CYCLE: Cross-Year Contrastive Learning in Entity-Linking

Pengyu Zhang
University of Amsterdam
Amsterdam, The Netherlands
p.zhang@uva.nl

Congfeng Cao
University of Amsterdam
Amsterdam, The Netherlands

Klim Zaporojets
Aarhus University
Aarhus, Denmark

Paul Groth
University of Amsterdam
Amsterdam, The Netherlands

Abstract

Knowledge graphs constantly evolve with new entities emerging, existing definitions being revised, and entity relationships changing. These changes lead to temporal degradation in entity linking models, characterized as a decline in model performance over time. To address this issue, we propose leveraging graph relationships to aggregate information from neighboring entities across different time periods. This approach enhances the ability to distinguish similar entities over time, thereby minimizing the impact of temporal degradation. We introduce **CYCLE**: **C**ross-**Y**ear **C**ontrastive **L**earning for **E**ntity-Linking. This model employs a novel graph contrastive learning method to tackle temporal performance degradation in entity linking tasks. Our contrastive learning method treats newly added graph relationships as *positive* samples and newly removed ones as *negative* samples. This approach helps our model effectively prevent temporal degradation, achieving a 13.90% performance improvement over the state-of-the-art from 2023 when the time gap is one year, and a 17.79% improvement as the gap expands to three years. Further analysis shows that CYCLE is particularly robust for low-degree entities, which are less resistant to temporal degradation due to their sparse connectivity, making them particularly suitable for our method. The code and data are made available at https://github.com/pengyu-zhang/CYCLE-Cross-Year-Contrastive-Learning-in-Entity-Linking.

CCS Concepts

• **Computing methodologies** → **Temporal reasoning**.

Keywords

Knowledge Graph; Entity Linking; Knowledge Acquisition; Contrastive Learning

ACM Reference Format:
Pengyu Zhang, Congfeng Cao, Klim Zaporojets, and Paul Groth. 2024. CYCLE: Cross-Year Contrastive Learning in Entity-Linking. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management (CIKM '24), October 21–25, 2024, Boise, ID, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3627673.3679702

CIKM '24, October 21–25, 2024, Boise, ID, USA

ACM ISBN 979-8-4007-0436-9/24/10
https://doi.org/10.1145/3627673.3679702

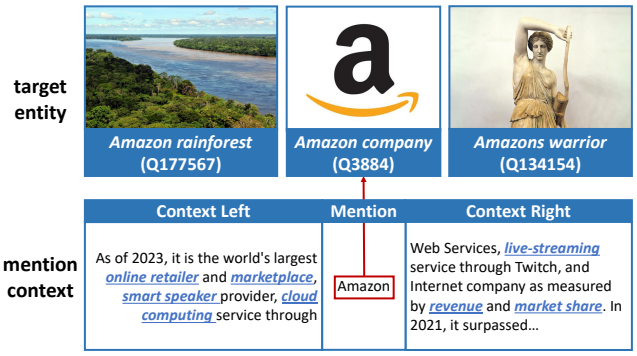

Figure 1: Entity linking (EL) connects text mentions to specific entities in a KG. In the example of this figure, the mention 'Amazon' could refer to *Amazon rainforest*, *Amazon company*, or mythical *Amazons warriors* entities. EL disambiguates this mention considering the surrounding *mention context* to pinpoint the correct entity: *Amazon company*.

1 Introduction

Knowledge graphs (KGs) are multi-relational graphs representing a wide range of real-world entities and knowledge structured as facts [5]. In KGs, facts are represented as triples, denoted as *(head entity, relation, tail entity)*. KGs like ICEWS [3], GDELT [10], and YAGO3 [15] not only contain numerous entities and their relations but also incorporate timestamps, allowing to track the evolution of the knowledge in these KGs. Each of these timestamps is part of a quadruplet *(head entity, relation, tail entity, timestamp)*, reflecting the point in time a particular relation between head and tail entities was created. Such temporal features enable KG users to track historical data trends, understand entity behavior over time, and predict future events or facts, highly relevant in domains such as medical and risk analysis systems [6], question-answering systems [21], and recommendation systems [2]. However, the continuous emergence of new entities and changes to existing ones, poses challenges in adapting entity representations, which in turn affects the performance on tasks such as Entity Linking (EL) [17].

EL involves mapping mentions in text to entities in a KG. In the example in Figure 1, EL maps the mention 'Amazon' in the text, with its correct entity *Amazon (company)* in the KG. The term 'Amazon' could refer to *Amazon (company)*, known for e-commerce, or *Amazon rainforest*. The correct entity is decided by analyzing the content of the context. If the context includes *cloud computing*,

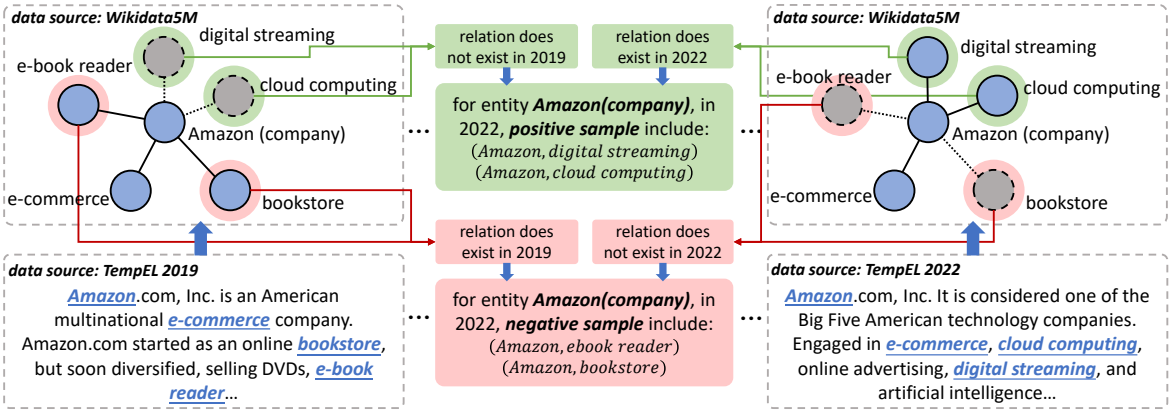

Figure 2: This is an example of learning the embedding of the entity *Amazon (company)* in 2022. Gray nodes represent entities that do not exist at a specific point in time. A (green) or (red) circle represents the use of a node as a *positive* or *negative* sample, respectively. In 2019, *Amazon (company)* had three neighbors: *e-commerce*, *e-book reader*, and *bookstore*. By 2022, while *e-commerce* continued as a neighbor, *e-book reader* and *bookstore* (negative samples) were replaced by *digital streaming* and *cloud computing* (positive samples). This shift indicates a change in *Amazon (company)*'s focus.

it likely refers to *Amazon (company)*. Beyond the challenge of disambiguating the correct entity given the mention and its context, this task becomes even more difficult in the face of the constant addition of new entities in a KG and the evolving meanings of existing entities (continual entities). This change leads to temporal degradation - a decline in model performance as the KG moves from the state where the EL model was initially trained.

In this paper, we develop a new model for EL that deals with temporal change. For instance, as shown in Figure 2, in 2019, the primary sources of income for *Amazon (company)* were *e-book reader* and *bookstore*. By 2022, however, *Amazon (company)* had established new connections, and recently added entities like *digital streaming* and *cloud computing* had become major contributors to its income. Concentrating on these new connections allows for a deeper understanding of the current state of the entity and its context. Although these changes might appear minor, they substantially affect the model's ability to represent entities accurately.

