Measuring Graph-to-Graph Semantic Similarity in Knowledge Graphs: An Empirical Evaluation of Knowledge Graph Embeddings

Anonymous Author(s)

Abstract

A Knowledge Graph (KG) represents facts as structured triples and is widely used to organize relational knowledge across diverse domains. Just as textual information ranges from words and sentences to complete documents, KG information can be interpreted at multiple levels, from entities, relations, and triples to subgraphs and entire KGs. However, existing KG embedding methods mainly focus on entities, relations, and triples, leaving graph-level semantics largely unaddressed. Conventional graph-level methods, which typically compare graphs based on structural patterns, are also insufficient because structural similarity alone cannot guarantee semantic similarity between KGs. To evaluate how well different methods capture such graph-level semantic information, we study graph-to-graph semantic similarity, which determines whether a pair of KGs represents semantically corresponding underlying information. To obtain reliable ground-truth correspondences, we construct a semantic matching dataset by modifying text documents, extracting KGs from both original and modified documents, and transferring their known correspondences to KG pairs. We compare text-based, structure-based, and KG embedding-based approaches on each dataset. For the KG embedding-based approach, we introduce two scoring functions: *EmbPairSim*, which uses maximal pairwise entity similarity, and *AvgEmbSim*, which uses a frequency-weighted centroid. Experiments on WikiText-2 and CC-News show that *EmbPairSim* achieves up to 5.3 pp higher MRR than Sentence-BERT while using substantially fewer parameters. These results suggest that KGE representations can serve as compact and effective signals for graph-to-graph semantic similarity in KGs. Our code is available at https://anonymous.4open.science/r/KG-to-KG-Semantic-Similarity-CD43/.

Keywords

Knowledge Graph, Knowledge Graph Embedding, Semantic Similarity, Knowledge Graph Similarity

1 Introduction

A *Knowledge Graph* (KG) represents a fact as a triple (h, r, t) that connects a **h**ead entity h to a **t**ail entity t via a typed **r**elation r—e.g., `(Paris, located_in, France)`. Due to their ability to organize complex relational knowledge in a structured form, KGs have become a widely adopted knowledge representation across diverse domains. Examples include encyclopedic knowledge in Wikidata [29], biomedical knowledge in BioKG [33], legal knowledge in LegalKG [7], scholarly knowledge in ORKG [9], event-centric knowledge in EventKG [8], and visual relationships in scene graphs [12].

Information stored in KGs can be analyzed at multiple levels, ranging from entities and relations to triples, subgraphs, and entire KGs. Entities and relations denote individual units and the types of connections between them. Triples represent specific relational facts. Subgraphs, formed by multiple triples, can capture broader semantic units, such as events, research contributions, or scene regions. An entire KG can represent a complete information source, such as encyclopedic, biomedical, scholarly, visual, or document-derived knowledge. Appendix A illustrates this multi-level view across representative KG domains. This view is analogous to textual information, which can be studied at different levels ranging from individual words and sentences to complete documents.

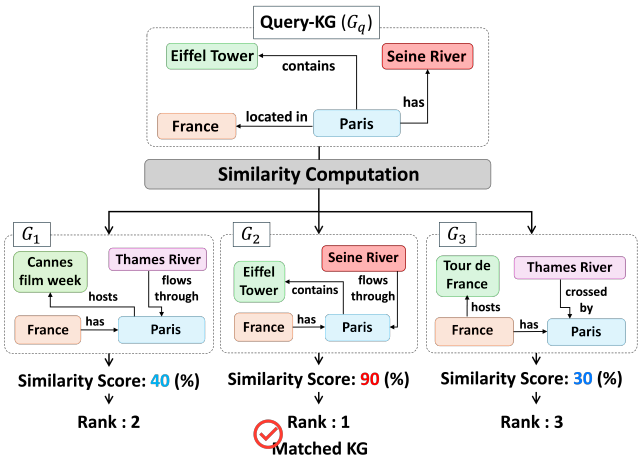


Figure 1: Overview of our semantic matching task.

Although semantic representations of KGs have been extensively studied, most existing approaches focus on local components such as entities, relations, and triples rather than subgraphs or entire KGs. Triple-centered language-model approaches such as KG-BERT [32], KEPLER [30], and RLKB [6] verbalize individual triples and encode them with Transformers [28], but rely primarily on triple-level semantics rather than the graph structure of an entire KG. Knowledge Graph Embedding (KGE) models, including transductive approaches [3, 25, 26, 31] and inductive approaches [13], incorporate both semantic and structural information. Nevertheless, these methods mainly model local KG components, such as entities, relations, and triples, rather than the semantic information represented by an entire KG.

Addressing KGs at the graph level may appear to align with conventional graph-level tasks. However, conventional graph-level tasks usually characterize graphs based on structural patterns, such as topology, connectivity, and substructures. This structural perspective is useful, but it is not sufficient for KGs, because KGs are constructed by organizing semantic information. In particular, structural similarity alone cannot guarantee that two KGs represent similar information.

Together, these limitations indicate that graph-level semantic information in KGs remains insufficiently addressed. Therefore, we aim to evaluate how well different types of methods can capture the semantic information represented by an entire KG. A natural evaluation setting for this purpose is graph-to-graph semantic similarity, where semantically corresponding KGs should be identified among other candidate graphs. Specifically, we formulate the task as determining whether two KGs represent the same underlying information at the graph level, rather than whether they merely share similar structures. Figure 1 exemplifies the task. Given a query KG, G_q (top), we rank candidate graphs $\{G_i\}$ such that the one encoding the *same real-world situation* surfaces first. An effective measure requires semantic awareness to bridge lexical variation, structural sensitivity to account for graph-level organization, and scalability to handle document-sized KGs.

To evaluate this task, however, we need reliable ground-truth correspondences between KGs. Such correspondences are difficult to obtain from existing KGs, which are often already processed into graph form, making it difficult to trace their original sources or determine whether two independently constructed KGs should be regarded as semantically similar. To obtain controllable semantic correspondences, we instead start from text documents. By applying meaning-preserving modifications, such as lexical substitution and paraphrasing, each original document can be paired with a modified document that expresses the same underlying information. After converting both the original and modified documents into KGs, the known document-level correspondence can be transferred to the resulting KG pair. This allows us to construct evaluation data in which each query KG has a clear ground-truth counterpart.

Based on this idea, we construct a **semantic matching** dataset: (i) documents from *WikiText-2* and *CC-News* are parsed into KGs with an LLM pipeline; (ii) each document is paraphrased at six lexical/structural intensities; (iii) KGs are re-extracted from the paraphrases; and (iv) every query graph must retrieve its true counterpart among hundreds of distractors. We report Hits@5, MRR, and NDCG across the graph pairs.

