Binarization of Knowledge Graph Embeddings with Semantic Preservation

Master Thesis

presented by
Vitor Faria de Souza
Matriculation Number 1844490

submitted to the
Data and Web Science Group
Prof. Dr. Heiko Paulheim
University of Mannheim

January 2024

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Knowledge graphs (KGs) have emerged as powerful tools for representing real-world information from various domains in a format that is both human-readable and machine-readable. A knowledge graph is a directed graph consisting of nodes representing entities and edges representing relationships between these entities. This graph-based representation allows the integration of various data sources and facilitates the exploration of connections and patterns within the data. Knowledge graphs find applications in diverse domains such as semantic search, recommendation systems, question answering, and data integration.

As symbolic representations of knowledge, their applicability can be expanded when they are converted to numerical representations, and this is where knowledge graph embeddings (KGEs) play a crucial role. KGEs capture semantic information inherent in the structure of the graph, such as complex relationships between entities, in a compact and expressive manner. These dense and low-dimensional vector representations enable computational models to calculate distances and similarities and to make predictions based on underlying patterns in the KG's triples. By representing entities and relationships as vectors, they facilitate the application of machine learning algorithms for tasks such as entity classification, entity clustering, and link prediction.

However, one of the significant drawbacks of traditional KGEs is their ever-growing size with the increasing scale of knowledge graphs. As knowledge graphs expand to incorporate more entities and relationships, the size of KGEs increases proportionally, leading to challenges in storage, computation, and scalability. This growth poses significant challenges in terms of storage and computational efficiency when working with traditional KGEs. As knowledge graphs continue to expand, the

need for more compact and scalable representations of KGEs becomes imperative. Binarization techniques offer a promising approach to address these challenges by compressing embeddings into binary vectors while preserving their semantic information, as already achieved in previous work for word embeddings [45, 24, 28]. By reducing the storage footprint and computational complexity of KGEs, binarization techniques enable efficient processing and analysis of large-scale knowledge graphs, without losing their applicability.

1.1 Problem Statement

Despite the advantages of KGEs, the storage requirements and computational overhead associated with their use become substantial as knowledge graphs scale. Traditional embeddings are often represented as high-dimensional floating-point vectors, resulting in large file sizes. The need for more compact representations that maintain semantic information is evident, especially in applications with limited storage capacity or those that require real-time processing. The challenge is to explore binarization techniques that significantly reduce the storage footprint while preserving the semantic richness of the original embeddings in downstream tasks.

However, compact representations are subject to the same trade-off observed in dimensionality reduction techniques: the more compact they become, the more information is probably lost. Therefore, semantic preservation should be analyzed on both sides. Binarized embeddings should drastically reduce file sizes and computational complexity of operations and still maintain competitive scores in performance metrics, such as accuracy in classification, across various tasks and datasets. In the best case, the loss in these scores when using binarized embeddings compared to their original version should not be statistically significant. Thus, the problem statement revolves around finding effective binarization techniques that strike a balance between storage efficiency and semantic preservation of KGEs, addressing the challenges posed by the ever-growing size of knowledge graphs and the diverse requirements of knowledge graph-based applications.

1.2 Pursued Approach

To address the challenges outlined above, this research pursues a comprehensive approach towards investigating binarization techniques for knowledge graph embeddings. Various binarization methods were evaluated using the established Evaluation Frameworks GEval [29, 30] and DLCC [33] on a variety of tasks and datasets, with the aim of systematically comparing and analyzing the semantic preservation

of KGEs in different scenarios. This approach involves experimenting with a naive binarization method as baseline, that simply binarizes vectors based on averages, and a more advanced technique, based on the autoencoding architecture proposed by Tissier et al. for binarizing word embeddings [45].