To address this challenge, a benchmark introduced by [8] leverages the differences between consecutive snapshots of Wikipedia and Wikidata to provide a platform for testing and training language models, enabling them to adapt to and update continually changing knowledge information. Furthermore, recent work has focused on reducing bias through graph contrastive learning [27]. In addition, recent studies like [38] on self-supervised biomedical EL and [26] on multimodal EL with contrastive learning have advanced the field by improving accuracy. However, existing methods do not leverage the temporal evolution of structured relations between entities in a KG across different years. To tackle this research gap, we introduce an expanded version of the TempEL [36] dataset named GCL-TempEL (see Section 4). The original TempEL dataset consists of 10 yearly snapshots, evenly distributed, from English Wikipedia entities, spanning from January 1, 2013, to January 1, 2022. Building upon this, we have incorporated the relationships between entities for each year from the KG introduced in the Wikidata5M [28] dataset, and the changes in these relationships over time.

We hypothesize that the evolution of KG relationships across different years in our newly introduced dataset, can provide crucial information about the changes in the entities (see Figure 2). To support this, we introduce **CYCLE**: **C**ross-**Y**ear **C**ontrastive **L**earning in **E**ntity-Linking. **CYCLE** is a novel approach to solve EL task in temporally evolving setting based on graph contrastive learning, leveraging the features of temporal data to construct a cross-year contrastive mechanism. Doing so ensures that similar entity representations remain distinct over time. Our contrastive learning approach uses KG relationships to obtain structurally enhanced entity representations. We define such *positive* and *negative* samples as follows:

1 *Positive samples*: sampled from a pool of newly added relationships between the target entity and its neighbors.
2 *Negative samples*: sampled from a pool of newly deleted relationships between the target entity and its neighbors.

Our experimental results demonstrate that our approach can be particularly advantageous in updating representations of less frequent, long tail, entities. Such entities have lower degree connectivity in a KG and, as a result, are inherently more vulnerable to the effects of temporal changes in their neighbors. Concretely, a change in the meaning of even one neighboring node can significantly impact the low-degree node's representation. This phenomenon has been also observed in other large-scale KGs such as ICEWS and GDELT [25, 30, 31].

Our contributions are summarized as follows:

- A dataset, GCL-TempEL, that incorporate cross-year KG temporal tracking of entity changes. Concretely, we define *positive* and *negative* samples with respect to a specific temporal snapshot.
- CYCLE: a novel model employing graph contrastive learning to mitigate temporal degradation in EL. CYCLE enhances

the temporal stability of low-degree node representations, which are more vulnerable to semantic changes due to sparse connections.
- Experiments across three EL datasets that demonstrate substantial improvements in model performance, particularly for low-degree nodes. The experiments highlight CYCLE's effectiveness in addressing structural vulnerabilities in KGs.

2 Related Work

2.1 Entity Linking

Generally, the Entity Linking (EL) task is categorized into three main phases: mention detection, candidate generation, and candidate ranking. Recent studies have developed end-to-end models that integrate all three phases into a single process [4, 16, 19]. Specifically, the CHOLAN model [19] proposed two transformer-based models integrated sequentially to tackle the EL task. The first transformer identifies entity mentions in the text, while the second assigns each mention to a predefined candidate, enhanced by contextual data from the sentence and Wikipedia. As the field evolved, addressing the linking of mentions to previously unseen entities - a scenario termed zero-shot EL - remained a significant challenge [20]. To tackle this, [32] introduce BLINK, a highly effective two-stage BERT-based architecture for zero-shot entity linking. This model and others [1, 20, 33] rely on traditional candidate retrieval methods and employ a cross-encoder for candidate ranking. Building on the BLINK model, KG-ZESHEL [20] aims to combine graph vectors with textual content to address the zero-shot problem. Their method enhances the model's ability to clarify ambiguities and improve EL accuracy.

However, the above work did not capture the impact of changes in entity relationships on future predictions when faced with dynamic KGs. To tackle this issue, our study introduces a cross-year contrastive mechanism to capture KG relationship changes across the years, thereby improving the performance on EL task in a temporally evolving setting.

2.2 Temporal Knowledge Graphs

Temporal Knowledge Graphs (TKGs), capturing the dynamic evolution of entities and their relationships over time, are gaining increased attention [25]. While most of the existing graph datasets are designed for static graphs, TKGs stand out for their ability to chronicle the continuous evolution of both entities and relations.

The application of TKGs spans various sectors, each benefiting from its capacity to track changes over time. For e-commerce, TKGs enhance understanding of consumer behavior patterns over time [39]. Similarly, the Internet of Things (IoT) provides a dynamic framework for interpreting evolving data from interconnected devices [12]. Healthcare applications of TKGs are particularly noteworthy, as they aid in tracking the progression of diseases and patient health trends [6]. In industrial settings, TKGs play a pivotal role in monitoring and predicting machinery's lifecycle and maintenance needs. Recent advances in large language models (LLMs) have further expanded the potential of TKGs, particularly in forecasting applications [13]. These architectures offer new ways to comprehend structured temporal data, potentially revolutionizing traditional embedding-based and rule-based applications using TKGs. Moreover, integrating temporal information into KG embedding has significantly improved model performance, underscoring the importance of time-aware approaches in knowledge representation [7, 11].

However, these studies overlook the challenges of low-degree nodes with very sparse connectivity, which can significantly impact the accuracy and robustness of temporal predictions. To address this, our model employs graph contrastive learning on positive and negative samples (see Figure 2) to enhance the neighbor information for low-degree nodes, thereby improving their representation in the graph.

2.3 Graph Contrastive Learning

Recent Graph Contrastive Learning (GCL) advancements highlight its growing significance in graph representation learning. PyGCL [40] emphasizes the importance of design elements like augmentation functions and contrasting modes. GraphCL [34] stands out for its ability to learn robust representations from unlabeled graphs, though its effectiveness relies on specific data augmentation strategies. POT-GCL [35] addresses the need to maximize similarity between positive node pairs and minimize it for negative pairs, while recognizing unresolved issues due to complex graph structures. In contrast, SGCL [24] demonstrates the low impact of negative samples for achieving top performance. Finally, EdgePruner [9] exposes GCL's vulnerability to poisoning attacks, suggesting a need for better defense mechanisms in graph learning models. These studies mark a significant evolution in GCL, indicating its potential and challenges in graph representation learning.

Our work expands the application of contrastive learning techniques within KGs to include temporal evolution. Specifically, we have developed a new method that considers the impact of both newly added and removed node relationships at each timestamp.

3 Task Formulation and Definition

Entity Linking (EL). The EL task takes a given text document $\mathbf{D}$ as input. This document is represented as a list of tokens $[w_1, \ldots, w_r]$, where r indicates the length of the document. Each document contains a list of entity mentions $\mathbf{M_D} = [m_1, \ldots, m_n]$, where each mention m_i corresponds to a span of continuous tokens in $\mathbf{D}$, represented as $m_i = \mathbf{D}[x, y]$. An EL model subsequently yields a list of mention-entity pairs $\{(m_i, e_i)\}_{i\in[1,n]}$. Every entity e_i is mapped to the corresponding entity in a Knowledge Base, such as Wikipedia. It is assumed that both the title and description of these entities are available, a standard premise in the EL task [14].

Graph. A graph is defined as $G = (V, E)$, where V is the set of N nodes (i.e., entities) $\{v_1, v_2, \cdots, v_N\}$. E is the set of M edges (i.e., relationships) represented as $\{e_1, e_2, \cdots, e_M\}$, where each e_i is a pair of nodes from V, such as $e_i = (v_a, v_b)$.