Using each dataset, we empirically compare text-based, structure-based, and KG embedding-based approaches for graph-to-graph semantic similarity. For the KG embedding-based approach, we introduce two complementary, scoring functions that lift off-the-shelf KGEs to the graph level: (1) **EmbPairSim**: computes maximal pairwise cosine similarity between entities (and optionally relations) in two graphs, preserving fine-grained correspondences; (2) **AvgEmbSim**: forms a single frequency-weighted centroid vector per graph, enabling more efficient retrieval.

For comparison, we use Sentence-BERT (SBERT) [20] as the text-based approach and Graph Kernel methods [22, 24] as the structure-based approach. For SBERT, KG triples are verbalized before being used as input. Across our dataset, *EmbPairSim* achieves up to 5.3 pp higher MRR than Sentence-BERT while using substantially fewer parameters. These results suggest that KGE representations can serve as compact and effective signals for graph-to-graph semantic similarity in KGs.

Our key contributions are as follows:

- **Task & dataset. To the best of our knowledge, we release the first benchmark specifically targeting document-derived KG-to-KG semantic retrieval under controlled paraphrasing conditions.**
- **Lightweight KGE aggregation. We propose *EmbPairSim* and *AvgEmbSim*, two scoring functions that require *no* retraining and run in sub-second time.**
- **Empirical findings. *EmbPairSim* achieves up to 5.3 pp higher MRR than Sentence-BERT while using an order of magnitude fewer parameters; an ablation study shows that frequency weighting generally helps, mean-centering is important for INGRAM-based similarity, and relation embeddings can introduce noise.**

2 Related Work

2.1 Language-based Representations

Word embeddings like Word2Vec [16] represent words with similar contexts close together in a continuous vector space. Sentence-level models such as Sentence-BERT (SBERT) [20] extend this to entire sentences or documents by capturing contextual information across tokens. These embeddings support semantic similarity, retrieval, and clustering. In our work, we leverage SBERT to measure semantic similarity between verbalized Knowledge Graphs (KGs). This approach serves as a language-based baseline for assessing graph-level similarity.

2.2 Structure-based Approaches

Graph-based methods compare graph structures by identifying matching substructures or edit operations. Early approaches such as Graph Edit Distance (GED) [1] and Ullmann's subgraph matching [27] provide exact structural similarity but are computationally expensive and do not scale well. Graph kernels address this by mapping graphs to high-dimensional feature spaces based on statistical patterns, enabling efficient similarity computation. Examples include the Vertex and Edge Histogram Kernels [24], which use label distributions, and the Weisfeiler-Lehman (WL) kernel [22], which iteratively relabels nodes to capture multi-level structural information. These kernels balance structural detail and scalability, making them practical for large graphs. In our work, we use graph kernels as structure-based baselines to evaluate how well surface-level structural features approximate semantic similarity between KGs.

2.3 Knowledge Graph Embedding (KGE) Methods

Knowledge Graph Embedding (KGE) methods map entities and relations into continuous vector spaces while preserving structural properties. They support tasks such as link prediction, entity classification, and graph completion. TransE [3] models relations as translations in embedding space, while DistMult [31] uses a multiplicative scoring function based on element-wise interactions. ComplEx [26] extends this by using complex-valued embeddings for richer relational patterns, and RotatE [25] models relations as rotations in complex space. These models are transductive and require all entities and relations during training. In contrast, INGRAM [13] follows an inductive paradigm: a graph neural network derives embeddings for previously unseen entities and relations directly from

their local graph structure. It relies exclusively on structural context and does not retain any entity or relation vectors produced during training. These KGE models are analogous to word embeddings but additionally encode graph-specific structure. We use them as structure-aware baselines to assess graph-level semantic similarity.

3 Preliminaries

3.1 Notations

A Knowledge Graph (KG) is defined as a tuple $G = (\mathcal{E}, \mathcal{R}, \mathcal{T})$, where $\mathcal{E}$ is the set of entities ($|\mathcal{E}|$ denotes its size), $\mathcal{R}$ is the set of relations ($|\mathcal{R}|$ denotes its size), and $\mathcal{T} \subseteq \{(h, r, t) \mid h, t \in \mathcal{E}, r \in \mathcal{R}\}$ is the set of triples, and $|\mathcal{T}|$ is their total count. Each triple (h, r, t) represents a directed relation from head entity h to tail entity t via relation r. Thus, a KG can be viewed as a directed labeled graph with entities as nodes and triples as edges.

3.2 Knowledge graph as a Document

Textual data exhibits a hierarchical structure: words form sentences, sentences combine into paragraphs, and paragraphs compose a document. Although a knowledge graph arranges entities and relations in a non-linear network without a fixed reading order, we can view it through the same hierarchy: entities and relations act as words, each triplet as a sentence, and subgraphs—or the entire graph—as paragraphs or even a full document. Viewing KGs this way allows us to apply document-level operations, such as similarity measurement, while capturing both structure and semantics.

3.3 Graph Kernel-based Approach

As a baseline for measuring similarity between KGs, we use graph kernel methods. Given two KGs, G and G', their similarity is computed as:

$$S_{\text{kernel}} = K(G, G'). \tag{1}$$

This approach relies purely on structural features. We use two kernels in our main experiments: the Vertex Histogram Kernel [24], and the Weisfeiler-Lehman (WL) kernel [22]. The Vertex Histogram Kernel compares distributions of discrete entity identifiers without using learned semantic embeddings. The WL Kernel further incorporates local graph topology by iteratively updating node labels, providing richer structural similarity than simple histograms.

4 Similarity Evaluation Setup

Problem formulation. To objectively evaluate the graph-to-graph semantic similarity task, we formulate a semantic matching problem between two KG sets. Let $\mathcal{G} = \{G_1, \ldots, G_m\}$ be a set of candidate KGs and let $\mathcal{G}' = \{G'_1, \ldots, G'_n\}$ be a set of query KGs, where $m \geq n$. This formulation assumes that each G'_i represents the same underlying information as G_i, while it is distinct from other candidate graphs G_j ($j \neq i$). Based on this assumption, the goal is to identify, for each query KG $G'_i \in \mathcal{G}'$, the candidate KG $G_i \in \mathcal{G}$ that semantically matches it.

Construction of aligned KG pairs. To obtain controllable semantic correspondences, we first construct semantically corresponding document pairs and then convert each document into a KG. Specifically, given a document set $\mathcal{D} = \{D_1, \ldots, D_m\}$, we first select a subset $\mathcal{D}_n = \{D_1, \ldots, D_n\} \subset \mathcal{D}$ ($m \geq n$). We then generate modified documents $\mathcal{D}' = \{D'_1, \ldots, D'_n\}$ using a document modification function that preserves the meaning of the original document. Each modified document D'_i preserves the underlying information of its original document D_i while remaining distinct from other documents D_j ($j \neq i$).