Furthermore, this research emphasizes the importance of comparing different types of KGEs, including various models such as RDF2Vec [37], TransE [5], ComplEx [47], DistMult [53], RESCAL [26], RotatE [44] and TransR [17]. By evaluating the performance of binarization techniques across a diverse set of embeddings, the research aims to identify the most effective approaches to preserving semantic information while reducing the storage footprint. By evaluating its performance on various tasks and datasets, this research also examines how the size of the datasets and the complexity of the task influence the effectiveness of binarization techniques. Thus, the pursued approach seeks not only to provide insight into optimizing storage and computation without compromising the semantic richness of KGEs, but also to interpret *what* information is lost when applying such processes.

The findings of this research have practical implications for applications that rely on knowledge graph embeddings. By identifying effective binarization techniques and understanding their impact on diverse tasks and datasets, this research contributes to the development of more efficient and scalable solutions for handling large-scale knowledge graphs. The exploration of autoencoding methods for near-lossless binarization opens avenues for optimizing storage and computation without compromising semantic information.

1.3 Outline

This thesis is organized into several chapters, each addressing different aspects of the research. Chapter 2 provides a comprehensive review of the basic concepts and related work, highlighting existing approaches to knowledge graph embeddings and binarization techniques. Chapter 3 details the methodology used in the experimentation, including the source of the original KGE files, the evaluation frameworks, and the binarization process. The results and analysis are presented in Chapter 4, followed by a discussion of the implications and limitations. Finally, Chapter 5 offers a conclusion, summarizing key findings and suggesting avenues for future research. This document also includes two Appendices: Appendix A, linking to the source code and notebooks used to implement and analyze the experiments, and Appendix B, with granular experimental results.

Chapter 2

Theoretical Framework

In this chapter, important concepts from the literature that form the basis for this thesis are discussed and reviewed. It starts with the introduction of Knowledge Graphs and Knowledge Graph Embeddings in Sections 2.1 and 2.2, followed by the challenge of preserving semantics when attempting to reduce the dimensionality or cardinality of vectors in Section 2.3. Furthermore, relevant work in a direction similar to that presented in this thesis is presented in Section 2.4.

2.1 Knowledge Graphs

2.1.1 General Concepts and Properties

Knowledge graphs are knowledge representations that model information in the form of entities and relationships between them or their own attributes in a graph data structure. They are able to encode real-world information about any type of entity, such as people, places, and institutions, in a directed labeled graph, whose triples *(subject, predicate, object)* are machine-readable. For example, the sentence *"Emily was born in the USA and works as a Marketing Analyst at Savoir in Paris, France"* can be visualized graphically as in Figure 2.1 and encoded as the following triples *(s, p, o)*:

```
(Emily, birthPlace, United_States_of_America)
(Emily, occupation, Marketing_Analyst)
(Emily, worksAt, Savoir)
(Savoir, locatedIn, Paris)
(Paris, locatedIn, France)
```

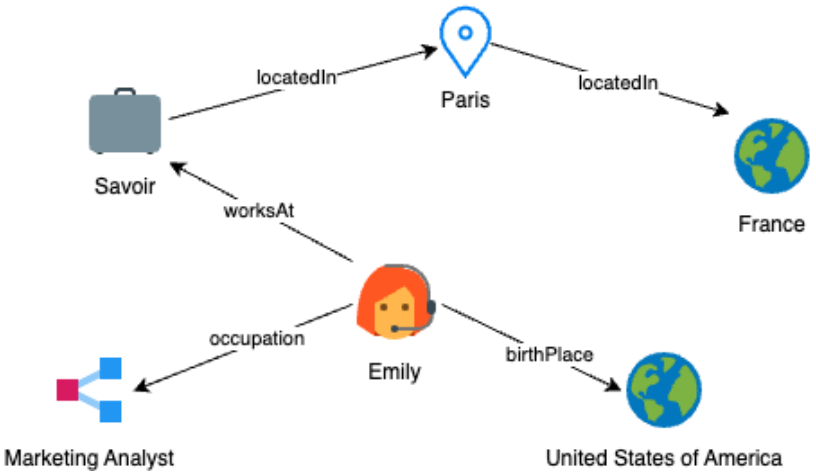


Figure 2.1: Illustration of a knowledge graph.