4 Dataset Construction

We extend the TempEL[1] dataset, a benchmark for temporal Entity Linking (EL), with Wikidata5M[2]. The resulting GCL-TempEL dataset comprises two text-based components inherited from TempEL, namely **entity description** and **mention context**, extended with

[1]https://cloud.ilabt.imec.be/index.php/s/RinXy8NgqdW58RW
[2]https://deepgraphlearning.github.io/project/wikidata5m

three graph-based components: **relation graph**, **feature graph**, and **feature matrix**. Additionally, it includes both positive and negative samples. Both the relation and feature graphs depict yearly entity relationships. The construction process is illustrated in Figure 3, and it introduces the following components:

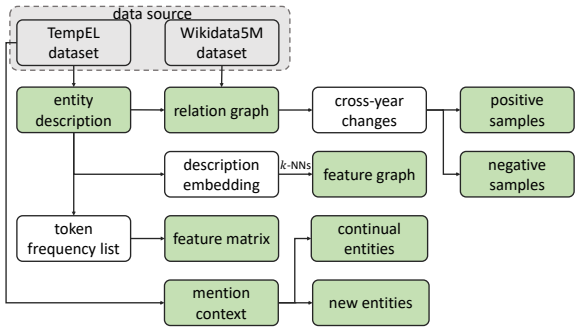

Figure 3: The dataset construction process. We use Wikidata5M to extend TempEL with *strutured graph* representations. For each year, we identify *newly added* and *removed* edges for a target entity. Furthermore, we extract *feature graph* and *feature matrix* based on the textual description of the target entity. The green section represents the input to our model.

Entity descriptions and mention contexts. First, we categorized each year of data from the TempEL dataset into *entity descriptions* and *mention context* parts based on the year. The *entity description* comprises the title, text, document ID, and the unique ID of the entity (its QID). The *mention context* consists of the text surrounding the entity mention (to the left and to the right), the mention itself, target entity as label, QID, and category. Furthermore, we categorized the mentions into two groups: those linked to *continual entities*, which exist across all the years in GCL-TempEL, and those linked to *new entities*, which are created in a specific year and do not exist in previous years.

Relation graph. We create a *relation graph* based on the Knowledge Graph (KG) relationships in the Wikidata5M dataset and the entity IDs in the TempEL dataset. We matched the entities involved in the relationships in Wikidata5M with the ones that exist in the TempEL dataset. Concretely, we keep a relationship if both entities are involved in a relationship in the Wikidata5M data and are also present in TempEL. The relation graph is an $n \times n$ adjacency matrix, where n represents the total number of entities in the GCL-TempEL dataset. Each row indicates whether an entity has a connection with another entity. Concretely, the adjacency matrix is made up of 0s and 1s. If entity i and entity j are connected, the value in the i^{th} row and j^{th} column of the matrix is 1; otherwise, it is 0. After we construct the relation graph, we use the differences in node relationships across the years (*cross-year changes* in Figure 3) to construct *positive* and *negative* samples, which are defined as follows:

1 *Positive samples*: sampled from a pool of newly added relationships between the target entity and its neighbors.
2 *Negative samples*: sampled from a pool of newly deleted relationships between the target entity and its neighbors.

Feature graph. KGs are inherently incomplete and sparse [18]. As a result, we consider that *relation graph* derived from Wikidata5M KG above does not contain all the possible relations between entities, which limits its expressiveness. To address this, we introduce *feature graph* which extends the edges in the *relation graph* with additional edges given by k-nearest neighbors (k-NN) with other entities. In order to create this graph, we use pre-trained bert-base-uncased model to embed the textual description of each of the entities. We use the resulting *description embeddings* to identify the k-NN entities for each of the target entities using cosine similarity. The resulting *feature graph* highlights the connections between entities based on their entity descriptions. We hypothesize that such connections will provide additional information that would allow to generate more representative entity embeddings for a given temporal snapshot. Similarly to the relation graph, the feature graph is represented as $n \times n$ adjacency matrix, where n is the total number of entities in the dataset. Each row indicates whether an entity has a connection with other entities. If entity i and entity j are connected, the value in the i^{th} row and j^{th} column of the matrix is 1; otherwise, it is 0.

Feature matrix. Building on previous research [29], we developed a *Feature matrix* to derive more representative entity embeddings based on their descriptions in the dataset. This matrix not only provides a robust representation of entities but also improves the graph aggregation process, enabling the generation of more nuanced embeddings. The goal of the *Feature matrix* is to represent each entity based on the tokens from entity descriptions in the dataset. After obtaining the token IDs for each entity using the pre-trained bert-base-uncased model, we filtered all token IDs based on their frequency of occurrence (see *token frequency list* in Figure 3). We retained those token IDs that appeared between 46 and 200 times. We discarded highly frequent token IDs since these tokens, such as 'is,' 'an,' 'the,' and other common words, do not offer meaningful differentiation among entities. Also, the less frequent token IDs were removed due to the possibility of them being meaningless noise or random codes, and including an excess of these rare tokens would make the matrix too sparse, slowing down computation. The resulting *feature matrix* is an $n \times m$ dimensional adjacency matrix, where n is the total number of entities in the dataset, and m is the number of retained token IDs. This matrix is composed of 1s and 0s, if the data in the i^{th} row and j^{th} column of the matrix is 1, it indicates that entity i contains the j^{th} token.

5 Approach

Figure 4 illustrates the framework of our model. The core idea is to exploit the relationships in *graph-based input* between entities across different years in the dataset. These relationships are contained in *relation graph* (G_r), *feature graph* (G_f) and *feature matrix* (X) (see Section 4 for details). When Knowledge Graph (KG) entities appear or disappear at a specific point in time, they can be used as positive and negative samples, respectively. Once these positive and negative samples are identified across different years in our GCL-TempEL dataset, we employ them in graph contrastive learning module. This module employs contrastive learning loss to efficiently adapt entity embeddings to temporal changes. Besides

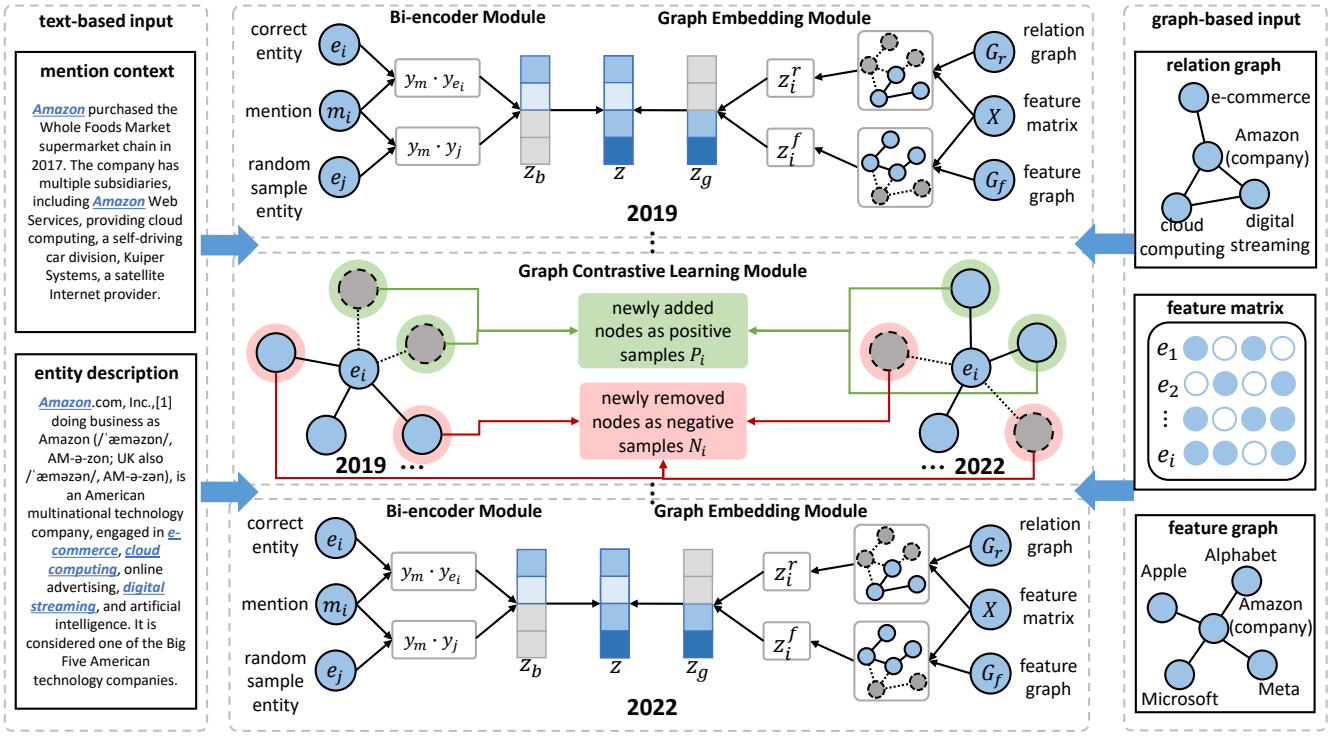

Figure 4: The proposed CYCLE architecture leverages both *text-based* (left) and *graph-based* (right) inputs. Additionally, it introduces a novel *Graph Contrastive Learning Module* to efficiently adapt entity representations to temporal changes in the graph-based inputs. The figure illustrates positive (green) and negative (red) samples used in this Graph Contrastive Learning Module to capture temporal changes in graph-based inputs of the year 2022 with respect to those of the year 2019 for entity e_i.

