An Examination of Embedding Methods for Entity Comparison in Text-Rich Knowledge Graphs

Huan Chen[1,3,*,†], Gareth J. F. Jones[2,3,*,†] and Rob Brennan[1,3,*,†]

[1] *University College Dublin, Dublin, Ireland*
[2] *Dublin City University, Dublin, Ireland*
[3] *ADAPT SFI Research Centre, Dublin, Ireland*

Abstract

We report a comparative analysis of entity comparison within text-rich, domain-specific knowledge graphs which evaluates the performance of various embedding techniques: graph-based embedding, traditional word embedding, contextual embedding, and combined embedding. This work is motivated by the shortcomings of current symbolic graph querying approaches for knowledge graphs that are rich in textual attributes or linked documents. Current approaches can struggle to represent the semantics or similarity of context for text-based entity attributes when comparing entities in knowledge graphs. This study demonstrates that models like BERT and all-MiniLM-L6-v2, which leverage contextual embeddings, are significantly more effective than static embeddings like Word2Vec for both well-defined and structured short phrases (such as those modelled by OWL classes or SKOS concepts in knowledge graphs) and free text in entity comparison tasks for the entities in our initial study within a domain-specific text-rich knowledge graph. On the other hand, we find that combining Word2Vec with all-MiniLM-L6-v2 achieves better performance than combining BERT with all-MiniLM-L6-v2, despite BERT outperforming Word2Vec when used individually. Furthermore, our study highlights the importance of splitting entity attributes based on their characteristics, distinguishing between structured short phrases and free-text attributes to apply the most suitable embedding technique for each. By identifying the optimal embedding techniques for different characteristics of text data, we provide valuable insights into improving entity comparison accuracy in GRC domain text-rich knowledge graphs.

Keywords

Knowledge Graph, Entity Comparison, Embedding Techniques, Neuro-symbolic AI

1. Introduction

The size and scope of knowledge graphs (KGs) are expanding to incorporate rich information stores that include unstructured data, linked documents or other multimodal entity attributes[1]. For instance, DBpedia and Microsoft academic graph. For these heterogeneous, text-rich or multimodal graphs, graph languages such as SPARQL often struggle to deliver accurate and pertinent results. One of the primary challenges with SPARQL queries is their limited ability to deliver context-aware results, which restricts their ability to capture the nuanced relationships between entities. Additionally, querying entity attributes that include links or external references presents difficulties, since SPARQL lacks robust mechanisms to handle these complex structures. In this environment, studying techniques to enable the comparison of text-rich KG entities has become a crucial subject.

KG embeddings are employed for many similarity-based tasks, such as entity and relation similarity, however, KG embedding models focus on the transformation of entities into vectors but have limited ability to capture associated context information. This limitation affects entity comparison, especially in text-rich KGs. Text embeddings are better at capturing language characteristics such as semantic meaning, syntactic structure, and contextual usage [2], which allows them to learn textual similarities more effectively.

AICS'24: 32nd Irish Conference on Artificial Intelligence and Cognitive Science, December 09–10, 2024, Dublin, Ireland
[*] Corresponding author.
[†] These authors contributed equally.
huan.chen2@ucdconnect.ie (H. Chen); Gareth.jones@dcu.ie (G. J. F. Jones); rob.brennan@ucd.ie (R. Brennan)
0009-0007-8549-0414 (H. Chen); 0000-0003-2923-8365 (G. J. F. Jones); 0000-0001-8236-362X (R. Brennan)

However, there are a variety of text embeddings, such as BERT[3] generates dynamic, context-aware embeddings using a deep bidirectional transformer model, which captures the meaning of words based on their surrounding context[3] or all-MiniLM-L6-v2 (MiniLM)[1] is used for generating sentence-level embeddings, enabling us to capture the broader meaning of sentences in a computationally efficient way. Since entity attributes in KGs are diverse (e.g., text descriptions, labels, metadata), it is crucial to apply the appropriate embedding methods to different types of data to maximize the accuracy and effectiveness of entity comparison. In this work, we explore the use of different entity attributes (e.g., SKOS concepts, metadata, values) in text-rich KGs for entity comparison. Specifically, we are interested in examining how different embedding methods enhance the accuracy of entity comparison within text-rich KGs. We study an example of text-rich KGs created by a platform for the GRC (Governance, Risk and Compliance) domain by utilizing the graph from the platform describing risk projects. The KGs in the platform are used to semantically link heterogeneous risk, and safety improvement projects modelled in OWL2. They feature natural language tagged with terminologies defined as SKOS concepts and linked evidence data such as documents, spreadsheets and arbitrary web content[4]. We have made public the source code via a GitHub link [2], due to privacy concerns, the dataset cannot be made available.

The contributions of this work are:

- Investigate methods for entity comparison in KGs by focusing on diverse existing approaches for comparing entities within knowledge graphs.
- Explore the partition of different KG entity attributes based on their characters and apply alternative embedding techniques to improve entity comparison in text-rich KGs, particularly in the GRC domain.
- Investigate the integration of various embedding techniques to identify opportunities for enhancing the accuracy of entity comparison in text-rich KGs.