Given a finite set of entities $\mathcal{E}$ and a finite set of relations $\mathcal{R}$, a knowledge graph $\mathcal{K}$ can be formally defined as a set of triples *(subject, predicate, object)* with $\mathcal{K} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$. Although certain relations between entities can be symmetric, such as *spouse* (if Anne is married to Bob, Bob is also married to Anne), KGs are directed.

This idea can be dated back from 2001, when Berners-Lee et al. proposed the Semantic Web [2] and its technologies, such as the Resource Description Framework (RDF), Unique Resource Identifiers (URIs), and the Web Ontology Language (OWL), which were part of W3C's recommendation to represent, publish, and interlink information on the Web, and enhance its functionality and interoperability. The term Knowledge Graph came years later, when Google started using these technologies to improve their Web search, under the motto "things, not strings" [41]. Google used their Knowledge Graph to enrich search results with semantically structured summaries, as well as other features that improve the search experience for users and the search engine's ability itself.

To better represent real-world information, entities and relations can also have special properties. In the current example, Emily is an instance of class *Person*, while Savoir is an *Organization*, and France/USA are of type *Country*. The city of Paris is not only *located* in France, but also its capital, so the relation *capitalOf* can be used as a subproperty of *locatedIn*. Such properties and hierarchies can be added to the knowledge graph and used for further logical reasoning to imply if a certain statement holds even when not explicitly present in the graph.

2.1.2 Open Knowledge Graphs

Knowledge graphs are generally structured for different purposes by organizations and communities, and although some of them are private, there are also open KGs published online, with accessible content for the public good [12]. Prominent examples of open KGs that cover a wide variety of domains are DBpedia [15], YAGO [43], Wikidata [49], and the discontinued Freebase [4]. The first two are extracted from sources such as Wikipedia, while the latter were built mainly by volunteers [12]. Freebase is a good example of open KG that became private after its acquisition by Google, becoming an important part of the creation of the Google Knowledge Graph [41].

These open KGs contain a lot of valuable information in millions of queriable triples, but differ in terms of size, focus, data quality, and number of instances per class, making each KG more suitable for each task and application [36]. They are also interlinked, as shown in Figure 2.2, and this is where DBpedia plays a central role when compared to Wikidata, YAGO and two other KGs: the deprecated expert-curated OpenCyc [16] and NELL [7], which extracts knowledge from a large Web corpus.

Open KGs may also differ by construction method, which is crucial for their accuracy and completeness. Nickel et al. [25] classified KGs into four different construction methods:

- *Curated* approaches, where triples are produced manually by a closed group of experts (e.g. OpenCyc [16]);
- *Collaborative* approaches, where triples are produced manually by an open group of volunteers (e.g., Wikidata [49] and Freebase [4]);
- *Automated semi-structured* approaches, where triples are extracted automatically from semi-structured text (such as infoboxes in Wikipedia) with predefined rules or regular expressions (e.g., DBpedia [15], YAGO [43] and Freebase [4]);
- *Automated unstructured* approaches, where triples are extracted automatically from unstructured text using machine learning and natural language processing techniques (e.g., NELL [7]).

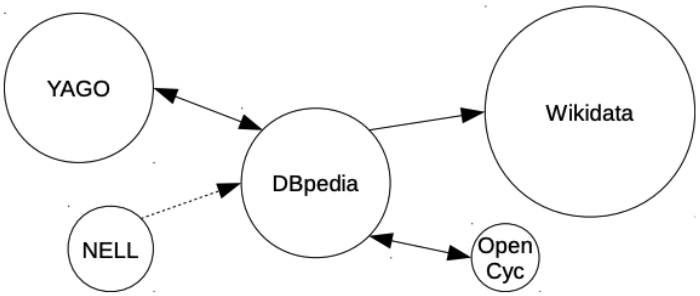


Figure 2.2: Open Knowledge Graphs interlinked, extracted from [36]. The size of the circles is proportional to the number of graph instances as of 2017, except for OpenCyc, which would need to be depicted an order of magnitude smaller and was later shut down.