using the graph-based input X, G_r, G_f, CYCLE also leverages *textual information* (see left part of Figure 4) composed of *mention contexts* and *entity descriptions*.

First, the bi-encoder module in Section 5.1 employs two separate BERT transformers to transform mention context and entity description into dense vectors y_m and y_e. Entity candidates are scored via the dot product of these vectors. We introduce L_e to maximize the correct entity's score against randomly sampled entities. Second, we input the pre-constructed relation graph G_r, feature graph G_f, feature matrix X, and cross-year positive and negative samples into the Graph Embedding Module in Section 5.2. Our model encodes entities from relation and feature graphs and uses graph contrastive learning losses L_f and L_r across different years. Lastly, all the loss functions are unified for joint optimization.

5.1 Bi-encoder Module

Mention Representation. Following [32], the mention representation τ_m is constructed from tokens of the surrounding context and the mention:

$$\tau_m = [\text{CLS}]\ \text{ctxt}_l\ [\text{M}_s]\ \text{mention}\ [\text{M}_e]\ \text{ctxt}_r\ [\text{SEP}], \tag{1}$$

where ctxt_l, ctxt_r denote tokens before and after the mention, and $[\text{M}_s]$, $[\text{M}_e]$ tag the mention. The input's maximum length is set to 128, consistent with the baseline.

Entity Representation. The representation τ_e consists of tokens of the entity title and its description:

$$\tau_e = [\text{CLS}]\ \text{title}\ [\text{ENT}]\ \text{description}\ [\text{SEP}], \tag{2}$$

where [ENT] separates the title and description.

Encoding. We use the bi-encoder architecture from [32] to encode descriptions into the vectors y_e and y_m:

$$\boldsymbol{y_m} = \text{red}\left(T_1\left(\tau_m\right)\right), \tag{3}$$

$$\boldsymbol{y_e} = \text{red}\left(T_2\left(\tau_e\right)\right), \tag{4}$$

where, T_1 and T_2 are transformers, and red(.) takes the last layer of the output of the [CLS] token to reduce the sequence of vectors into a single vector.

Scoring. Entity candidate scores are computed via dot-product:

$$s\left(m, e_i\right) = \boldsymbol{y_m} \cdot \boldsymbol{y_{e_i}}. \tag{5}$$

5.2 Graph Embedding Module

We aim to enable the learning connections between nodes from the *relation graph* G_r. By inputting the relation graph G_r and node features X, we obtain specific embeddings, denoted as z_r.

In the relation graph, we consider the target node e_i that is connected to s other nodes $\{n_1, n_2, ..., n_s\}$. Thus, the set of neighbors for node e_i can be defined as N_i^s. For node e_i, each neighbor contributes differently to its embedding. To effectively integrate these

contributions, we employ an attention mechanism to aggregate messages from the neighbors to the target node e_i:

$$z_i^r = \sigma\left(\sum_{j \in N_i^s} \alpha_{i,j} \cdot x_j\right), \tag{6}$$

where σ is a nonlinear activation, x_j is the feature of node e_j, and $\alpha_{i,j}$ denotes the attention value of node j to node e_i. It is calculated as follows:

$$\alpha_{i,j} = \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}^\top \cdot \left[x_i \| x_j\right]\right)\right)}{\sum_{l \in N_i} \exp\left(\text{LeakyReLU}\left(\mathbf{a}^\top \cdot [x_i \| x_l]\right)\right)}, \tag{7}$$

where $\mathbf{a}$ is the attention vector and $\|$ denotes concatenate operation. Following [29], we randomly sample a subset of neighbors in N_i^s during each epoch. If the number of neighbors exceeds a predefined threshold, we sample neighbors as N_i^s. If the predefined threshold number of positive samples cannot be found, all the positive samples will be selected. In this case, the number of positive samples will be less than the predefined threshold. This way, we ensure that every node aggregates the same amount of information from neighbors, and promote diversity of embeddings in each epoch.

When using the feature graph G_f and node features X as inputs, the embeddings are denoted as z_i^f.

5.3 Cross-year Contrastive Module

The input feature graph embedding z_i^f and relation graph embedding z_i^r are passed through a multi-layer perceptron with a single hidden layer. This process maps the features into a representation space where the contrastive loss is calculated:

$$z_i^{fp} = W^{(2)}\sigma(W^{(1)}z_i^f + b^{(1)}) + b^{(2)}, \tag{8}$$

$$z_i^{rp} = W^{(2)}\sigma(W^{(1)}z_i^r + b^{(1)}) + b^{(2)}, \tag{9}$$

where, σ represents exponential linear unit, a type of non-linear activation function. The parameter sets $W^{(2)}, W^{(1)}, b^{(2)}, b^{(1)}$ are shared across the input graphs for embedding consistency.

To define the positive P_i^r and negative N_i^r samples for node e_i in relation graph G_r at time t_2:

$$P_i^r(t_2) = \{j \mid (i,j) \notin E_{t_1} \wedge (i,j) \in E_{t_2}\}, \tag{10}$$

$$N_i^r(t_2) = \{j \mid (i,j) \in E_{t_1} \wedge (i,j) \notin E_{t_2}\}, \tag{11}$$

here, E_{t_1} and E_{t_2} denote the edge sets at times t_1 and t_2, respectively. $(i,j) \notin E_{t_1}$ indicates that there was no edge between nodes e_i and e_j at t_1, whereas $(i,j) \in E_{t_2}$ indicates that an edge between nodes e_i and e_j exists at time t_2. The symbol $\wedge$ is used to denote the logical AND operation.

In a feature graph G_f, positive samples P_i^f includes all neighboring nodes directly connected to e_i, while the set of negative samples N_i^f includes all nodes that are not connected to e_i.

5.4 Objective Function

The objective function is calculated based on three components: two contrastive loss functions L_f and L_r, and a task-specific EL loss function L_e defined below.