We then convert both original and modified documents into KGs using the same LLM pipeline. Specifically, using a same Large Language Model (LLM) [19] pipeline, we extract a KG G_i from each original document D_i and a KG G'_i from each modified document D'_i, forming $\mathcal{G} = \{G_1, \ldots, G_m\}$ and $\mathcal{G}' = \{G'_1, \ldots, G'_n\}$, respectively Since G_i and G'_i are extracted from semantically corresponding documents, the pair (G'_i, G_i) provides a ground-truth semantic match. As a result, we obtain KG sets with controllable semantic correspondences that are suitable for the semantic matching problem

Evaluation protocol. Given the constructed KG sets $\mathcal{G}$ and $\mathcal{G}'$, we compute pairwise similarity scores between each query KG $G'_i \in \mathcal{G}'$ and every candidate KG $G_j \in \mathcal{G}$. These scores form a similarity matrix whose (i, j)-th entry indicates the predicted semantic similarity between G'_i and G_j. For each query KG G'_i, candidate KGs are ranked by their similarity scores. The ranking is evaluated by checking whether the ground-truth counterpart G_i is ranked highly among all candidates. We report standard retrieval metrics, including Hits, NDCG, and MRR.

5 Methods

5.1 Embedding-Based Pairwise Similarity

We propose **Embedding-based Pairwise Similarity (*EmbPairSim*)** to measure semantic similarity between two Knowledge Graphs (KGs), $G = (\mathcal{E}, \mathcal{R}, \mathcal{T})$ and $G' = (\mathcal{E}', \mathcal{R}', \mathcal{T}')$. Instead of exact label matching, which is often too rigid, we use embeddings-vector representations of entities and relations-to compare elements based on their semantic meaning.

5.1.1 Embedding Generation. We generate embeddings for all entities and relations using an embedding function $\text{Embed}(\cdot)$. For entities, this yields:

$$\mathbf{e}_i = \text{Embed}(e_i), \text{ for } e_i \in \mathcal{E}, \tag{2}$$

$$\mathbf{e}'_j = \text{Embed}(e'_j), \text{ for } e'_j \in \mathcal{E}'. \tag{3}$$

Each $\mathbf{e}_i$ and $\mathbf{e}'_j$ encodes the semantic meaning of the corresponding entity in G and G', respectively. The embedding function $\text{Embed}(\cdot)$ can be instantiated either by a transductive KGE model, where entity embeddings for both G and G' are learned jointly in a shared embedding space by training on the union of their triples, or by an inductive KGE model that derives embeddings from graph structure using pretrained parameters.

5.1.2 Mean Centering. To align embeddings from both graphs, we apply **mean centering** by computing their joint mean vector and subtracting it from each entity and relation embedding before similarity calculation.

$$\mu_E = \frac{1}{|\mathcal{E}| + |\mathcal{E}'|} \left(\sum_{i=1}^{|\mathcal{E}|} \mathbf{e}_i + \sum_{j=1}^{|\mathcal{E}'|} \mathbf{e}'_j \right), \quad \mathbf{e}_i \leftarrow \mathbf{e}_i - \mu_E, \quad \mathbf{e}'_j \leftarrow \mathbf{e}'_j - \mu_E. \tag{4}$$

We use a joint mean vector from both graphs to preserve the relative positions between embeddings across the two KGs while spreading them around the origin, thereby increasing the contrast of cosine similarity distributions during comparison.

5.1.3 Stacking Embedding Vectors to Matrix. We stack these embedding vectors to obtain matrices:

$$\mathbf{E} = \left[\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_{|\mathcal{E}|}\right], \quad \mathbf{E}' = \left[\mathbf{e}'_1, \mathbf{e}'_2, ..., \mathbf{e}'_{|\mathcal{E}'|}\right], \tag{5}$$

where $\mathbf{E} \in \mathbb{R}^{d\times|\mathcal{E}|}$ and $\mathbf{E}' \in \mathbb{R}^{d\times|\mathcal{E}'|}$ contain the entity embeddings of G and G', respectively. These matrices are used in the subsequent pairwise similarity computation.

5.1.4 Pairwise Similarity Calculation. We compute pairwise **cosine similarities** by first applying column-wise normalization to embedding matrices $\mathbf{E}$ and $\mathbf{E}'$ and then taking the dot product between the normalized embedding matrices.

$$\mathbf{Sim}^{\mathbf{E}} = \hat{\mathbf{E}}^\top \hat{\mathbf{E}}', \quad \hat{\mathbf{E}} = \mathbf{E}\,\mathrm{diag}(\mathbf{E}^\top\mathbf{E})^{-\frac{1}{2}}, \quad \hat{\mathbf{E}}' = \mathbf{E}'\,\mathrm{diag}(\mathbf{E}'^\top\mathbf{E}')^{-\frac{1}{2}}, \tag{6}$$

where $\mathrm{diag}(\cdot)$ extracts the diagonal elements of a square matrix and forms a diagonal matrix from them. Each value in the matrix $\mathbf{Sim}^{\mathbf{E}}$ is the cosine similarity between an entity from G and one from G'.

5.1.5 Handling Size Differences. When comparing KGs of different sizes, we measure how well the entities in the smaller KG are covered by entities in the larger KG. If similarity is instead aggregated from the larger KG to the smaller KG, multiple entities in the larger KG are inevitably matched to the same entity in the smaller KG, which can make additional information in the larger KG negatively affect the similarity score. For each entity in the smaller KG, we find its most similar entity in the larger KG by taking the maximum similarity value from $\mathbf{Sim}^{\mathbf{E}}$. Each row of $\mathbf{Sim}^{\mathbf{E}}$ corresponds to an entity in G and each column to one in G'. Selecting the maximum per row or column yields the closest match for entities in one graph to the other. Multiple entities can still share the same nearest matching entity.

$$S_V = \begin{cases} \{\max_{i\in\mathcal{E}},\ \mathbf{Sim}^{\mathbf{E}}_{i,j} | j \in \mathcal{E}'\}, & \text{if } |\mathcal{E}| > |\mathcal{E}'|, \\ \{\max_{j\in\mathcal{E}'}, \mathbf{Sim}^{\mathbf{E}}_{i,j} | i \in \mathcal{E}\}. & \text{otherwise.} \end{cases} \tag{7}$$

Here, $\mathbf{Sim}^{\mathbf{E}}_{i,j}$ denotes the similarity between the i-th entity in G and the j-th entity in G'. S_V collects the highest similarity scores for entities in the smaller KG to their best matches in the larger KG.