2.1.3 Applications and Challenges

Knowledge graphs and their capabilities are useful in a variety of applications. Zou's survey [54] encapsulated some of them in the following fields:

- **Information Retrieval**: Knowledge graphs can be used to improve the relevance of search results by providing additional context about the entities searched for, as well as to improve the search engine's ability to understand queries and documents.
- **Recommender Systems**: Knowledge graphs can be used to recommend products, services and information to users based on their past interactions and interests, as well as relationships and similarities between items. They can also improve the accuracy of collaborative filtering recommenders by tackling the cold-start problem with side information.
- **Question Answering**: Knowledge graphs can enable virtual assistants to answer questions based on known facts. IBM's Watson, for example, used several knowledge bases such as YAGO [43] and DBpedia [15] as its data source to defeat human experts in the Jeopardy Chalenge [10].
- **Domain Specific**: The survey presents specific applications of Knowledge Graphs in the medical and financial domains, as well as in cybersecurity, news, and education. In the financial domain, for example, KGs are used together with news crawling, sentiment analysis, Named Entity Recognition (NER), and other methods to predict stock prices [19].

Most of these applications can benefit from the use of precalculated embeddings, as discussed in Section 2.2. Among them, Information Retrieval should be noted as a very relevant application since Google launched the term. Google's Knowledge Graph [41] was said to contain 3.5 billion facts about the world at its launch (and more than 70 billion as of 2019 [27]), and was designed to improve their search engine with semantically meaningful results. Its current size is unknown as it is a private property of the company. Besides Google and IBM, other big tech companies have also created their own industry-scale KGs for internal use, such as Facebook [27], Amazon [13], eBay [27], and Microsoft [40].

However, they also have known limitations. The most relevant challenges involve completeness and data quality, which can relate to containing precise facts but also to being consistent with the KG's own schema rules. For example, KGs based on Wikipedia crawling tend to include only facts contained in Wikipedia, which has considerably more information on American actors and actresses than Indian or Nigerian ones, although Indian and Nigerian film industries (Bollywood and Nollywood) are actually larger than the American one in terms of output [25]. This suggests not only incompleteness, but also biased completeness.

In terms of data quality, it is important to note that some KG construction methods prioritize accuracy over completeness, and vice versa. Curated KGs tend to sacrifice completeness to achieve better accuracy, whereas automated approaches tend to yield a few unprecise statements [25]. For all construction methods, it is acceptable not to contain all possible information about a certain entity when following the Open World Assumption, i.e., a fact not contained in the KG is not assumed to be false. However, the place of birth attribute was missing for 71% of all people included in Freebase as of 2014, although this was a mandatory property of the schema [50]. Dealing with missing links between entities is such a challenge that link prediction (LP) was established as a relevant task with several possible approaches. One of them is the usage of Knowledge Graph Embeddings, discussed in the next section.

2.2 Knowledge Graph Embeddings

2.2.1 General Concepts and Applications

Knowledge graph embeddings (KGEs) are a type of representation learning technique that converts entities and optionally relationships in a knowledge graph into low-dimensional vectors. These vectors capture the semantic meaning of entities and relationships in a numerical form and can therefore be used to perform a variety

of tasks and potentialize the applicability of KGs in most of the fields mentioned in Section 2.1.3. They are a more specific kind of graph embeddings, which can profit from the fact that edges are labeled to achieve more nuanced representations. Graph embeddings methods (including KGEs) are extensively described in surveys [6] and [52], and Figure 2.3 contains an example that illustrates it.