Entity Linking Loss Function L_e. The objective is to train the network such that it maximizes the score of the correct entity compared to the other entities from the same batch. Formally, for each training pair (m_i, e_i) within a batch of N pairs, the loss L_e is defined as:

$$L_e\left(m_i, e_i\right) = -s\left(m_i, e_i\right) + \log \sum_{j=1}^{N} \exp\left(s\left(m_i, e_j\right)\right). \tag{12}$$

Contrastive Learning Loss Functions L_f and L_r. The contrastive loss function L_f and L_r for a given set of positive (P_i) and negative (N_i) samples. The purpose of L_f is to compute the contrastive loss where the embedding of node e_i is from graph G_f, while the positive and negative samples are from graph G_r. Conversely, L_r computes the contrastive loss where the embedding of node e_i is from graph G_r, and the positive and negative samples are from graph G_f. This approach maximizes the advantages of contrastive learning by comparing node embeddings from different graphs and different sets of positive and negative samples, thereby capturing complex structural and semantic information from the graphs:

$$L_f = -\log \frac{\sum_{j \in P_i^r} \exp\left(\frac{\text{sim}\left(z_i^{fp}, z_j^{rp}\right)}{\tau}\right)}{\sum_{k \in \{P_i^r \cup N_i^r\}} \exp\left(\frac{\text{sim}\left(z_i^{fp}, z_k^{rp}\right)}{\tau}\right)}, \tag{13}$$

$$L_r = -\log \frac{\sum_{j \in P_i^f} \exp\left(\frac{\text{sim}\left(z_i^{rp}, z_j^{fp}\right)}{\tau}\right)}{\sum_{k \in \left\{P_i^f \cup N_i^f\right\}} \exp\left(\frac{\text{sim}\left(z_i^{rp}, z_k^{fp}\right)}{\tau}\right)}, \tag{14}$$

where the function *sim* computes cosine similarity between two vectors, and the temperature parameter τ helps prevent the model from getting stuck in local optima during training. P_i^r and P_i^f are the sets of positive samples for node e_i, and N_i^r and N_i^f are the sets of negative samples for node e_i.

Overall Objective Function. The final objective function is a weighted sum of the individual L_e, L_r, and L_f loss functions calculated above:

$$L = aL_e + bL_f + cL_r, \tag{15}$$

where a, b and c are weights for the three losses defined above.

6 Evaluation

In this section, we evaluate the performance of the proposed model across three Entity Linking (EL) datasets. The implementation of our approach is based on the original codebase BLINK[3] and HeCo[4] [29]. We selected BLINK and SpEL[5] [22] because of their relevance and performance benchmarks in the field. BLINK has excellent scalability and serves as part of our model's codebase. Furthermore, SpEL is the latest state-of-the-art as of 2023. Comparison with these models highlights the influence of integrating additional structured *graph-based input* and conducting graph *contrastive learning* on node embeddings in CYCLE to mitigate temporal performance degradation. Experimental details can be found in [37].

[3]https://github.com/facebookresearch/BLINK
[4]https://github.com/liun-online/HeCo
[5]https://github.com/shavarani/SpEL

6.1 Datasets

Our proposed model is evaluated on three datasets which are summarized in Table 1.

Table 1: Summary of the used datasets' statistics.

Dataset	Train	Validation	Test	Entities
GCL-TempEL: Continual Entities	1,764	42,096	48,215	136,227
GCL-TempEL: New Entities	1,764	42,096	48,215	136,227
ZESHEL	49,275	10,000	10,000	492,321
WikiLinksNED	2,188,782	10,000	10,000	5,455,160

GCL-TempEL.[6] This dataset covers four years, from 2019 to 2022. GCL-TempEL is extended from Graph-TempEL[7] and incorporates additional *positive* and *negative* samples with respect to specific temporal snapshot. Each year within the dataset is divided into training (1,764 samples), validation (approximately 42k samples, matching the original TempEL dataset), and testing sets (approximately 48k samples, also matching the original TempEL dataset). The number of entities remains consistent across all temporal snapshots. It is important to note that the training set includes only 1,764 samples because the original TempEL training dataset, which contains around 130,000 samples, has only 1,764 samples of *new entities*. The *new entities* subset, reflecting evolving trends in KGs, comprises entities with limited historical data, posing a greater challenge for accurate distinction. To ensure a balanced comparison, we also randomly selected 1,764 samples for *continual entities*.

Zero-shot Entity Linking (ZESHEL).[8] This dataset covers various subjects, such as a fictional universe from a book or film series, mentions, and entities with detailed document descriptions. The train, validation, and test sets have 49K, 10K, and 10K examples, respectively. Specifically, the training set includes the following domains: 'american football', 'doctor who', 'fallout', 'final fantasy', 'military', 'pro wrestling', 'star wars', 'world of warcraft'. The validation set includes 'coronation street', 'muppets', 'ice hockey', and 'elder scrolls' domains. Finally, the test set includes 'forgotten realms', 'lego', 'star trek', and 'Yugioh' domains. A distinctive feature of this dataset is the variation in domains between the training set and the validation and test sets. This setup effectively reflects integrating newly added entities into a Knowledge Graph (KG). The dataset features a range of 10K to 100K entity candidates per domain, with a total of 500K entities.

WikiLinksNED.[9] This dataset was curated to address the challenges in the field of named entity disambiguation [23]. Spanning a wide array of topics, from historical events to contemporary figures, the entities in this dataset are associated with detailed document descriptions. The dataset is partitioned into train, dev, and test sets with 2.1M, 10K, and 10K examples, respectively.

[6]https://doi.org/10.5281/zenodo.12794944
[7]https://doi.org/10.5281/zenodo.12794960
[8]https://github.com/facebookresearch/BLINK/tree/main/examples/zeshel
[9]https://github.com/yasumasaonoe/ET4EL

6.2 Training Details

We compare our approach to the zero-shot EL BLINK model [32]. Concretely, we reuse the same hyperparameter settings and the same bert_uncased_L-8_H-512_A-8 pre-trained model to train the bi-encoder.

Parameter Settings. We utilize recall@N as our evaluation metric, with N being 1, 2, 4, 8, 16, 32, and 64. A prediction is correct if the true answer is within the model's top N predictions. The bi-encoder model is trained on the ZESHEL dataset for 5 epochs, using mention context and entity description tokens at a learning rate 1e-05. On the GCL-TempEL dataset, training is performed for 1 epoch under similar conditions. The model undergoes annual training and testing on test sets from 2019 to 2022, with each year's model being trained and validated on that year's data before testing across other years.

Training Environment and Inference Time. Software versions: Python 3.11.5; PyTorch 2.1.0_cuda 12.1_cudnn 8.9.2; Faiss-gpu 1.6.5; Numpy 1.26; SciPy 1.11.3; scikit-learn 1.3.2. All the experiments were run on a single A100 GPU with 40GB RAM.

6.3 Main Results

Table 2 showcases the effectiveness of our model in addressing temporal degradation on the GCL-TempEL dataset. We evaluate the performance on *continual* and *new* entities sets. Each column in the table represents the years' gap between the training and testing datasets, as denoted by the digits from 0 to 3. For instance, 0 indicates that training and testing datasets come from the same year, while 3 indicates that the model was trained in 2019 and tested in 2022. The rows are divided based on various metrics: @1 to @64. 'Boost' displays a comparison between our model CYCLE and SpEL model, calculated as Boost = $\frac{\text{Our Model's Result} - \text{SpEL Model's Result}}{\text{SpEL Model's Result}}$.

Table 2 demonstrates that our model consistently outperforms the BLINK baseline as well as SpEL, with its superiority becoming increasingly clear as the temporal gap between training and testing datasets grows. Specifically, when the temporal gap is one year, and the evaluation metric is @1, our model exhibits a 23.78% (continual entities) and 12.86% (new entities) improvement. This improvement boosts to 30.24% (continual entities) and 14.68% (new entities) with a two-year gap and jumps to 36.26% (continual entities) and 15.25% (new entities) when the gap extends to three years. The observed performance improvement can be attributed to the baseline model's limited exposure to previously unseen new entities, resulting in a lack of samples for effective learning. In our model, graph contrastive learning enhances the distinction of each node's unique characteristics. This approach effectively enables the model to identify and accurately represent new entities, even without similar historical data.

Figure 5 displays recall@1 results from the continual and new entities datasets. We compare our proposed model against the baseline models. The x-axis indicates the year gap between training and testing sets. Overall, our model consistently outperforms the baselines. Two different evaluation approaches are examined: 1) *forward and backward (f & b)*, referring to training on the past and testing on the future data, and vice versa, and 2) *only forward (f)* where the models are only trained on the past and evaluated on the future data. For example, when the year gap is 3, the *forward*

Table 2: The performance of models on the GCL-TempEL dataset across varying time gaps for both *new* and *continual* entity sets. The best results are highlighted in bold.