5.1.6 Computing the Final Similarity Score. Finally, the overall similarity between G and G' is defined as the fraction of scores s in S_V exceeding a threshold t:

$$S_{\text{pair}} = \frac{|\{s \in S_V | s > t\}|}{|S_V|}. \tag{8}$$

A higher S_{pair} indicates greater semantic similarity between two KGs.

5.2 Averaged Embedding-Based Similarity

EmbPairSim preserves the full information in KG embeddings but must compute and aggregate pair-wise similarities for all entity pairs, which becomes costly as the graph grows. Moreover, it does not produce a single representation of a KG. To obtain a single graph-level representation and avoid this quadratic overhead, we propose **Averaged Embedding-based Similarity (*AvgEmbSim*)**, which represents each graph by the frequency-weighted average of its embeddings; this summary vector provides a candidate proxy for the whole graph whose efficiency we will explicitly evaluate, and comparing weighted versus plain averages lets us test whether entity frequency matters in KGs as term frequency does in text.

5.2.1 Frequency-Weighted Averaging. To reflect the importance of each entity, we weight their embeddings by frequency, the number of triples in the knowledge graph that contain each entity, similar to word frequency in NLP. The frequency-weighted average embeddings are:

$$\bar{\mathbf{e}} = \frac{\sum_{e\in\mathcal{E}} \mathrm{freq}(e)\ \mathrm{Embed}(e)}{\sum_{e\in\mathcal{E}} \mathrm{freq}(e)}, \bar{\mathbf{e}'} = \frac{\sum_{e'\in\mathcal{E}'} \mathrm{freq}(e')\ \mathrm{Embed}(e')}{\sum_{e'\in\mathcal{E}'} \mathrm{freq}(e')}, \tag{9}$$

where $\mathrm{freq}(e)$ denotes the frequency of entity e.

5.2.2 Similarity Calculation. Finally, we compute the cosine similarity between the averaged entity embeddings of G and G':

$$S_{\text{avg}} = \frac{\bar{\mathbf{e}} \cdot \bar{\mathbf{e}'}}{||\bar{\mathbf{e}}||\ ||\bar{\mathbf{e}'}||}. \tag{10}$$

6 Experiments

6.1 Experimental Settings

6.1.1 Datasets & KG Extraction. We use two datasets: *WikiText-2* [15] (645 Wikipedia documents) and *CC-News* [14] (550 news articles). We treat each dataset as a separate document collection and construct $\mathcal{D}$, $\mathcal{D}'$, $\mathcal{G}$, and $\mathcal{G}'$ independently from each dataset. Knowledge Graphs (KGs) are extracted using the LLMGraphTransformer from LangChain [4] with GPT-3.5-turbo [19], yielding graph sets $\mathcal{G}$ and $\mathcal{G}'$. We use the default prompt and extraction pipeline provided by the LLMGraphTransformer without additional prompt engineering or task-specific modifications. All steps are performed independently for *WikiText-2* and *CC-News*.

6.1.2 Document Modification. (1) **Synonym Replacement**: words are randomly selected and replaced with synonyms from WordNet [18]. (2) **Context Replacement**: BERT [5] is used to select contextually important words, which are then replaced with synonyms from WordNet [18]. (3) **DIPPER Paraphraser**: generates diverse paraphrases with the DIPPER model [11]. We categorize the modified document sets by method: **Synonym**, **Context**, and **DIPPER**. For **Synonym** and **Context**, modification strengths **30** (%) and **60** (%) indicate the fraction of tokens changed. For **DIPPER**, we test **60/0** and **60/20**, where the format L/O denotes lexical diversity (L) and order diversity (O). These variations test the impact of different paraphrasing strategies on KG extraction and similarity. For each dataset, we randomly sample 200 documents and generate modified versions to form $\mathcal{D}'$.

6.1.3 Evaluation Metrics. We assess similarity performance using Hits@5, Mean Reciprocal Rank (MRR), and Normalized Discounted Cumulative Gain (NDCG). Hits@5 measures how often the correct KG appears in the top 5, MRR captures the average rank of the correct match, and NDCG evaluates ranking quality considering

both relevance and position. Together, these metrics indicate how well the scores reflect true semantic similarity.

6.1.4 Baselines. We use text-based, graph kernel-based, and KG embedding-based baselines to compare how well each approach captures semantic similarity. For the embedding-based baselines, including Word2Vec, FastText, and transductive KGE models, we mainly followed the standard/default hyperparameter settings provided in the original papers or official implementations. As a result, the transductive KGE models were evaluated under the same hyperparameter settings, including embedding dimension, margin, batch size, and learning rate.

- **Text-based Similarity.** Each KG is verbalized into OpenIE sentences (e.g., $\langle h, r, t\rangle \rightarrow$"*h r t*.") with NLTK [2], and graph similarity is computed from sentence embeddings given by the pretrained Sentence-BERT (SBERT) checkpoint `all-mpnet-base-v2` [21]. Moreover, in Table 5 we adopted two word embedding models, Word2Vec[16] and FastText[10]. For a non-pretrained setting, we train both models on the OpenIE sentences for 10 epochs, using 200-dimensional vectors, a window size of = 5, negative samples = 5, and min_count= 2 with uniform initialization for both models. In the pretrained configuration, we employ the 300-dimensional Google News Word2Vec embeddings [17], and the FastText embeddings trained on the Wikipedia 2017, UMBC WebBase, and News Crawl corpora [10].
- **Graph Kernel-based Similarity.** We use the Vertex Histogram (VH) and Weisfeiler-Lehman (WL) kernels [22, 24] implemented with GraKeL [23] to compare the graphs based on node and edge label distributions.
- **Transductive KG Embedding-based Similarity.** We used TransE [3], DistMult [31], ComplEx [26], and RotatE [25] trained on the combined graphs of all the compared graphs. For ComplEx and RotatE, we use the real-valued entity embedding vectors returned by the PyG implementation when computing cosine similarity. We use an embedding dimension of 32, a margin of 2, a batch size of 128, and a learning rate of 0.01 with early stopping. We used initial embeddings in their original papers—uniform initialization for TransE and RotatE, and Xavier-uniform initialization for DistMult and ComplEx. The threshold for *EmbPairSim* is determined via grid search and set to $t = 0.8$.
- **Inductive KG Embedding-based Similarity.** We use INGRAM [13] as an inductive KGE model. We adopt the pretrained model trained on NELL-995 provided by the official code of [13] with an embedding dimension of 32. The threshold for *EmbPairSim* is selected by grid search and set to $t = 0.95$. Unlike transductive models, INGRAM derives context solely from the structural patterns and does not reference individual entities encountered during training. Consequently, its performance indicates how much contextual information can be captured using structural patterns alone.