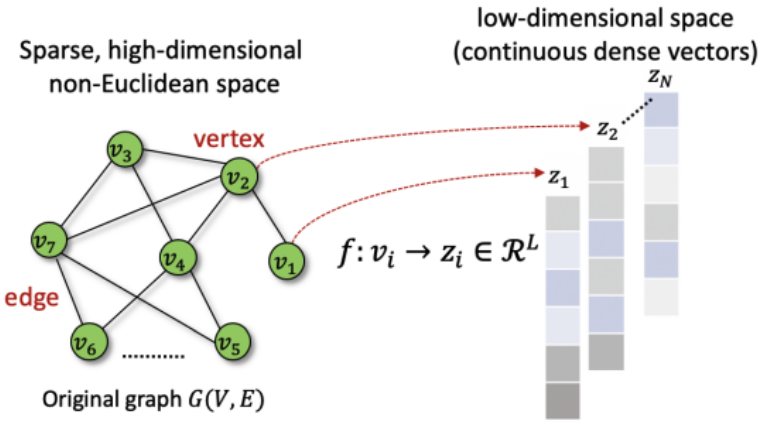


Figure 2.3: Illustration of node embeddings, extracted from [52], where each vertex v_i of a graph $G = (V, E)$ is represented as a L-dimensional continuous vector z_i using the graph embedding model f.

Given the formal description of knowledge graphs in Section 2.1.1, their embeddings can be defined as a projection Π for all entities $e \in \mathcal{E}$ and optionally for all relations $r \in \mathcal{R}$ into a multidimensional space of dimension Δ, so that $\Pi = \{\pi_i \in \Re^{\Delta}\}$, where $i \in \{1, 2, ..., |\mathcal{E}|\}$, or $i \in \{1, 2, ..., |\mathcal{E}| + |\mathcal{R}|\}$ if relations are also represented [34]. For techniques that project vectors in a complex vector space, such as ComplEx [47] and RotatE [44], the definition of Π must be extended to $\Pi = \{\pi_i \in \mathbb{C}^{\Delta}\}$. In order to achieve a dense vector representation, $\Delta \ll |\mathcal{E}|$ is substantial.

There are several techniques for learning KGEs, and many of them are trained with the established link prediction (LP) task in mind. When that is the case, KGE models generally fall into the category of *edge reconstruction*, as defined by [6], and attempt to capture the plausibility of a triple by optimizing some scoring function, so that a nonexistent triple can be evaluated as plausible or not, given the KG. However, they can also be trained with other tasks in mind, such as data mining, information retrieval, and recommendation systems, where the (dis-)similarity be-

tween entities tends to be more important than the plausibility of a triple. The KGE techniques exploited in this work are described in Section 2.2.2.

2.2.2 Models and Techniques

The survey by Cai [6] categorizes graph embedding techniques into 5 different types: matrix factorization, deep learning, edge reconstruction, graph kernel, and generative models. In this work, techniques from the first three of these categories are used and a short comparison of their advantages and disadvantages is presented in Table 2.1.

Category	Subcategory	Advantages	Disadvantages
matrix factorization	graph Laplacian eigenaps	consider global node proximity	large time and space consumption
	node proximity matrix factorization		
deep learning	with random walk	effective and robust, no feature engineering	a) only consider local context within a path b) hard to find optimal sampling strategy
	without random walk		high computation cost
edge reconstruction	maximize edge reconstruct probability	relatively efficient training	optimization using only observed local information, i.e., edges (1-hop neighbour) or ranked node pairs
	minimize distance-based loss		
	minimize margin-based ranking loss		

Table 2.1: Comparison of Graph Embedding techniques exploited in this work, adapted from [6].

From the matrix factorization category, three different approaches are used in this work: the well-known RESCAL [26], which models a graph as a three-way tensor and subsequently applies tensor decomposition, as well as its extensions DistMult [53] and ComplEx [47]. The matrix factorization of RESCAL is illustrated in Figure 2.4, and is considered an expensive operation. DistMult is a scalability improvement over RESCAL that assumes symmetric relationships, i.e., all triples

(i, k, j) and their symmetric pairs (j, k, i) are assigned the same scores, which can drastically reduce the tensor's size. ComplEx, on the other hand, extends DistMult by using complex vector spaces. Both DistMult and ComplEx are more scalable than RESCAL and can handle asymmetric relationships.