		0	1	2	3	0	1	2	3
		Continual Entities				New Entities			
@1	BLINK	0.177	0.181	0.182	0.177	0.132	0.132	0.132	0.142
	SpEL	0.229	0.234	0.228	0.221	0.172	0.169	0.167	0.192
	CYCLE	**0.286**	**0.289**	**0.297**	**0.301**	**0.184**	**0.191**	**0.192**	**0.222**
	Boost (%)	25.12	23.78	30.24	36.26	6.99	12.86	14.68	15.28
@2	BLINK	0.260	0.265	0.268	0.263	0.197	0.197	0.198	0.211
	SpEL	0.320	0.328	0.327	0.322	0.239	0.247	0.258	0.261
	CYCLE	**0.403**	**0.409**	**0.413**	**0.413**	**0.271**	**0.282**	**0.274**	**0.331**
	Boost (%)	26.11	24.73	26.27	28.31	13.23	14.15	6.29	26.73
@4	BLINK	0.357	0.364	0.367	0.362	0.277	0.277	0.278	0.294
	SpEL	0.429	0.436	0.430	0.429	0.329	0.340	0.333	0.354
	CYCLE	**0.522**	**0.527**	**0.532**	**0.543**	**0.378**	**0.389**	**0.388**	**0.434**
	Boost (%)	21.65	21.09	23.73	26.55	14.80	14.52	16.31	22.49
@8	BLINK	0.463	0.469	0.475	0.470	0.370	0.370	0.374	0.392
	SpEL	0.546	0.544	0.554	0.548	0.423	0.440	0.439	0.472
	CYCLE	**0.633**	**0.640**	**0.647**	**0.648**	**0.487**	**0.497**	**0.491**	**0.548**
	Boost (%)	15.94	17.60	16.77	18.30	15.11	12.93	12.00	16.25
@16	BLINK	0.571	0.576	0.581	0.578	0.472	0.471	0.474	0.491
	SpEL	0.645	0.645	0.656	0.652	0.539	0.541	0.554	0.551
	CYCLE	**0.719**	**0.719**	**0.724**	**0.715**	**0.593**	**0.602**	**0.596**	**0.644**
	Boost (%)	11.58	11.54	10.31	9.76	9.97	11.25	7.61	16.98
@32	BLINK	0.675	0.680	0.685	0.683	0.576	0.576	0.577	0.593
	SpEL	0.732	0.739	0.744	0.741	0.641	0.646	0.637	0.673
	CYCLE	**0.811**	**0.812**	**0.817**	**0.812**	**0.689**	**0.697**	**0.688**	**0.731**
	Boost (%)	10.90	9.91	9.81	9.64	7.54	7.96	7.98	8.56
@64	BLINK	0.769	0.774	0.778	0.776	0.677	0.676	0.679	0.694
	SpEL	0.820	0.827	0.825	0.824	0.732	0.733	0.739	0.754
	CYCLE	**0.871**	**0.872**	**0.874**	**0.878**	**0.780**	**0.784**	**0.778**	**0.811**
	Boost (%)	6.30	5.42	5.94	6.49	6.59	6.93	5.27	7.52
Ave. Boost (%)		16.80	16.29	17.58	19.33	10.60	11.51	10.02	16.26

and backward (f & b) setting includes two scenarios: the model trains in 2019 and tests in 2022, and trains in 2022 and tests in 2019. However, the *only forward (f)* setting only includes one scenario: the model trains in 2019 and tests in 2022. Notably, a zero-year gap implies the same years for training and testing are used, leading to equal recall values in both *forward and backward (f & b)* and *only forward (f)* setting. Our model's *only forward (f)* setting consistently outperforms its *forward and backward (f & b)* counterpart. We believe this demonstrates that when using previously non-existent entities as the training set, our model is more adept at capturing the evolving trends of KGs, thereby enhancing its ability to predict future developments.

Furthermore, our model's improvement margin diminishes as the recall metric threshold N is increased, as illustrated in Figure 6. The x-axis represents the year gap between training and testing datasets, while the y-axis shows our model's performance improvement over the SpEL model. Solid bars denote performance on the *new entities* dataset and hollow bars on *continual entities*. This diminishing effect is likely because at the @64 threshold, the model needs only to correctly predict one of the top 64 answers, thus allowing a higher error tolerance. Additionally, our model's performance enhancement grows with the year gap, as shown by the regression lines for both *continual* and *new* entities.

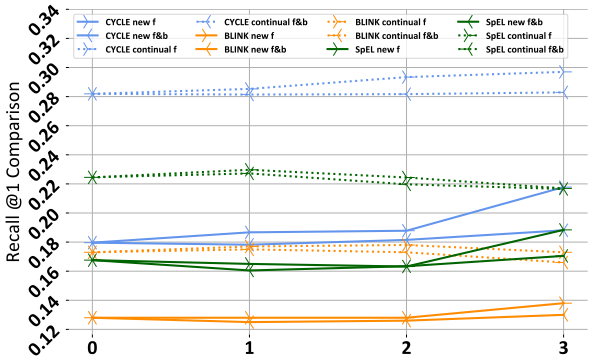

Figure 5: Recall@1 performance over 3-year gaps shows CYCLE outperforming both the BLINK and the SpEL on *continual* and *new* entities. Performance gains increase with larger temporal gaps, highlighting the robustness of our approach.

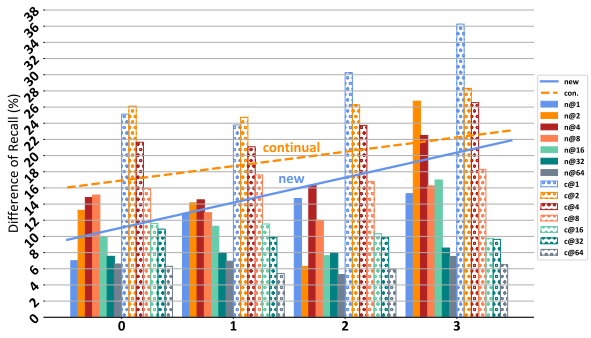

Figure 6: This figure compares our model with the SpEL across 3-year gaps in *only forward* settings. Improvements on the *new* and *continual entities* datasets are depicted by solid and hollow bars respectively. We observe a consistent growth in the performance gain for both types of entities (see regression lines) as the temporal gap increases, demonstrating the temporal robustness of our approach.

6.4 Results on Low-degree Entities and Fairness Analysis

In Figure 7, we analyze the improvement of our model's performance on entities with different degrees. The model is trained on the 2019 *new entities* training set and tested on the test sets from 2019 to 2022. The figure shows the entity's degree on the x-axis and the corresponding improvement in model performance compared to the BLINK model on the y-axis. We observe that our model has

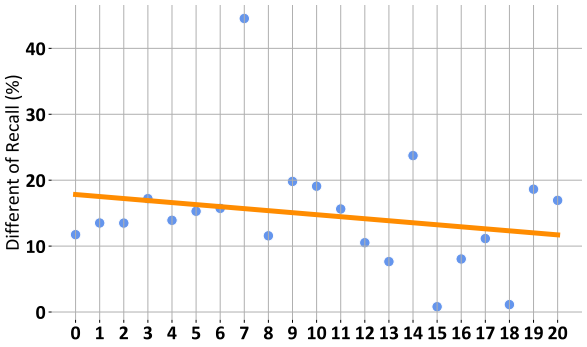

Figure 7: CYCLE performance improvement over the BLINK model as the degree of target entities in the *relation graph* increases (x-axis). The orange regression line illustrates a trend where CYCLE, on average, achieves better performance enhancements, particularly on low-degree entities.

a more pronounced improvement for lower-degree entities. When facing entities with a relatively high degree and entities with more neighbors, our model still demonstrates an improvement, although it is less noticeable.