6.2 Results

6.2.1 Overall Performance Comparisons. The comparative results for the models outlined in Section 6.1.4 are presented in Table 1. As detailed in Section 6.2.2, RotatE achieves the best performance among all transductive embedding models in *EmbPairSim* and *AvgEmbSim*. The **Emb size** column confirms that low-dimensional KG embeddings are more parameter-efficient than language-model ones. On the other hand, *EmbPairSim*'s pairwise entity matching scales poorly on entity-dense sets like WikiText, whereas *AvgEmbSim* uses a single graph embedding and is therefore much more efficient than SBERT or *EmbPairSim*.

The Role of Semantic Information in Enhancing Similarity Detection. EmbPairSim outperforms both graph kernels, highlighting the advantage of semantic embeddings over purely statistical structural methods. This suggests that incorporating semantic information significantly improves graph similarity detection.

Comparative Performance of EmbPairSim and SBERT. EmbPairSim achieves the highest overall performance, even surpassing SBERT in most cases. This shows that KG embeddings capture semantic meaning more effectively than text-based embeddings. However, under the **DIPPER** paraphrasing setting, SBERT performs better, indicating a limitation of *EmbPairSim* when handling substantial lexical and syntactic changes.

Effectiveness of Structural Information in Inductive Models. EmbPairSim and *AvgEmbSim* perform similarly on both RotatE and INGRAM. Despite INGRAM's fully inductive setting, its performance is comparable to transductive models, suggesting that structural information is sufficient for capturing semantic meaning in KGs. However, although INGRAM achieves almost the same *EmbPairSim* performance as RotatE, it requires a substantially higher similarity threshold (0.95 versus 0.80 for RotatE), suggesting that INGRAM produces entity embeddings with generally higher cosine similarities and less dispersion in the embedding space than RotatE.

Performance Gap between Vertex Histogram and Weisfeiler-Lehman Kernels. The Vertex Histogram (VH) kernel slightly outperforms the Weisfeiler-Lehman (WL) Kernel. This appears to stem from WL's relabeling stage, which merges entity and relation labels, thereby blurring their distinction. Supporting this, we observed that WL using either the Vertex or Edge Histogram as its base yields identical results. Similarly, *EmbPairSim*–based solely on entity embeddings–outperforms models that mix entity and relation information, highlighting the importance of preserving entity-level signals in semantic similarity tasks.

Limitations under Structural Paraphrasing. While *EmbPairSim* and *AvgEmbSim* generally outperform text-based baselines, they show clear limitations when faced with substantial lexical and structural variations. The most notable performance drop is observed with the **DIPPER** Paraphraser, which introduces both large-scale vocabulary changes and global sentence reordering. Unlike the more localized word substitutions in the **Synonym** and **Context** settings, these changes disrupt surface patterns while preserving underlying semantics. This degradation stems from the fact that both *EmbPairSim* and *AvgEmbSim* represent KGs through local statistics without capturing the graph's relational structure or topology at a higher level. Although KGE-based methods consistently outperform SBERT in overall accuracy, their reliance on simple aggregation limits their ability to detect semantic alignment when meaning is preserved but surface forms are rearranged. Modeling higher-order relational structure and graph-level semantics may

Table 1: Overall performance (Hits@5, MRR, and NDCG) on CC News and WikiText for text-, kernel-, and KGE-based methods under Synonym, Context, and DIPPER paraphrasing strengths. Seen means that the method directly accesses (or was trained with) the evaluation KG triples. For transductive KGE model, we use RotatE and for inductive KGE model, we use INGRAM. Emb size denotes the total parameter footprint used during similarity computation, calculated as *(number of stored embedding vectors)* × *(embedding dimension)* across all original KGs in the corresponding dataset. Since VH Kernel and WL Kernel do not generate embedding vectors, we mark their Emb size with "N/A". In the Seen column, ○ indicates that the method directly accesses or is trained on the evaluation KG triples, whereas × indicates that the evaluation KG triples are unseen during training.

Metric	Method	Seen	CC News							WikiText						
			Emb size	Synonym		Context		DIPPER		Emb size	Synonym		Context		DIPPER	
				30	60	30	60	60/0	60/20		30	60	30	60	60/0	60/20
Hits@5	VH Kernel	○	N/A	0.970	0.965	0.955	0.950	0.850	0.870	N/A	0.990	0.960	0.975	0.955	0.935	0.965
	WL Kernel	○	N/A	0.920	0.930	0.930	0.900	0.805	0.815	N/A	0.980	0.965	0.960	0.940	0.905	0.945
	SBERT	×	422,400	0.970	0.965	0.960	0.965	**0.920**	**0.945**	495,360	0.955	0.955	0.985	0.965	0.940	0.940
	EmbPairSim (RotatE)	○	188,928	**0.975**	**0.980**	**0.970**	**0.970**	0.875	0.915	779,008	**0.995**	**0.980**	**0.990**	**0.970**	**0.955**	**0.975**
	EmbPairSim (INGRAM)	×	188,928	**0.975**	**0.980**	0.965	**0.970**	0.875	0.915	779,008	**0.995**	**0.980**	**0.990**	**0.970**	**0.955**	**0.975**
	AvgEmbSim (RotatE)	○	17,600	0.800	0.710	0.800	0.695	0.510	0.495	20,640	0.750	0.400	0.750	0.585	0.750	0.357
	AvgEmbSim (INGRAM)	×	17,600	0.790	0.705	0.745	0.585	0.410	0.445	20,640	0.675	0.565	0.755	0.670	0.400	0.365
MRR	VH Kernel	○	N/A	**0.942**	0.940	0.926	0.890	0.786	0.804	N/A	0.970	0.904	0.947	0.913	0.858	0.867
	WL Kernel	○	N/A	0.881	0.876	0.885	0.821	0.717	0.729	N/A	0.960	0.914	0.931	0.902	0.812	0.869
	SBERT	×	422,400	0.925	0.911	0.920	0.924	**0.840**	**0.853**	495,360	0.935	0.909	0.963	0.924	**0.909**	**0.901**
	EmbPairSim (RotatE)	○	188,928	**0.942**	**0.953**	**0.928**	**0.928**	0.830	0.834	779,008	**0.988**	**0.952**	**0.975**	**0.945**	0.887	0.897
	EmbPairSim (INGRAM)	×	188,928	**0.942**	0.946	**0.928**	0.889	0.827	0.833	779,008	**0.988**	**0.945**	**0.973**	**0.945**	0.879	0.889
	AvgEmbSim (RotatE)	○	17,600	0.751	0.655	0.739	0.625	0.438	0.437	20,640	0.698	0.314	0.748	0.543	0.492	0.314
	AvgEmbSim (INGRAM)	×	17,600	0.731	0.650	0.703	0.526	0.362	0.383	20,640	0.620	0.461	0.667	0.601	0.345	0.300
NDCG	VH Kernel	○	N/A	0.953	0.952	0.940	0.912	0.824	0.835	N/A	0.978	0.923	0.960	0.930	0.886	0.896
	WL Kernel	○	N/A	0.905	0.903	0.908	0.858	0.771	0.777	N/A	0.969	0.930	0.946	0.921	0.850	0.895
	SBERT	×	422,400	0.943	0.931	0.938	0.942	**0.876**	**0.887**	495,360	0.949	0.928	0.971	0.941	**0.928**	**0.923**
	EmbPairSim (RotatE)	○	188,928	**0.954**	**0.963**	**0.943**	**0.949**	0.861	0.861	779,008	**0.991**	**0.960**	**0.981**	**0.954**	0.907	0.918
	EmbPairSim (INGRAM)	×	188,928	**0.954**	0.957	**0.943**	0.913	0.857	0.859	779,008	**0.991**	0.954	**0.980**	**0.954**	0.900	0.912
	AvgEmbSim (RotatE)	○	17,600	0.794	0.708	0.784	0.686	0.519	0.518	20,640	0.748	0.410	0.835	0.606	0.566	0.398
	AvgEmbSim (INGRAM)	×	17,600	0.782	0.715	0.759	0.611	0.469	0.483	20,640	0.689	0.559	0.729	0.672	0.451	0.411