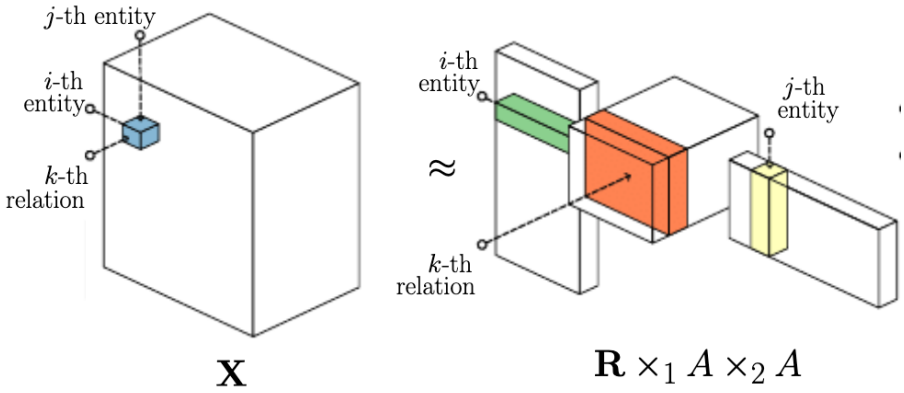


Figure 2.4: RESCAL illustration extracted from [14], where the triple (i, k, j) is approximated by a matrix multiplication.

Three other KGE techniques based on edge reconstruction are also exploited. TransE [5], a widely adopted approach to the LP task, models relations by interpreting them as translations of the entities in the low-dimensional embedding vector space. Inspired by the translation properties of word embeddings [23], this technique trains embeddings h, r, t, such that $h + r \approx t$. For example, when summing the vectors of entities Berlin and Paris with the vector of the relation *capitalOf*, the results are expected to lie close to the vectors of Germany and France in the embedding space, respectively. Two main loss functions are used when training a TransE model: the $L1-$ and $L2-$ norm, which deal differently with noise and outliers in the dataset.

TransR [17] extends TransE by introducing separate embedding spaces for entities and relations (see Figure 2.5), allowing it to model compositional rules and nonisomorphic relationships more effectively.

Unlike TransE and TransR, RotatE [44] conceptualizes relations as rotations of vertices in a complex space, allowing a more nuanced representation of complex relationship patterns, as shown in Figure 2.6.

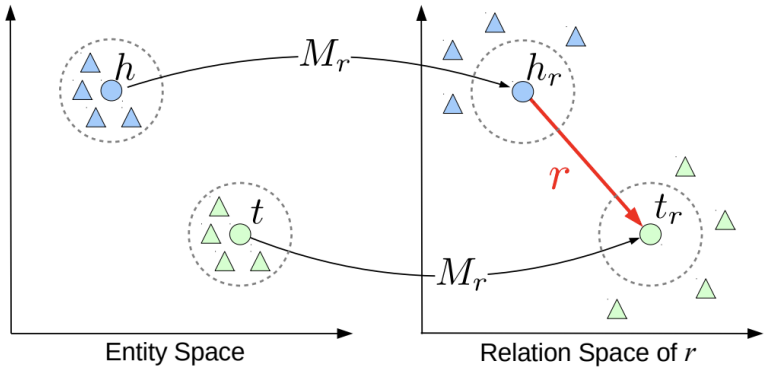
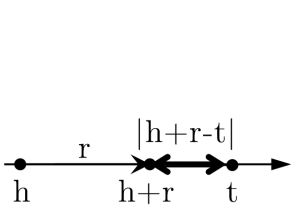
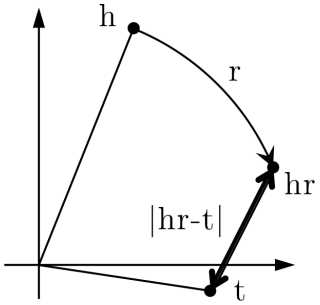


Figure 2.5: TransR illustration from [17].