2. Case Study - Text Rich Knowledge Graphs in a Governance Risk and Compliance (GRC) Platform

The Governance, Risk, and Compliance (GRC) domain involves managing interconnected datasets that support organizational safety, risk assessment, and regulatory compliance. These datasets often include structured and unstructured data such as taxonomies, narrative fields, and linked evidence[5]. The ARK (Access Risk Knowledge) platform [3] is a GRC application built on Semantic Web technology. It facilitates expert analysis of 'safety projects,' such as virus risk assessments or cybersecurity projects. The project consists of phases, each of which has a project analysis concept which includes questionnaire responses made up of symbols (e.g. date, staff involved, priority, dimension of analysis) and narrative analysis in natural language annotated by domain concepts (drawn from taxonomies specified using the Simple Knowledge Organisation System[6]). Analysts can also link or reference evidence (datasets, reports, standards, presentations, email correspondence, etc.) to support their analysis. Additional context and synthesis can be added via linked risk assessments and a realist evaluation which consists of more narrative (text) fields, annotations and evidence.

The GRC platform curates knowledge graph-based models of safety improvement projects. The graph is rich in textual data whose nodes include summaries, links to other sources of information, or descriptive text. An example of a text-rich KG fragment describing a safety project in the GRC tool is shown in Figure 1. It shows the use of natural language, SKOS concept annotations, OWL classes and properties and linked files within one knowledge graph.

Each safety improvement project has a life cycle and there are related projects, and project hierarchies (i.e. derived projects or synthesis projects that draw on the results of many other projects). Within an

[1]https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2
[2]https://github.com/Foina/multi-embeddings-entity-comparison
[3]https://openark.adaptcentre.ie/

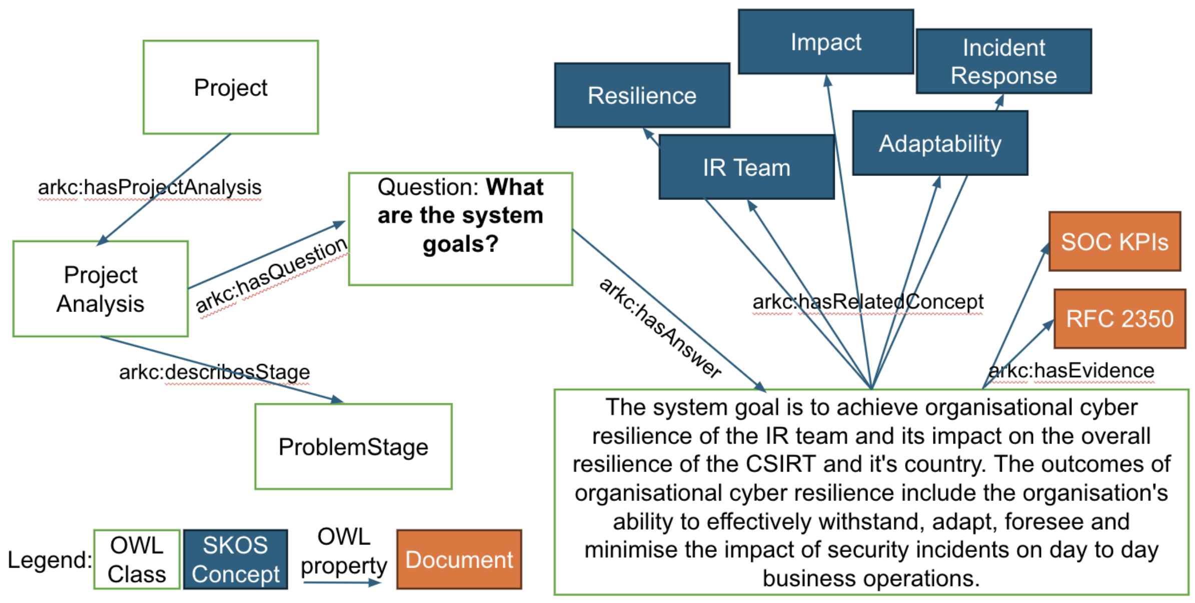
Project
arkc:hasProjectAnalysis
Project Analysis
arkc:hasQuestion
Question: What are the system goals?
arkc:describesStage
ProblemStage
Resilience
Impact
Incident Response
IR Team
Adaptability
arkc:hasAnswer
arkc:hasRelatedConcept
SOC KPIs
RFC 2350
arkc:hasEvidence
The system goal is to achieve organisational cyber resilience of the IR team and its impact on the overall resilience of the CSIRT and it's country. The outcomes of organisational cyber resilience include the organisation's ability to effectively withstand, adapt, foresee and minimise the impact of security incidents on day to day business operations.
Legend:
OWL Class
SKOS Concept
OWL property
Document

Figure 1: Example text-rich knowledge graph in GRC tool showing natural language, key SKOS concepts and OWL classes. For clarity, most properties/graph edges were omitted.

organisation, there are many aspects of projects that repeat over time, e.g. addressing similar issues, involving the same staff, targeting the same risk or operational process, using the same evidence. Project governance often requires finding similar projects and comparing them, synthesizing the results or identifying best practices from frequent similarities. In addition, project data entry is laborious so recommending similar projects to copy data from is useful. Traditional graph query methods lack the ability to query the semantics of the text that makes up so much of the project data. Hence, in this research, we focus on evaluating new ways to compare these text-rich knowledge graph project entities.