therefore provide a more robust way to handle substantial paraphrastic transformations. As a result, they remain vulnerable to paraphrastic variations that obscure local signals while maintaining global intent.

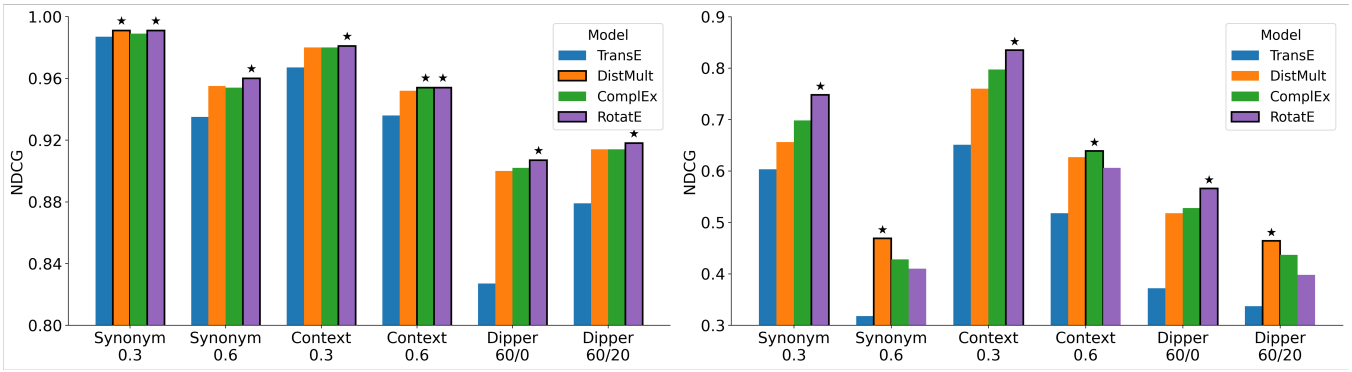
6.2.2 Selecting KGE Models for EmbPairSim and AvgEmbSim. We evaluate four transductive KG embedding models—TransE, DistMult, ComplEx, and RotatE—as back-ends for *EmbPairSim* and *AvgEmbSim*, and summarize their NDCG performance across different paraphrasing settings in Figure 2. RotatE consistently delivers the highest scores, plausibly because its representation of relations as rotations in complex space aligns with the cosine similarity intrinsic to both metrics, retaining angular semantics. TransE performs worst; its purely translational assumption fails to provide the fine-grained relational separation required for reliable similarity estimation. DistMult and ComplEx behave similarly under *EmbPairSim* due to their shared bilinear form, yet ComplEx slightly outperforms DistMult in *AvgEmbSim*, reflecting its capacity to model asymmetric relations. Overall, models with multiplicative or angular inductive biases (RotatE, ComplEx) produce more effective embeddings for both pairwise and averaged similarity calculations.

Table 2: Ablation study on *AvgEmbSim* using the RotatE model on WikiText-2, comparing setting with (w/ freq) and without (w/o freq) entity frequency information.

Metric	Option	Synonym		Context		DIPPER	
		30	60	30	60	60/0	60/20
Hits@5	w/ freq	**0.750**	0.400	**0.750**	**0.585**	**0.750**	**0.357**
	w/o freq	0.675	**0.595**	0.655	0.560	0.385	0.320
MRR	w/ freq	**0.698**	0.314	**0.748**	**0.543**	**0.492**	**0.314**
	w/o freq	0.600	**0.524**	0.609	0.470	0.329	0.268
NDCG	w/ freq	**0.748**	0.410	**0.835**	**0.606**	**0.566**	**0.398**
	w/o freq	0.666	**0.597**	0.672	0.556	0.429	0.378

6.2.3 Ablation Study.

Frequency of Elements Influences the Semantic Representation of a Knowledge Graph. As part of the ablation study, we weight each embedding by its occurrence frequency instead of computing a simple average. Table 2 shows that this frequency-weighted approach generally improves performance. These findings suggest that the frequency of entities plays a crucial role in capturing the overall



(a) EmbPairSim: NDCG results on WikiText-2

(b) AvgEmbSim: NDCG results on WikiText-2

Figure 2: NDCG scores of *EmbPairSim* and *AvgEmbSim* on *WikiText* using different transductive KGE models across all modification settings. For each setting, the highest NDCG score is marked with a black outline and a star.

Table 3: Ablation study on *EmbPairSim* using the INGRAM model on WikiText-2, comparing setting with (w/ cent) and without (w/o cent) mean centering.

Metric	Option	Synonym		Context		DIPPER	
		30	60	30	60	60/0	60/20
Hits@5	w/ cent	**0.995**	**0.980**	**0.990**	**0.970**	**0.955**	**0.975**
	w/o cent	0.000	0.000	0.000	0.000	0.000	0.000
MRR	w/ cent	**0.988**	**0.945**	**0.973**	**0.945**	**0.879**	**0.889**
	w/o cent	0.002	0.002	0.002	0.002	0.002	0.002
NDCG	w/ cent	**0.991**	**0.954**	**0.980**	**0.954**	**0.900**	**0.912**
	w/o cent	0.108	0.108	0.109	0.108	0.108	0.109

semantic information of a KG, similar to how word frequency aids in understanding text documents.