(a) TransE models r as a translation in real line.



(b) RotatE models r as a rotation in complex plane.

Figure 2.6: Illustrations of TransE and RotatE with only one dimension of embedding, from [44].

From the deep learning category, the RDF2vec [37] family of knowledge graph embeddings is exploited. It first generates random walks of specific length in the knowledge graph and then fits a Word2Vec [23] model on the generated text corpus. Therefore, the varieties of this family fall into the random walk subcategory among deep learning models, which is illustrated in Figure 2.7. It can be considered as an extension of the well-known Node2Vec [11] technique for graph embedding.

The Word2Vec step can be implemented in a CBOW fashion (training the model to predict the center word w from its context k) or Skipgram (predicting the context

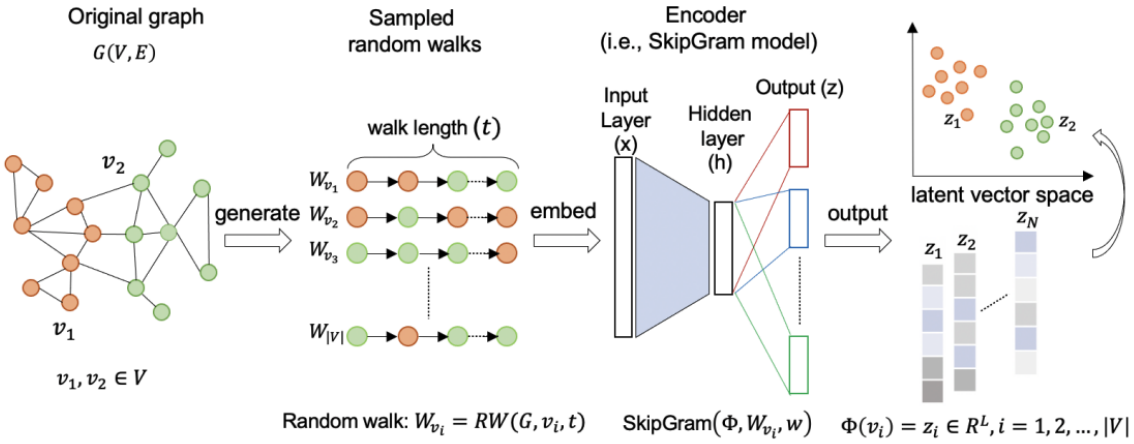


Figure 2.7: Illustration of deep learning Graph Embeddings based on random walking, extracted from [52].

k from the center word *w*), which are depicted in Figure 2.8, and in both cases the hidden layer is used to produce word embeddings. Both steps can be adapted to different situations, giving birth to an entire family of RDF2vec variations [34].

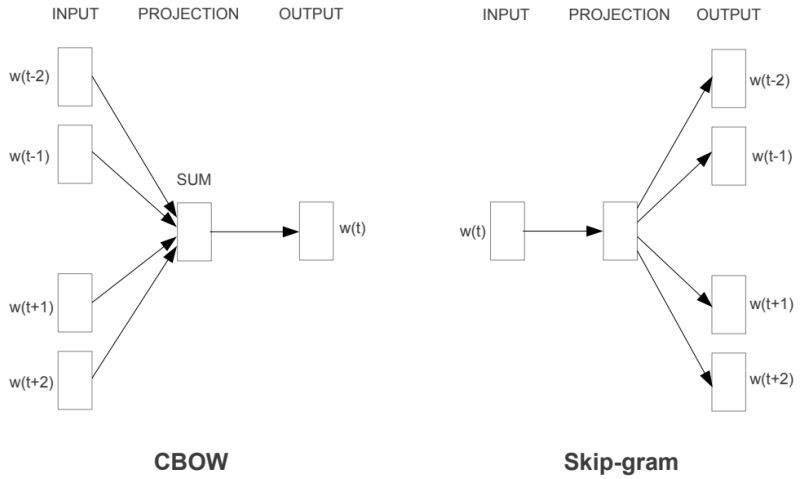


Figure 2.8: Word2vec's architectures CBOW and Skipgram, extracted from [23].