3. Related Work

Research on techniques for comparing similarities between entities has garnered increasing attention in the last few years. Existing techniques can be divided into categories including query-based techniques[7, 8, 9], graph embedding approaches[10, 11], semantic similarity methods[12, 13].

Query-based approaches, using SPARQL queries to compare entities involve comparing features such as the properties of entities, and entity attributes. A similarity query computation method is proposed in [7], this can be employed for large KGs and expands SPARQL similarity queries by supporting numeric filter expressions. A framework for modelling comparison using both the most general difference queries (MGDQs) and the most specific similarity queries (MSSQs) was presented in [8]. However, using SPARQL queries to compare entities is limited in understanding the semantics of attribute values and lacks mechanisms to capture the context of attributes. For instance, SPARQL cannot distinguish "apple" as a company from "apple" as a type of fruit. Furthermore, SPARQL can not handle synonymy or polysemy.

By transforming entities into vectors, graph embedding approaches enable similarity metrics to be used to compute the similarity between entities. The matching of entities across different KGs is studied in [10], specifically, this explores the similarity between DBpedia and Wikidata. They found that a simple multi-layer perceptron based on representations derived from RDF2VEC graph embeddings was effective. However, graph-embedding approaches for entity comparison tasks limit the capture of the context of attributes, especially textual descriptions of attributes, since they focus on the graph structure.

Semantic textual similarity methods, by utilizing natural language processing and machine learning

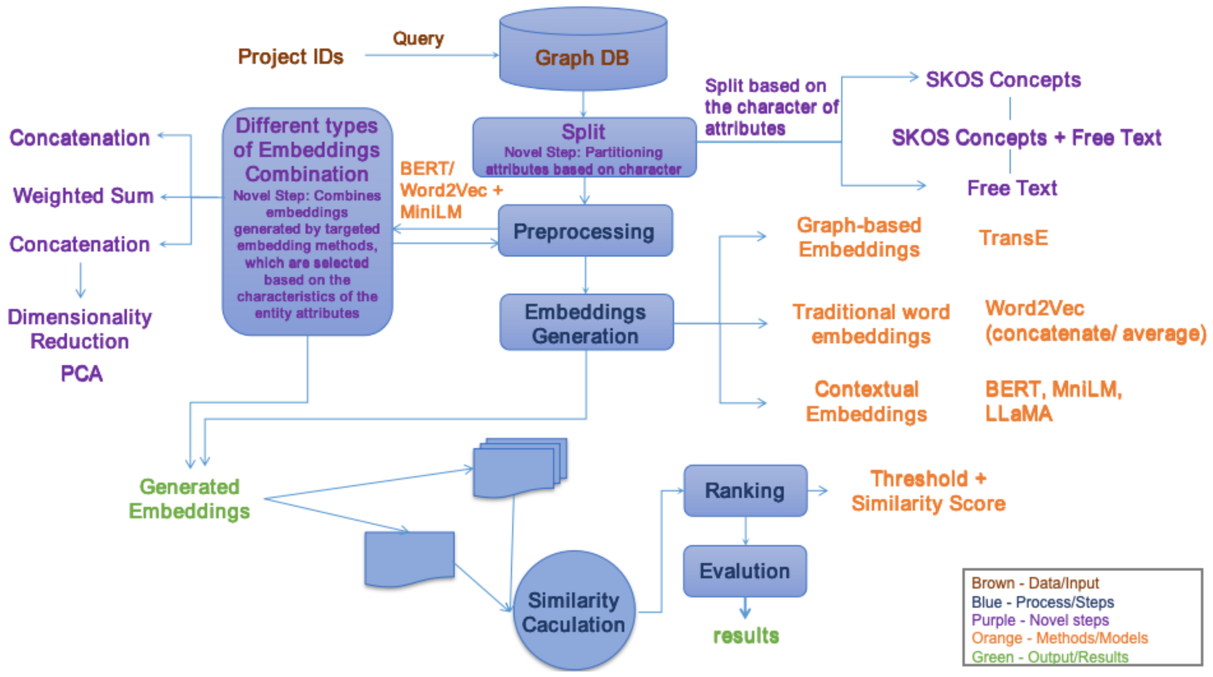

Figure 2: Embedding-Based Text-Rich Knowledge Graph Entity Comparison and Ranking Pipeline

to measure how similar pieces of text are. Precise assessment of semantic textual similarity between words is crucial for tasks such as document clustering, and information retrieval [14]. To enhance entity comparison in text-rich knowledge graphs, it is important to integrate semantic textual similarity approaches as part of the solution. By employing a variety of text embeddings—each focusing on different features, such as word-level and sentence-level representations—we can effectively leverage the textual information in attributes. For instance, Word2Vec works particularly well at the word level by generating static word embeddings, where each word is represented as a fixed vector in a continuous vector space[15]. It effectively captures the semantic relationships between words based on their co-occurrence patterns[15]. However, its inability to distinguish between distinct word meanings in different contexts is due to its static nature. Moving beyond word-level embeddings, BERT enables a deep understanding of both semantic and syntactic structures, making it highly effective for tasks requiring context-aware similarity. However, BERT is not specifically pretrained for sentence similarity tasks, meaning its embeddings often require fine-tuning or additional frameworks to achieve strong performance in sentence-level similarity tasks. Furthermore, BERT is computationally expensive and has a large model size, which makes it less suitable for resource-constrained or real-time environments. In scenarios where computational efficiency is crucial, MiniLM offers a streamlined alternative. It excels at generating sentence-level embeddings, making it well-suited for tasks such as sentence similarity. However, as a distilled model, MiniLM may struggle to capture the nuanced meanings of rare or domain-specific words. In contrast, LLaMA[16] is a unidirectional model that generates embeddings based on the sequential context of words, enabling it to capture contextual relationships dynamically as it processes text. However, like other large language models, it demands significant computational resources and may be best suited for generative tasks.