Mean-Centering Enhances Entity-Level Similarity. Overall, mean-centering has little impact when *EmbPairSim* employs transductive KGE models. However, Table 3 shows that without mean-centering, the INGRAM-based *EmbPairSim* fails to distinguish graphs. This suggests that INGRAM produces entity vectors that are compressed into a narrow subspace. This behavior may arise because INGRAM is pretrained on a single large KG. Such pretraining captures structural patterns in the source graph, but when applied to the much smaller KGs in our experiments, it can map many entities into a narrow region of the embedding space. Mean-centering alleviates this collapse by removing the global bias in the embedding space, making relative differences between entity vectors more visible for cosine-based matching.

6.2.4 Relation Embedding Considerations. Extending from prior experiments based solely on entity embeddings, we observe that including relation embeddings degrades similarity performance rather than improving it. For *EmbPairSim*, we compute pairwise cosine similarities between relation embeddings in the same way as entities (Eq. (6)), and include the result in the final score (Eq. (8)). For *AvgEmbSim*, we calculate the frequency-weighted mean vector for relations (Eq. (9)) and add its cosine similarity to that of the entity vectors (Eq. (10)). As shown in Table 4, incorporating relation embeddings consistently degraded performance. This suggests

Table 4: Comparison of *EmbPairSim* (*pair*) and *AvgEmbSim* (*avg*) on WikiText-2 (RotatE embeddings). w/o rel: entity-only; w/ rel: entity + relation embeddings.

Metric	Option	Method	Synonym		Context		Dipper	
			30	60	30	60	60/0	60/20
Hits@5	w/o rel	pair	**0.995**	**0.980**	**0.990**	**0.970**	**0.955**	**0.975**
	w/ rel	pair	**0.995**	0.955	**0.990**	0.955	0.910	0.925
MRR	w/o rel	pair	**0.988**	**0.952**	**0.975**	**0.945**	**0.887**	**0.897**
	w/ rel	pair	**0.988**	0.921	0.973	0.905	0.811	0.816
NDCG	w/o rel	pair	**0.991**	**0.960**	**0.981**	**0.954**	**0.907**	**0.918**
	w/ rel	pair	**0.991**	0.936	0.980	0.924	0.850	0.856
Hits@5	w/o rel	avg	**0.750**	**0.400**	**0.750**	**0.585**	**0.750**	**0.357**
	w/ rel	avg	0.250	0.180	0.330	0.190	0.145	0.095
MRR	w/o rel	avg	**0.698**	**0.314**	**0.748**	**0.543**	**0.492**	**0.314**
	w/ rel	avg	0.229	0.170	0.282	0.180	0.122	0.096
NDCG	w/o rel	avg	**0.748**	**0.410**	**0.835**	**0.606**	**0.566**	**0.398**
	w/ rel	avg	0.345	0.290	0.392	0.298	0.247	0.219

that relation vectors may introduce noise rather than helpful signals, likely because semantic information from relations is already implicitly captured during entity embedding training.

6.2.5 Comparing KG Embeddings with Word Embeddings. Because KG embeddings act as distributional representations, we can likewise apply *EmbPairSim* by representing entities with word embeddings. Table 5 contrasts *EmbPairSim* when entities are represented by KG embeddings versus word embeddings. The results show that KGE-based embeddings are more effective to capture KG-to-KG semantic similarity than word embeddings. This suggests that explicitly encoding entities through relational structure is more suitable for comparing KGs than relying on word embeddings learned from sequential textual contexts. A likely reason is that KGE directly exploits graph-structured relations among entities, whereas word embeddings capture relational signals indirectly from sequential co-occurrence.

6.2.6 Runtime Efficiency. Table 6 reports the runtime of each graph-to-graph similarity method on WikiText-2 under the synonym replacement setting. Among the baselines, VH Kernel and WL Kernel require 1.569 and 3.770 seconds, respectively, while SBERT requires

Table 5: EmbPairSim performance on WikiText-2 using Word2Vec and FastText embeddings. In the Pretrained column, ○ indicates that externally pretrained embeddings are used, whereas × indicates that the embeddings are trained only on the WikiText-2 corpus.

Metric	Model	Pretrained	Synonym		Context		Dipper	
			30	**60**	**30**	**60**	**60/0**	**60/20**
Hits@5	FastText	×	0.990	0.940	0.985	0.955	0.915	0.955
	Word2Vec	×	0.680	0.445	0.635	0.495	0.300	0.285
	RotatE	×	**0.995**	**0.980**	**0.990**	**0.970**	**0.955**	**0.975**
MRR	FastText	×	0.972	0.865	0.957	0.903	0.778	0.793
	Word2Vec	×	0.399	0.258	0.375	0.344	0.170	0.186
	RotatE	×	**0.988**	**0.952**	**0.975**	**0.945**	**0.887**	**0.897**
NDCG	FastText	×	0.979	0.893	0.968	0.922	0.823	0.839
	Word2Vec	×	0.531	0.399	0.510	0.477	0.321	0.334
	RotatE	×	**0.991**	**0.960**	**0.981**	**0.954**	**0.907**	**0.918**
Hits@5	FastText	○	**0.995**	0.975	0.975	0.965	0.950	0.965
	Word2Vec	○	0.755	0.530	0.695	0.570	0.575	0.590
	INGRAM	○	**0.995**	**0.980**	**0.990**	**0.970**	**0.955**	**0.975**
MRR	FastText	○	0.974	0.925	0.959	0.921	0.847	0.870
	Word2Vec	○	0.592	0.381	0.523	0.450	0.434	0.413
	INGRAM	○	**0.988**	**0.945**	**0.973**	**0.945**	**0.879**	**0.889**
NDCG	FastText	○	0.980	0.940	0.968	0.936	0.876	0.897
	Word2Vec	○	0.680	0.500	0.626	0.557	0.541	0.528
	INGRAM	○	**0.991**	**0.954**	**0.980**	**0.954**	**0.900**	**0.912**

Table 6: Runtime comparison of graph-to-graph similarity methods.

Method	Runtime (s)
VH Kernel	1.569
WL Kernel	3.770
SBERT	12.374
EmbPairSim (RotatE)	0.381
AvgEmbSim (RotatE)	0.228

12.374 seconds due to sentence-level encoding over verbalized KG triples. In contrast, the KGE-based scoring functions show substantially lower runtime. EmbPairSim with RotatE completes the similarity computation in 0.381 seconds, and AvgEmbSim with RotatE further reduces the runtime to 0.228 seconds. This result supports the efficiency of the proposed KGE-based scoring functions, especially AvgEmbSim, which represents each KG with a single frequency-weighted centroid instead of computing pairwise entity similarities. Together with the performance results in Table 1, these runtime results indicate that KGE-based graph-to-graph similarity can provide an efficient alternative to text-based and kernel-based methods.

