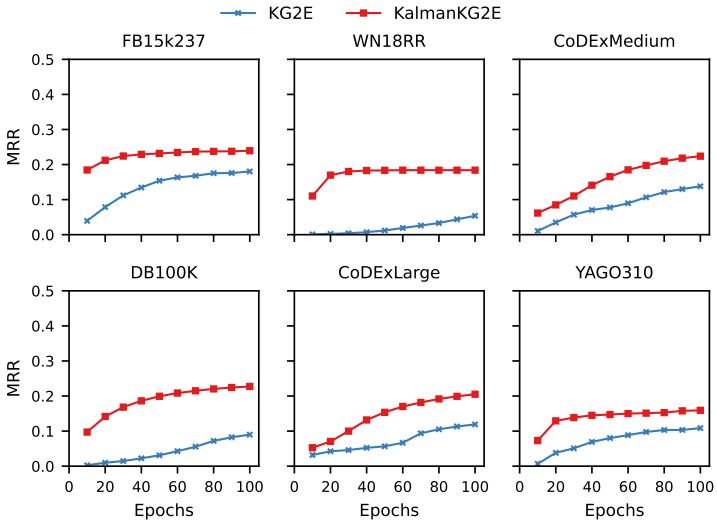


Fig. 2. MRR over training epochs for KalmanKG2E.

7 Conclusion

We presented a general framework for online knowledge graph embedding that integrates Kalman filtering with Gaussian representations. Our method updates the mean and covariance of entity and relation embeddings incrementally as each new triple arrives, allowing continuous adaptation to new facts, explicit uncertainty tracking, and support for evolving semantics. Experiments on six benchmark datasets show consistent gains in Hit@10 and MRR, with faster convergence and stronger robustness under sparsity and dynamic relational structures. These results highlight the value of probabilistic online updates in knowledge graph embedding. Future work includes extending Kalman filtering to other embedding families and incorporating task-specific priors for finer control over uncertainty and drift.

References

1. Ali, M., Berrendorf, M., Hoyt, C.T., Vermue, L., Sharifzadeh, S., Tresp, V., Lehmann, J.: PyKEEN 1.0: A Python Library for Training and Evaluating Knowledge Graph Embeddings. Journal of Machine Learning Research **22**(82), 1–6 (2021). URL http://jmlr.org/papers/v22/20-825.html
2. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: NIPS (2013)
3. Chen, X., Boratko, M., Chen, M., Dasgupta, S.S., Li, X.L., McCallum, A.: Probabilistic box embeddings for uncertain knowledge graph reasoning. arXiv preprint arXiv:2104.04597 (2021)
4. Chen, X., Chen, M., Shi, W., Sun, Y., Zaniolo, C.: Embedding uncertain knowledge graphs. In: AAAI, vol. 33, pp. 3363–3370 (2019)
5. Dettmers, T., Minervini, P., Stenetorp, P., Riedel, S.: Convolutional 2d knowledge graph embeddings. In: AAAI (2018)
6. Ding, B., Wang, Q., Wang, B., Guo, L.: Improving knowledge graph embedding using simple constraints. In: ACL, pp. 110–121 (2018)
7. He, S., Liu, K., Ji, G., Zhao, J.: Learning to represent knowledge graphs with gaussian embedding. In: CIKM, pp. 623–632 (2015)
8. Khodarahmi, M., Maihami, V.: A review on kalman filter models. Archives of Computational Methods in Engineering **30**(1), 727–747 (2023)
9. Lin, Y., Liu, Z., Sun, M., Liu, Y., Zhu, X.: Learning entity and relation embeddings for knowledge graph completion. In: AAAI, pp. 2181–2187. AAAI Press (2015)
10. Mahdisoltani, F., Biega, J., Suchanek, F.M.: Yago3: A knowledge base from multilingual wikipedias. In: CIDR (2013)
11. Safavi, T., Koutra, D.: Codex: A comprehensive knowledge graph completion benchmark. arXiv preprint arXiv:2009.07810 (2020)
12. Toutanova, K., Chen, D.: Observed versus latent features for knowledge base and text inference. In: Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality, pp. 57–66. ACL (2015)
13. Trouillon, T., Welbl, J., Riedel, S., Gaussier, E., Bouchard, G.: Complex embeddings for simple link prediction. In: ICML (2016)
14. Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge graph embedding by translating on hyperplanes. In: AAAI, pp. 1112–1119. AAAI Press (2014)