4. Methodology

This section presents a new methodology used for conducting entity comparison within a text-rich KG. Unlike existing methods that often rely on static embeddings or focus on single data modalities, our approach adopts a multi-step, hybrid methodology. A key innovation lies in explicitly partitioning entity attributes based on their character (e.g., free text vs. SKOS concepts), rather than treating all attributes uniformly. This partitioning enables more targeted embedding generation for each data type.

For each partition, we select and apply the most appropriate embedding techniques, combining multiple strategies to extract representative features from heterogeneous data structures within the KG. The process steps are: (i) Querying from the graph database, (ii) Split: split entity attributes based on their character, (iii) Preprocessing, (iv) Embedding generation: generating embeddings using the most appropriate model for the type, (v) Combination: combining different types of embeddings, and (vi) Similarity computation and ranking: Computing similarity scores and based on the similarity scores ranking entities. The pipeline is shown in Figure 2.

Querying from Graph DataBase The pipeline takes a list of project IDs as input to retrieve project entities and related textual attributes, such as SKOS Concepts, and relevant OWL Concepts for projects (e.,g. Context, Questions, Answers) from the Triplestore Jena Fuseki server[4].

Split Once the project entities and attributes are extracted, the attributes are split into two types based on their character:

- SKOS Concepts: Structured attributes are mapped to SKOS concepts, aligning them with terms.
- Free Text: Unstructured text attributes are treated as free text, requiring additional processing for embedding generation.

To facilitate comparison, we also utilize both free text and SKOS concepts in our experiment.

Preprocessing Preprocessing is mainly applied to prepare free text attributes for embedding generation. This step includes:

- Text Cleaning: Lowercasing, removing special characters in the text.
- Stopword Removal: Removing common stopwords (the NLTK English stopwords list [17]) from the text.
- Lemmatization: Converting words to their base forms to standardize vocabulary across contexts. We apply both stemming and lemmatization techniques. Specifically, we used the Porter Stemmer to reduce words to their stems (e.g., "running" to "run") and the WordNet Lemmatizer from the NLTK library, which maps words to their root forms based on linguistic rules

We process SKOS concepts by extracting only the labels associated with each concept, discarding metadata or structural components such as concept URIs, hierarchical relations, or definitions. This allows us to focus on the actual descriptive terms for use in embedding models.

Embedding Generation After preprocessing, we apply and evaluate various embedding techniques and strategies to extract representative features from data formats: free text and SKOS concepts' labels. The method takes a project's attributes (SKOS concepts' labels, free text, or both) as input and maps each of them to different vector representations using graph-based, traditional and contextual embedding models.

- Graph-Based Embeddings (TransE)[18]: We use PyKEEN[19] to implement the TransE to generate embeddings for entities and relationships within the KG, capturing the relational structure between entities. The input is a set of triples (h,r,t), where h (head) is the entity, r (relation) is the relationship type, and t (tail) is the associated SKOS concept or free-text.
- Traditional Word Embeddings (Word2Vec): Since SKOS concepts' labels are short phrases, well-defined and structured, we are interested in evaluating how Word2Vec performs in representing these SKOS concepts's labels(terms). Each word is tokenized using the NLTK tokenizer, and a Word2Vec model is trained on the resulting tokens with the following parameters: vector size = 100, window size = 5, and min_count = 1. We apply different strategies, averaging:

$$e_{term} = \frac{1}{n}\sum_{i=1}^{n} e_i$$

[4]https://jena.apache.org/

and concatenation:

$$e_{term} = [e_1||e_2||...||e_n]$$

derive a single embedding for each term (e_{term}) from the Word2Vec word embeddings, ensuring that SKOS concept labels are represented by one comprehensive embedding.

- Contextual Embeddings:
  - BERT: We are particularly interested in evaluating its performance in representing both SKOS concepts's labels, which are structured, and free text, which is more unstructured and potentially noisy. The terms are tokenized using the BERT tokenizer, which ensures that the model receives appropriately formatted input. The pre-trained BERT model (bert-base-uncased) is employed to generate embeddings for the input terms.
  - MiniLM: We are interested in evaluating its performance on free text, where understanding sentence-level context is crucial for comparison and analysis.
  - LLaMA: Llama-2-7b-chat-hf is used to generate embeddings. We expect it to produce high-quality embeddings for free text, leveraging its ability to understand and represent nuanced meanings within sentences.