7 Conclusion

We presented a systematic approach to compute semantic similarity between KGs using SBERT, graph kernels, and KGE-based methods. To leverage KGE more effectively, we proposed two complementary functions, *EmbPairSim* and *AvgEmbSim*. Experimental results show that KGE can capture semantic information, supporting the view that a KG can be interpreted hierarchically like a document, with entities and relations as words and triples and subgraphs forming higher-level meaning. Future work should explore end-to-end graph neural architectures that better encode structural patterns and relational dependencies for richer graph-level semantics.

References

[1] Zeina Abu-Aisheh, Romain Raveaux, Jean-Yves Ramel, and Patrick Martineau. 2015. An Exact Graph Edit Distance Algorithm for Solving Pattern Recognition Problems. In *ICPRAM*.

[2] Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*.

[3] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*.

[4] Harrison Chase and the LangChain contributors. 2025. LangChain: A framework for building applications with large language models (0.3.26).

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*.

[6] Miao Fan, Qiang Zhou, Thomas Fang Zheng, and Ralph Grishman. 2017. Distributed representation learning for knowledge graphs with entity descriptions. *Pattern Recognition Letters* (2017).

[7] Erwin Filtz. 2017. Building and Processing a Knowledge-Graph for Legal Data. In *The Semantic Web*.

[8] Saiping Guan, Xueqi Cheng, Long Bai, Fujun Zhang, Zixuan Li, Yutao Zeng, Xiaolong Jin, and Jiafeng Guo. 2023. What is Event Knowledge Graph: A Survey. *IEEE Transactions on Knowledge and Data Engineering* (2023).

[9] Mohamad Yaser Jaradeh, Allard Oelen, Kheir Eddine Farfar, Manuel Prinz, Jennifer D'Souza, Gábor Kismihók, Markus Stocker, and Sören Auer. 2019. Open Research Knowledge Graph: Next Generation Infrastructure for Semantic Scholarly Knowledge. In *K-CAP*.

[10] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of Tricks for Efficient Text Classification. In *EACL*.

[11] Kalpesh Krishna, Yixiao Song, Mohit Iyyer, Regina Barzilay, and Daniel Khashabi. 2023. Paraphrasing evades detectors of AI-generated text, but retrieval is an effective defense. In *NeurIPS*.

[12] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Fei-Fei Li. 2017. Visual Genome: Connecting Language and Vision Using Crowdsourced Dense Image Annotations. *IJCV* (2017).

[13] Jaejun Lee, Chanyoung Chung, and Joyce Jiyoung Whang. 2023. InGram: Inductive Knowledge Graph Embedding via Relation Graphs. In *ICML*.

[14] Joel Mackenzie, Rodger Benham, Matthias Petri, Johanne R. Trippas, J. Shane Culpepper, and Alistair Moffat. 2020. CC-News-En: A Large English News Corpus. In *CIKM*.

[15] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer Sentinel Mixture Models. In *ICLR*.

[16] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *ICLR Workshop*.

[17] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *NIPS*.

[18] George A. Miller. 1995. WordNet: a lexical database for English. *Commun. ACM* (1995).

[19] OpenAI. 2023. GPT-3.5-turbo API.

[20] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *EMNLP-IJCNLP*.

[21] Nils Reimers and Iryna Gurevych. 2021. all-mpnet-base-v2: A Sentence-BERT Model. https://huggingface.co/sentence-transformers/all-mpnet-base-v2.

[22] Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M. Borgwardt. 2011. Weisfeiler-Lehman Graph Kernels. *Journal of Machine Learning Research* (2011).

[23] Giannis Siglidis, Giannis Nikolentzos, Yannis Limnios, Christos Giatsidis, and Michalis Vazirgiannis. 2020. GraKeL: A Graph Kernel Library in Python. *Journal of Machine Learning Research* (2020).

[24] Mahito Sugiyama and Karsten Borgwardt. 2015. Halting in Random Walk Kernels. In *NeurIPS*.

[25] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. In *ICLR*.

[26] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex Embeddings for Simple Link Prediction. In *ICML*.

[27] J. R. Ullmann. 1976. An Algorithm for Subgraph Isomorphism. *J. ACM* (1976).

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Neurips*. https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[29] Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM* 57, 10 (2014), 78–85.

[30] 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. In *ACL*.
[31] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In *ICLR*.
[32] Likun Yao, Chao Mao, and Yuanfang Luo. 2019. KG-BERT: BERT for Knowledge Graph Completion. In *EMNLP*.
[33] Yuan Zhang, Xin Sui, Feng Pan, Kaixian Yu, Keqiao Li, Shubo Tian, and Jinfeng Zhang. 2025. A comprehensive large scale biomedical knowledge graph for AI powered data driven biomedical research. *Nature Machine Intelligence* (2025).

A Examples of KG Information at Different Granularities

In scholarly KGs like ORKG [9], contribution-centered subgraphs represent contributions and are used to compare related literature, and the entire KG represents a scholarly knowledge base and is used to support research comparison and thematic review across papers or research areas, while entities and relations represent scholarly units and their connections and triples encode structured scholarly statements. In event-centric KGs like EventKG [8], event-centered subgraphs represent event contexts and are used to support event exploration and timeline generation, and the entire KG represents a temporal event knowledge base and is used for event-centric analysis, while entities and relations represent events, participants, locations, times, and temporal or semantic connections and triples encode individual event facts. In scene graphs [12], region-level subgraphs represent image regions and are used for region-level visual understanding, and the entire KG represents image-level visual content and is used for image-level visual understanding, while entities and relations represent visual objects, attributes, and interactions and triples encode local visual relationships. In encyclopedic KGs like Wikidata [29], item- or topic-centered subgraphs represent item-level knowledge and are used to retrieve item information, statements, and provenance, and the entire KG represents an encyclopedic knowledge base and is used for structured search over entities and attributes across domains, while entities and relations represent encyclopedic units and their properties and triples correspond to item-property-value statements. In biomedical KGs like BioKG [33], disease-, drug-, or pathway-centered subgraphs represent biomedical contexts and are used to trace relations among diseases, drugs, genes, and pathways for biomedical knowledge discovery, and the entire KG represents a biomedical knowledge base and is used for causal inference, drug repurposing, and drug target identification, while entities and relations represent biomedical units and their associations and triples encode individual biomedical facts. In legal KGs like LegalKG [7], case- or regulation-centered subgraphs represent legal contexts and are used to trace dependencies between legal provisions and judicial decisions, and the entire KG represents a legal knowledge base and is used to search, connect, and process legal norms and court decisions, while entities and relations represent legal units and their dependencies and triples encode individual legal statements.