We hypothesize that this trend is partly due to sparse information of low-degree nodes. Such nodes are highly sensitive to new connections, making graph contrastive learning module especially effective at integrating new data and adapting to dynamic changes. This sensitivity is critical, as it allows low-degree nodes to more effectively capture and utilize new information, improving their embeddings and adaptability over time.

6.5 Results on Non-temporal datasets

Our model achieves state-of-the-art performance on temporally evolving GCL-TempEL dataset (see Table 2). In this section, we evaluate the performance of our model in a traditional setting, characterized by *static* (i.e., lacking temporal evolution) EL datasets without graph-based input. Concretely, Table 3 describes EL results using recall@N metric, for our CYCLE model compared to BLINK and SpEL models on the ZESHEL and WikilinksNED *static* datasets. Even without graph-based input and temporal evolution in these datasets, our model maintains consistent performance on par with BLINK, underscoring its adaptability.

Table 3: Performance of CYCLE on EL datasets that do not contain graph-based input and are static. Our model continues to exhibit competitive performance.

Dataset	Model	@1	@4	@8	@16	@32	@64
Zeshel: Forgotten Realms	BLINK	0.5183	0.7400	0.7950	0.8375	0.8683	0.8942
	SpEL	**0.5717**	**0.8092**	**0.8646**	**0.8969**	**0.9373**	**0.9498**
	CYCLE	0.5150	0.7423	0.7963	0.8298	0.8676	0.8978
Zeshel: Lego	BLINK	0.4170	0.6647	0.7548	0.8090	0.8599	0.8841
	SpEL	**0.4672**	**0.7216**	**0.8113**	**0.8685**	**0.9197**	**0.9420**
	CYCLE	0.4127	0.6726	0.7547	0.8059	0.8628	0.8825
Zeshel: Star Trek	BLINK	0.3717	0.5798	0.6475	0.7052	0.7563	0.7999
	SpEL	**0.4316**	**0.6358**	**0.7030**	**0.7574**	**0.8122**	**0.8534**
	CYCLE	0.3749	0.5839	0.6493	0.7083	0.7528	0.7936
Zeshel: Yugioh	BLINK	0.2828	0.4769	0.5504	0.6094	0.6577	0.6935
	SpEL	**0.3361**	**0.5270**	**0.6056**	**0.6615**	**0.7110**	**0.7529**
	CYCLE	0.2745	0.4764	0.5476	0.6075	0.6533	0.6946
WikilinksNED	BLINK	0.1721	0.4192	0.5467	0.6505	0.7340	0.7907
	SpEL	**0.2315**	**0.4761**	**0.5976**	**0.7084**	**0.7913**	**0.8414**
	CYCLE	0.1746	0.4276	0.5657	0.6584	0.7275	0.7934
Average	BLINK	0.3524	0.5761	0.6589	0.7223	0.7752	0.8125
	SpEL	**0.4076**	**0.6339**	**0.7164**	**0.7785**	**0.8343**	**0.8679**
	CYCLE	0.3503	0.5806	0.6627	0.7220	0.7728	0.8124

7 Conclusion and Future Work

This paper introduces **CYCLE**, a model specifically designed to use graph contrastive learning to mitigate temporal degradation in evolving Knowledge Graphs (KGs). The model captures cross-year changes between entities by utilizing newly added or removed graph nodes as positive and negative samples in a contrastive learning framework. We conducted experiments on three Entity Linking (EL) datasets, setting a new benchmark on the temporally evolving GCL-TempEL dataset. Specifically, our model achieved a 13.90% performance improvement over the SpEL model when the time gap is one year, and this improvement increased to 17.79% as the gap extended to three years. Additionally, our model demonstrated competitive performance on static datasets. Looking ahead, we identify two key areas for future research:

Multimodal Temporal KGs with Contrastive Learning. In addition to temporal information, TKGs can also contain multimodal data, such as text, images, and audio. This multimodal data can be used further to improve the performance of contrastive learning for TKGs. For example, a pair of entities with a relationship in multiple timestamps that are also mentioned in the audio can be considered a more informative positive sample than a pair of entities with a relationship in multiple timestamps but not in any other type of data.

Temporal KGs with Large Language Models. Considering the challenge of aligning similar concepts across languages where direct translations often fail to convey identical meanings, using Large Language Models (LLMs) for aligning conceptually similar entities in multilingual KGs is a promising direction. This approach can improve the coherence of these graphs. If entity descriptions and entity relationships exist at language A, the model could benefit from these diverse language resources, further improving language B accuracy and better preventing temporal degradation.

Acknowledgments

The first author is supported by the China Scholarship Council (NO. 202206540007) and the University of Amsterdam. This funding source had no influence on the study design, data collection, analysis, or manuscript preparation and approval. This work is partially supported by the EU's Horizon Europe programme, in the ENEXA project (grant Agreement no. 101070305).

References

[1] GP Shrivatsa Bhargav, Dinesh Khandelwal, Saswati Dana, Dinesh Garg, Pavan Kapanipathi, Salim Roukos, Alexander Gray, and L Venkata Subramaniam. 2022. Zero-shot entity linking with less data. In *Findings of the Association for Computational Linguistics: NAACL 2022*. 1681–1697.

[2] Veronika Bogina, Tsvi Kuflik, Dietmar Jannach, Maria Bielikova, Michal Kompan, and Christoph Trattner. 2023. Considering temporal aspects in recommender systems: a survey. *User Modeling and User-Adapted Interaction* 33, 1 (2023), 81–119.

[3] Elizabeth Boschee, Jennifer Lautenschlager, Sean O'Brien, Steve Shellman, James Starz, and Michael Ward. 2015. ICEWS coded event data. *Harvard Dataverse* 12 (2015), 2.

[4] Samuel Broscheit. 2019. Investigating Entity Knowledge in BERT with Simple Neural End-To-End Entity Linking. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, Mohit Bansal and Aline Villavicencio (Eds.). Association for Computational Linguistics, Hong Kong, China, 677–685. https://doi.org/10.18653/v1/K19-1063

[5] Borui Cai, Yong Xiang, Longxiang Gao, He Zhang, Yunfeng Li, and Jianxin Li. 2023. Temporal knowledge graph completion: a survey. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence (IJCAI '23)*. Article 734, 9 pages. https://doi.org/10.24963/ijcai.2023/734

[6] Jiahui Chen, Shaobo Zhong, Xingtong Ge, Weichao Li, Hanjiang Zhu, and Ling Peng. 2021. Spatio-temporal knowledge graph for meteorological risk analysis. In *2021 IEEE 21st International Conference on Software Quality, Reliability and Security Companion (QRS-C)*. IEEE, 440–447.

[7] Yuanfei Dai, Wenzhong Guo, and Carsten Eickhoff. 2024. Wasserstein adversarial learning based temporal knowledge graph embedding. *Information Sciences* 659 (2024), 120061. https://doi.org/10.1016/j.ins.2023.120061

[8] Joel Jang, Seonghyeon Ye, Changho Lee, Sohee Yang, Joongbo Shin, Janghoon Han, Gyeonghun Kim, and Minjoon Seo. 2022. TemporalWiki: A Lifelong Benchmark for Training and Evaluating Ever-Evolving Language Models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. 6237–6250.

[9] Hiroya Kato, Kento Hasegawa, Seira Hidano, and Kazuhide Fukushima. 2024. EdgePruner: Poisoned Edge Pruning in Graph Contrastive Learning. In *2024 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*. 309–326. https://doi.org/10.1109/SaTML59370.2024.00022

[10] Kalev Leetaru and Philip A Schrodt. 2013. Gdelt: Global data on events, location, and tone, 1979–2012. In *ISA annual convention*, Vol. 2. Citeseer, 1–49.

[11] Jiang Li, Xiangdong Su, and Guanglai Gao. 2023. Teast: Temporal knowledge graph embedding via archimedean spiral timeline. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 15460–15474.