Combination In this step, different types of embeddings are combined. The input consists of two sets: a list of SKOS concepts' labels (terms) and free-text(sentences). Various strategies (concatenation, weighted sum and Principal Component Analysis) are employed for combining BERT/Word2Vec and MiniLM-generated embeddings. We use BERT/Word2Vec to generate embeddings for terms and use a pre-trained SentenceTransformer model (all-MiniLM-L6-v2) to generate embeddings for sentences, which produces embeddings with a dimensionality of 384. LLaMA is deemed too computationally intensive and resource-demanding for our use case, where scalability and speed are essential. For this reason, we have chosen not to use it here.

- Concatenation: Embedding from BERT/Word2Vec(SKOS concepts labels) (e_{terms}) and embedding from MiniLM(free text) ($e_{sentence}$) are concatenated into a single vector, preserving the individual properties of each embedding type. A liner layer is used to reduce or increase the dimensionality of BERT word embeddings to align their dimensions with MiniLM-generated embeddings $e_(term) = LinearLayer(e_term)$. This combines the embeddings from two different models into a single embedding of dimension 768 (384 + 384). The combined embedding:

$$e_{combined} = [e_{term}||e_{sentence}]$$

.

- Principal Component Analysis (PCA)[20]: PCA is also applied to reduce the dimensionality of the combined embeddings(concatenation), mitigating redundancy and improving computational efficiency without losing important information:

$$e_{PCA} = PCA(e_{combined})$$

- Weighted Sum: A weighted sum of embeddings is calculated, assigning different importance levels to embeddings to SKOS concepts' labels and free text. Since SKOS concepts' labels are well-defined and structured, while free text may contain more noise, we aim to assign greater weight to SKOS concepts' labels to prioritize their reliability and clarity in representing entities. The weighted sum of the term and sentence embeddings are computed using a parameter α:

$$e_{weighted\_sum\_embedding} = \alpha \cdot e_{word} + (1 - \alpha) \cdot e_{sentence}$$

Similarity Computation and Ranking Once the embeddings are generated by using different methods, we compute the cosine similarity between pairs of entities to determine how closely related they are in the vector space. A similarity threshold is applied to filter out entities with low similarity scores, ensuring that only the most relevant entities are considered. Embeddings generated by different methods produce cosine similarity values that operate on varying scales, necessitating the application of different thresholds. Therefore, the threshold is selected to yield the best performance results based on evaluation metrics. Entities are ranked according to their similarity scores, with higher scores indicating greater similarity. The ranking provides an ordered list of entities relevant to other entities.

5. Experimental Investigation

5.1. Datasets

We evaluate our work by using real-world data - the ARK platform dataset. We focus on the task of comparing "project" entities represented in a text-rich KG (Fig. 1). Projects are modelled by a set of W3C Web Ontology Language 2 (OWL2) [21] classes describing 5 stages of the project lifecycle. Each stage includes a set of 16 analysis questions, their answers in natural language, and an analysis synthesis in three natural language entities called context, mechanism and outcome. The ontology specification is available online[5]. There are over 250 classes and properties defined in the ontology. A project consists of structured data on participants, risks, metadata and 95 natural language fields with a mean of approximately 200 words each and a mean of 3 annotation concepts per natural language field.

Thus the project entity data in the KG can be split into 3 types: OWL2 symbols (OWL2 classes, properties and individuals), natural language property values (textual descriptions, question answers, analysis fields entered by experts i.e. free text), SKOS concepts (which have multi-word concept labels), and linked documents (Evidence). All of these KG elements could be important when determining the similarity of two or more projects. In this study, we focus on natural language property values and SKOS concepts used as annotations. A set of 8 text-rich knowledge graph projects was available to do the experiment. This selection was based on projects which had the presence of all 3 types of data: OWL2 symbols, free text and SKOS concepts. In addition, project relationships such as "related to" and project lifecycles are represented in the graph of projects. Each project contains approximately 36 SKOS concepts and 78 natural language property values, enabling a rich representation of project data.

5.2. Experimental Process

The hypothesis investigated in this experiment is that semantically related projects will be correctly identified by application of the Text-Rich Knowledge Graph Entity Comparison and Ranking Pipeline (Fig. 2) with the appropriate choice of embedding techniques and graph splitting strategies.

To test this hypothesis we present a comparative analysis of KG entity comparison models. In the experiment, the evaluation is done on text from SKOS concepts and free text. Linked documents and OWL schemas will be addressed in future work. One strategy, the TransE graph embedding, uses the whole OWL project entity graph (Table 4).

The graph splitting strategies investigated are: use SKOS Concepts only (Table 1), use free text only (Table 2), use both SKOS Concepts and free text (Table 3, Table 5), and use the entire project entity graph (Table 4). The embedding techniques investigated are Word2Vec, BERT, MiniLM, LLaMA, TransE, combined BERT and MiniLM. The embedding combination techniques investigated are: concatenation, mean, concatenation and PCA, and weighted sum.

For each possible comparison strategy (i.e. a combination of graph splitting strategy, embedding technique and embedding combination technique) we generate a final composite embedding for each project. This final project embedding is then compared to all the other final project embeddings for that strategy using cosine similarity. Then a ranked list of project similarities is generated. Projects below a set threshold are filtered out for having no similarity.

[5] https://w3id.org/ARK/Platform

5.2.1. Embeddings Generation

In this experiment, we employed models including transE, Word2Vec (concatenation), Word2Vec (mean), BERT, MiniLM, LLaMA, a combination of BERT with MiniLM, and a combination of Word2Vec with MiniLM to generate embeddings. For generating embeddings, we utilize Word2Vec (concatenation), Word2Vec (mean), BERT, MiniLM, and LLaMA to generate embeddings for the project's SKOS concepts labels, the project's free text, and a combination of both. Specifically, BERT/Word2Vec is employed to generate embeddings for SKOS concepts' labels and MiniLM to generate embeddings for free text for the combination embeddings. In the case of transE, the input is the project entities and their associated SKOS concepts and free text. The embeddings produced by these models are then used to compute the cosine similarity between projects. Based on these similarity scores, each project is ranked in relation to others, creating a ranked list of potentially related projects. We store the ranking results from each method (Word2Vec(concatenation), Word2Vec(mean), BERT, MiniLM, LLAma, combined BERT with MiniLM and combined Word2Vec with MiniLM) in CSV files as format: project 1, project 2, similarity score.

5.2.2. Ground Truth Generation

Within the GRC KG projects sometimes have an explicitly defined semantic relationship to each other in the graph, for example "has follow-up project". There are 9 such relations with a root property of "has related project". All projects studied in the experiment have a "has related project" relationship to at least one other project in the dataset. We take the presence or absence of this relationship between projects as an initial ground truth of similarity between projects. We query each project's related projects based on those 9 relations as the ground truth for determining similarity between projects (whether or not projects are related). We assume that if two projects share this relationship, they are considered similar, and if not, they are dissimilar. We construct a dictionary, y_true_dict, where each project is associated with a list of related projects that get from the query. The projects are then grouped by their shared relationships using a pandas function, which collects all related projects under each project identifier.

5.2.3. Evaluation Metrics

The evaluation metrics include Mean Average Precision (MAP), Normalized Discounted Cumulative Gain (NDCG) [22], Recall, Precision, and Hits. We compare the ranking results of project entity similarity with the ground truth, which consists of projects and lists of their related projects.

5.2.4. Evaluation

We read our stored CSV files from each method (Word2Vec(concatenation), Word2Vec(mean), BERT, MiniLM, LLAma, combined BERT with MiniLM and combined Word2Vec with MiniLM) which has format project 1, project 2, and similarity score. Initially, a threshold value of 0 was used to eliminate project pairs that have similarity scores above 0. The filter results are then grouped by project 1, mapping each project to a list of related projects (project 2) and storing them in a dictionary(results_dict), where the key is the project and the value is a list of predicted related projects. Then pass the results_dict and y_true_dict to our evaluation function evaluation(y_true_dict, results_dict) and calculate MAP, NDCG, Recall, Precision and Hits, particularly for K=1.

To evaluate the impact of the threshold, we experimented with different threshold values. For K=1 specifically, our results showed that changing the threshold had minimal impact on the evaluation metrics. This is because metrics like MAP, NDCG, and Hits are ranking-based, evaluating the order of similarity scores rather than their absolute values. Since only the top-ranked item is relevant for K=1, the similarity threshold does not influence the ranking as long as the scores are correctly sorted. However, for larger values of K, thresholds may have a more noticeable effect, as they could influence the inclusion of lower-ranked items in the evaluation.

Table 1
Experimental Results of SKOS Concepts

Embedding	MAP@K=1	NDCG@K=1	Recall@K=1	Precision@K=1	Hits@K=1
Word2Vec_(concatenation)	0.250	0.250	0.250	0.250	0.250
Word2Vec_(mean)	0.000	0.000	0.000	0.000	0.000
BERT	**0.750**	**0.750**	**0.750**	**0.750**	**0.750**
MiniLM	0.000	0.000	0.000	0.000	0.000
LLaMA	0.500	0.500	0.500	0.500	0.500

Table 2
Experimental Results of Free Text

Embedding	MAP@K=1	NDCG@K=1	Recall@K=1	Precision@K=1	Hits@K=1
Word2Vec_(concatenation)	0.250	0.250	0.250	0.250	0.250
Word2Vec_(mean)	0.000	0.000	0.000	0.000	0.000
BERT	0.500	0.500	0.500	0.500	0.500
MiniLM	**1.000**	**1.000**	**0.875**	**1.000**	**1.000**
LLaMA	0.500	0.500	0.500	0.500	0.500

Table 3
Experimental Results of SKOS Concepts and Free Text

Embedding	MAP@K=1	NDCG@K=1	Recall@K=1	Precision@K=1	Hits@K=1
Word2Vec_(concatenation)	0.250	0.250	0.250	25.000	25.000
Word2Vec_(mean)	0.000	0.000	0.000	0.000	0.000
BERT	0.500	0.500	0.500	0.500	0.500
MiniLM	**1.000**	**1.000**	**0.875**	**1.000**	**1.000**
LLaMA	0.500	0.500	0.500	0.500	0.500

5.3. Experimental Results

The following tables present the experimental results of Word2Vec(concatenation), Word2Vec(mean), BERT, MiniLM, LLAma, combined BERT with MiniLM and combined Word2Vec with MiniLM embedding techniques for comparing project entities within our GRC domain-specific knowledge graph (KG). We observe that the results are likely 0%, 25%, 50%, 75%, and 100%. This is perhaps because the dataset is very limited which likely causes the performance metrics to be coarse-grained, as there are fewer relevant items to rank. In addition, the evaluation focuses heavily on whether the correct result is retrieved at the very top (K=1), leading to discrete jumps in performance scores. Metrics like Precision, Recall, and Hits@K=1 assess whether the first-ranked item is relevant, so the scores are binary: if the top result is correct, the model scores 100%; if not, it scores 0%. Intermediate scores like 25% and 50% arise when the evaluation is averaged across multiple test cases where the models perform correctly in some but not all instances.