[12] Yuxi Li, Shuxuan Xie, Zhibo Wan, Haibin Lv, Houbing Song, and Zhihan Lv. 2023. Graph-powered learning methods in the Internet of Things: A survey. *Machine Learning with Applications* 11 (2023), 100441. https://doi.org/10.1016/j.mlwa.2022.100441

[13] Ruotong Liao, Xu Jia, Yangzhe Li, Yunpu Ma, and Volker Tresp. 2024. GenTKG: Generative Forecasting on Temporal Knowledge Graph with Large Language Models. In *Findings of the Association for Computational Linguistics: NAACL 2024*. 4303–4317.

[14] Lajanugen Logeswaran, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, Jacob Devlin, and Honglak Lee. 2019. Zero-shot entity linking by reading entity descriptions. *arXiv preprint arXiv:1906.07348* (2019).

[15] Farzaneh Mahdisoltani, Joanna Biega, and Fabian M Suchanek. 2013. Yago3: A knowledge base from multilingual wikipedias. In *CIDR*.

[16] Rostislav Nedelchev, Debanjan Chaudhuri, Jens Lehmann, and Asja Fischer. 2020. End-to-end entity linking and disambiguation leveraging word and knowledge graph embeddings. *arXiv preprint arXiv:2002.11143* (2020).

[17] Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2024. Unifying Large Language Models and Knowledge Graphs: A Roadmap. *IEEE Transactions on Knowledge and Data Engineering* 36, 7 (2024), 3580–3599. https://doi.org/10.1109/TKDE.2024.3352100

[18] Aleksandar Pavlović and Emanuel Sallinger. 2023. ExpressivE: A Spatio-Functional Embedding For Knowledge Graph Completion. In *International Conference on Learning Representations*.

[19] Manoj Prabhakar Kannan Ravi, Kuldeep Singh, Isaiah Onando Mulang, Saeedeh Shekarpour, Johannes Hoffart, and Jens Lehmann. 2021. CHOLAN: A modular approach for neural entity linking on Wikipedia and Wikidata. *arXiv preprint arXiv:2101.09969* (2021).

[20] Petar Ristoski, Zhizhong Lin, and Qunzhi Zhou. 2021. KG-ZESHEL: Knowledge Graph-Enhanced Zero-Shot Entity Linking. In *Proceedings of the 11th Knowledge Capture Conference* (Virtual Event, USA) *(K-CAP '21)*. Association for Computing Machinery, New York, NY, USA, 49–56. https://doi.org/10.1145/3460210.3493549

[21] Aditya Sharma, Apoorv Saxena, Chitrank Gupta, Seyed Mehran Kazemi, Partha Talukdar, and Soumen Chakrabarti. 2022. Twirgcn: Temporally weighted graph convolution for question answering over temporal knowledge graphs. *arXiv preprint arXiv:2210.06281* (2022).

[22] Hassan Shavarani and Anoop Sarkar. 2023. SpEL: Structured Prediction for Entity Linking. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Singapore, 11123–11137.

[23] Sameer Singh, Amarnag Subramanya, Fernando Pereira, and Andrew McCallum. 2012. Wikilinks: A large-scale cross-document coreference corpus labeled via links to Wikipedia. *University of Massachusetts, Amherst, Tech. Rep. UM-CS-2012* 15 (2012).

[24] Wangbin Sun, Jintang Li, Liang Chen, Bingzhe Wu, Yatao Bian, and Zibin Zheng. 2024. Rethinking and Simplifying Bootstrapped Graph Latents. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining* (Merida, Mexico) *(WSDM '24)*. Association for Computing Machinery, New York, NY, USA, 665–673. https://doi.org/10.1145/3616855.3635842

[25] Jiapu Wang, Boyue Wang, Meikang Qiu, Shirui Pan, Bo Xiong, Heng Liu, Linhao Luo, Tengfei Liu, Yongli Hu, Baocai Yin, et al. 2023. A survey on temporal knowledge graph completion: Taxonomy, progress, and prospects. *arXiv preprint arXiv:2308.02457* (2023).

[26] Peng Wang, Jiangheng Wu, and Xiaohang Chen. 2022. Multimodal entity linking with gated hierarchical fusion and contrastive training. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 938–948.

[27] Ruijia Wang, Xiao Wang, Chuan Shi, and Le Song. 2022. Uncovering the Structural Fairness in Graph Contrastive Learning. *Advances in Neural Information Processing Systems* 35 (2022), 32465–32473.

[28] Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2021. KEPLER: A unified model for knowledge embedding and pre-trained language representation. *Transactions of the Association for Computational Linguistics* 9 (2021), 176–194.

[29] Xiao Wang, Nian Liu, Hui Han, and Chuan Shi. 2021. Self-Supervised Heterogeneous Graph Neural Network with Co-Contrastive Learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining* (Virtual Event, Singapore) *(KDD '21)*. Association for Computing Machinery, New York, NY, USA, 1726–1736. https://doi.org/10.1145/3447548.3467415

[30] Zihao Wang, Kwun Ping Lai, Piji Li, Lidong Bing, and Wai Lam. 2019. Tackling long-tailed relations and uncommon entities in knowledge graph completion. *arXiv preprint arXiv:1909.11359* (2019).

[31] Han Wu, Jie Yin, Bala Rajaratnam, and Jianyuan Guo. 2023. Hierarchical Relational Learning for Few-Shot Knowledge Graph Completion. In *The Eleventh International Conference on Learning Representations*.

[32] Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. 2020. Scalable Zero-shot Entity Linking with Dense Entity Retrieval. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (Eds.). Association for Computational Linguistics, Online, 6397–6407. https://doi.org/10.18653/v1/2020.emnlp-main.519

[33] Zonghai Yao, Liangliang Cao, and Huapu Pan. 2020. Zero-shot Entity Linking with Efficient Long Range Sequence Modeling. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, Trevor Cohn, Yulan He, and Yang Liu (Eds.). Association for Computational Linguistics, Online, 2517–2522. https://doi.org/10.18653/v1/2020.findings-emnlp.228

[34] Yuning You, Tianlong Chen, Yang Shen, and Zhangyang Wang. 2021. Graph Contrastive Learning Automated. In *Proceedings of the 38th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 139)*, Marina Meila and Tong Zhang (Eds.). PMLR, 12121–12132.

[35] Yue Yu, Xiao Wang, Mengmei Zhang, Nian Liu, and Chuan Shi. 2023. Provable Training for Graph Contrastive Learning. In *Advances in Neural Information Processing Systems*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (Eds.), Vol. 36. Curran Associates, Inc., 50327–50345.

[36] Klim Zaporojets, Lucie-Aimée Kaffee, Johannes Deleu, Thomas Demeester, Chris Develder, and Isabelle Augenstein. 2024. TempEL: linking dynamically evolving and newly emerging entities. In *Proceedings of the 36th International Conference on Neural Information Processing Systems* (New Orleans, LA, USA) *(NIPS '22)*. Curran Associates Inc., Red Hook, NY, USA, Article 135, 17 pages.

[37] Pengyu Zhang. 2024. CYCLE: Cross-Year Contrastive Learning in Entity- Linking Supplementary Material. https://doi.org/10.5281/zenodo.12790219

[38] Sheng Zhang, Hao Cheng, Shikhar Vashishth, Cliff Wong, Jinfeng Xiao, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. 2021. Knowledge-rich self-supervision for biomedical entity linking. *arXiv preprint arXiv:2112.07887* (2021).

[39] Yuyue Zhao, Xiang Wang, Jiawei Chen, Yashen Wang, Wei Tang, Xiangnan He, and Haiyong Xie. 2022. Time-Aware Path Reasoning on Knowledge Graph for Recommendation. *ACM Trans. Inf. Syst.* 41, 2, Article 26 (dec 2022), 26 pages. https://doi.org/10.1145/3531267

[40] Yanqiao Zhu, Yichen Xu, Qiang Liu, and Shu Wu. 2021. An Empirical Study of Graph Contrastive Learning. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, Vol. 1.