Table 1 presents the evaluation results for Word2Vec (concatenation), Word2Vec (mean), BERT, MiniLM, and LLaMA on the SKOS Concepts dataset. Bert outperforms the other models, which may be attributed to the nature of the SKOS Concepts, as they consist of short phrases or domain-specific terms. BERT's contextual embeddings capture how individual words contribute to the overall meaning of a phrase, depending on their position and context, giving it an advantage over Word2Vec. In contrast, Word2Vec uses static embeddings, treating phrases as simple combinations of word vectors, which is less effective for understanding specialized, multi-word terms in this dataset.

Table 2 presents the evaluation results for Word2Vec, BERT, MiniLM, and LLaMA on the free text dataset. MiniLM outperforms the other models. In contrast, LLaMA may not be well-suited for the specific characteristics of our dataset, which is limited in size, leading to its suboptimal performance in this task.

Table 3 shows that MiniLM surpasses Word2Vec, BERT, and LLaMA on the evaluation results for both the SKOS Concepts and free text datasets. MiniLM outperforms the other models.

Table 4
Experimental Results of TransE

Embedding	MAP@K=1	NDCG@K=1	Recall@K=1	Precision@K=1	Hits@K=1
TransE	0.500	0.500	0.375	0.500	0.500

Table 4 shows that the result generated by graph-based embedding transE. The result shows it is not as good as MiniLM generated embeddings for both SKOS concepts and free text but outperformance Word2Vec.

Table 5
Experimental Results of Combination of Word2Vec(mean) with MiniLM

Embedding	MAP@K=1	NDCG@K=1	Recall@K=1	Precision@K=1	Hits@K=1
Word2Vec&MiniLM_(concatenation)	**1.000**	**1.000**	**0.875**	**1.000**	**1.000**
Word2Vec&MiniLM_(concatenation&pca)	0.500	0.5000	0.5000	0.5000	0.5000
Word2Vec&MiniLM_(weighted sum)	**1.000**	**1.000**	**0.875**	**1.000**	**1.000**

For the weighted sum, α is set to 0.1, in light of the previous results indicating that MiniLM-generated embeddings achieve better performance for free text, we have chosen to allocate more weight to MiniLM.

Table 5 shows the Word2Vec-generated embeddings combined with MiniLM-generated embeddings in different strategies. Concatenation and weighted sum get the same results. They provide the best overall performance. The combined embeddings of Word2Vec and MiniLM also yield results that outperform Word2Vec alone but do not surpass the performance of MiniLM on its own in terms of concatenate Word2Vec-generated embeddings and MiniLM-generated embeddings. Surprisingly, when reducing diminution with PCA the performance is worse than both Word2Vec and MniLM individually.

Table 6
Experimental Results of Combination of BERT with MiniLM

Embedding	MAP@K=1	NDCG@K=1	Recall@K=1	Precision@K=1	Hits@K=1
BERT&MiniLM_(concatenation)	**0.750**	**0.750**	**0.750**	**0.750**	**0.750**
BERT&MiniLM_(concatenation&pca)	0.500	0.500	0.500	0.500	0.500
BERT&MiniLM_(weighted sum)	**0.750**	**0.750**	**0.750**	**0.750**	**0.750**

For the weighted sum, α is set to 0.1, in light of the previous results indicating that MiniLM-generated embeddings achieve better performance for free text, we have chosen to allocate more weight to MiniLM.

Table 6 shows the BERT-generated embeddings combined with MiniLM-generated embeddings in different strategies. Concatenation and weighted sum get the same results. Concatenation provides the best overall performance. The combined embeddings of BERT and MiniLM results are the same as BERT alone and do not surpass the performance of MiniLM on its own.

Surprisingly, combining Word2Vec with MiniLM delivers better performance than combining BERT with MiniLM, even though BERT outperforms Word2Vec when used on its own. While BERT is a more powerful model individually, the simpler embeddings from Word2Vec seem to complement MiniLM more effectively than BERT's already complex representations.

6. Conclusions and Future work

This study examines existing methods for entity comparison in text-rich, domain-specific knowledge graphs (KGs). It explores various embedding techniques—such as TransE embeddings and Word2Vec embeddings—and evaluates different strategies for generating term embeddings using Word2Vec for comparing entities within these specialized KGs. We also experimented with using different properties and partitioning KG entity attributes based on their structure (e.g., free text vs. SKOS concepts) to improve entity comparison accuracy.

The results indicate that models like BERT and MiniLM, which leverage contextual embeddings, are significantly more effective than static embeddings like the traditional word embedding method Word2Vec for both well-defined and structured short phrases and free text datasets. Additionally, we explore the opportunities to further enhance entity comparison accuracy by combining different embedding techniques using strategies like PCA, concatenation, and weighted sum. Interestingly, we find that combining Word2Vec with MiniLM achieves better performance than combining BERT with MiniLM, despite BERT outperforming Word2Vec when used individually.

In future work, we will extend our experiments by integrating graph-based embeddings with language embeddings to better capture the structure and content of knowledge graphs. Furthermore, we aim to expand the experiments to larger, public text-rich KG datasets such as DBpedia, which contains abundant textual data, including unstructured content, concepts, and links to documents, to validate our findings. Ultimately, we will incorporate these improvements into our GRC platform, making it more effective in real-world scenarios. Although our dataset is limited, this study provides valuable insights for constructing GRC systems using knowledge graphs. By demonstrating the advantages of different embedding approaches, our findings contribute to advancing the state of the art in entity comparison and enhancing the utility of text-rich KGs. These improvements have potential applications in areas such as semantic search, recommendations, and decision-support systems.

Acknowledgments

This research was conducted with the financial support of Research Ireland under the ADAPT Centre for AI-driven Digital Content Technology which is funded under the Research Ireland Research Centres Programme (Grant 13/RC/2106_P2). For the purpose of Open Access, the authors have applied a CC-BY public copyright license to any author-accepted manuscript version arising from this submission.

References

[1] D. Daza, D. Alivanistos, P. Mitra, T. Pijnenburg, M. Cochez, P. Groth, Bioblp: a modular framework for learning on multimodal biomedical knowledge graphs, Journal of Biomedical Semantics 14 (2023) 20.

[2] A. Ettorre, A. Bobasheva, C. Faron, F. Michel, A systematic approach to identify the information captured by knowledge graph embeddings, in: IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, 2021, pp. 617–622.

[3] J. Devlin, Bert: Pre-training of deep bidirectional transformers for language understanding, arXiv preprint arXiv:1810.04805 (2018).

[4] A. Crotti Junior, M. Basereh, Y. Abgaz, J. Liang, N. Duda, N. McDonald, R. Brennan, The ark platform: Enabling risk management through semantic web technologies (2020).

[5] C. HANDBOOK, Governance, risk, and compliance handbook (2008).

[6] A. Miles, S. Bechhofer, Skos simple knowledge organization system reference, W3C recommendation (2009).

[7] A. Petrova, E. V. Kostylev, B. Cuenca Grau, I. Horrocks, Query-based entity comparison in knowledge graphs revisited, in: The Semantic Web–ISWC 2019: 18th International Semantic Web Conference, Auckland, New Zealand, October 26–30, 2019, Proceedings, Part I 18, Springer, 2019, pp. 558–575.

[8] A. Petrova, E. Sherkhonov, B. Cuenca Grau, I. Horrocks, Entity comparison in rdf graphs, in: The Semantic Web–ISWC 2017: 16th International Semantic Web Conference, Vienna, Austria, October 21–25, 2017, Proceedings, Part I 16, Springer, 2017, pp. 526–541.

[9] S. El Hassad, F. Goasdoué, H. Jaudoin, Learning commonalities in sparql, in: International Semantic Web Conference, Springer, 2017, pp. 278–295.

[10] M. Azmy, P. Shi, J. Lin, I. F. Ilyas, Matching entities across different knowledge graphs with graph embeddings, arXiv preprint arXiv:1903.06607 (2019).

[11] A. Parravicini, R. Patra, D. B. Bartolini, M. D. Santambrogio, Fast and accurate entity linking via graph embedding, in: Proceedings of the 2nd Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA), 2019, pp. 1–9.

[12] C. Pesquita, D. Faria, A. O. Falcao, P. Lord, F. M. Couto, Semantic similarity in biomedical ontologies, PLoS computational biology 5 (2009) e1000443.

[13] M. A. Rodriguez, M. J. Egenhofer, Determining semantic similarity among entity classes from different ontologies, IEEE transactions on knowledge and data engineering 15 (2003) 442–456.

[14] A. Maind, A. Deorankar, P. Chatur, Measurement of semantic similarity between words: A survey, International Journal of Computer Science, Engineering and Information Technology 2 (2012) 51–60.

[15] K. W. Church, Word2vec, Natural Language Engineering 23 (2017) 155–162.

[16] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al., Llama: Open and efficient foundation language models, arXiv preprint arXiv:2302.13971 (2023).

[17] S. Bird, E. Klein, E. Loper, Natural language processing with Python: analyzing text with the natural language toolkit, " O'Reilly Media, Inc.", 2009.

[18] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, O. Yakhnenko, Translating embeddings for modeling multi-relational data, Advances in neural information processing systems 26 (2013).

[19] M. Ali, M. Berrendorf, C. T. Hoyt, L. Vermue, S. Sharifzadeh, V. Tresp, J. Lehmann, Pykeen 1.0: a python library for training and evaluating knowledge graph embeddings, Journal of Machine Learning Research 22 (2021) 1–6.

[20] A. Maćkiewicz, W. Ratajczak, Principal components analysis (pca), Computers & Geosciences 19 (1993) 303–342.

[21] G. Antoniou, F. v. Harmelen, Web ontology language: Owl, Handbook on ontologies (2009) 91–110.

[22] K. Järvelin, J. Kekäläinen, Cumulated gain-based evaluation of ir techniques, ACM Transactions on Information Systems (TOIS) 20 (2002) 422–446.