To address Word2vec's shortcoming of being insensitive to the order of the context

words in a sentence and their actual distance from the central word *w*, Ling et al. [18] proposed a structured extension of the Word2vec algorithm that takes its order into account, originally named CWindow and Structured Skip-Ngram (Figure 2.9). This extension was further implemented in RDF2vec embeddings (the so-called ORDER-AWARE or OA) with significant performance gains [32].

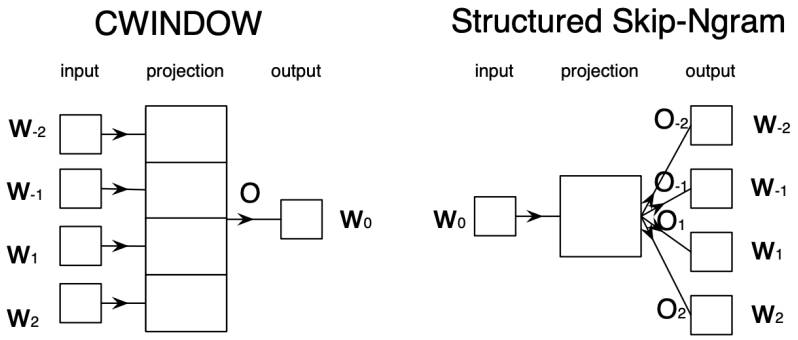


Figure 2.9: Structured Word2vec architecture, extracted from [18].

2.2.3 Evaluation and Metrics

Knowledge graph embeddings can be evaluated differently depending on the task, and a common way to do it is to apply link prediction in the domain of knowledge base completion. With the release of TransE [5], in 2013, two evaluation datasets based on real knowledge graphs were published: FB15k, with Freebase triples [4], and WN18, built on WordNet [42]. These datasets were initially introduced for link prediction tasks, where known triples $(head, relation, tail)$ are used to predict the tail (given the head and the relation) or the head (given the relation and the tail) of unknown triples. However, data leakage concerns have been raised regarding the simplicity of inferences in these datasets due to inverse relations and logical reasoning. For example, the triple $(BarackObama, nationality, UnitedStated)$ could be directly inferred from the triples $(BarackObama, placeOfBirth, Honolulu)$ and $(Honolulu, cityOf, UnitedStates)$, and the triple $(Beyonce, spouse, JayZ)$ could be directly inferred from $(JayZ, spouse, Beyonce)$. Similar inferences are possible in the case of hypernym/ hyponym relations. This led to the proposal of more challenging corrections for these datasets, namely FB15k-237 [46], with selected 237 relations, and WN18RR [9], with selected 11 relations.

Together with WN18RR, [9] also introduced YAGO3-10 as a larger evaluation dataset and a corrected variant of its predecessor YAGO3 [20]. Similarly, Shi et al. [39] introduce the large evaluation sets DBpedia50k and DBpedia500k, suitable for evaluating a model under both open and closed world assumptions. In these datasets, KGEs are typically evaluated by mean reciprocal rank (MRR) and HITS@k (usually with $k \leq 10$). Later, Alshagari et al. [1] proposed the Concept2vec framework to evaluate ontological concepts, covering categorization, hierarchy, and logic validation. This evaluation is mostly based on coherence, assuming that ontological concepts play a crucial role in knowledge graphs and deserve especially high-quality embeddings. Figure 2.10 illustrates how entities at the instance level should be close to those of the ontological level. For example, the embeddings of Berlin, Paris and London are expected to be close to the embedding of City in the embedding vector space and far from entities of other types, such as Barack Obama and Bill Clinton, which are expected to be closer to the embedding of President.

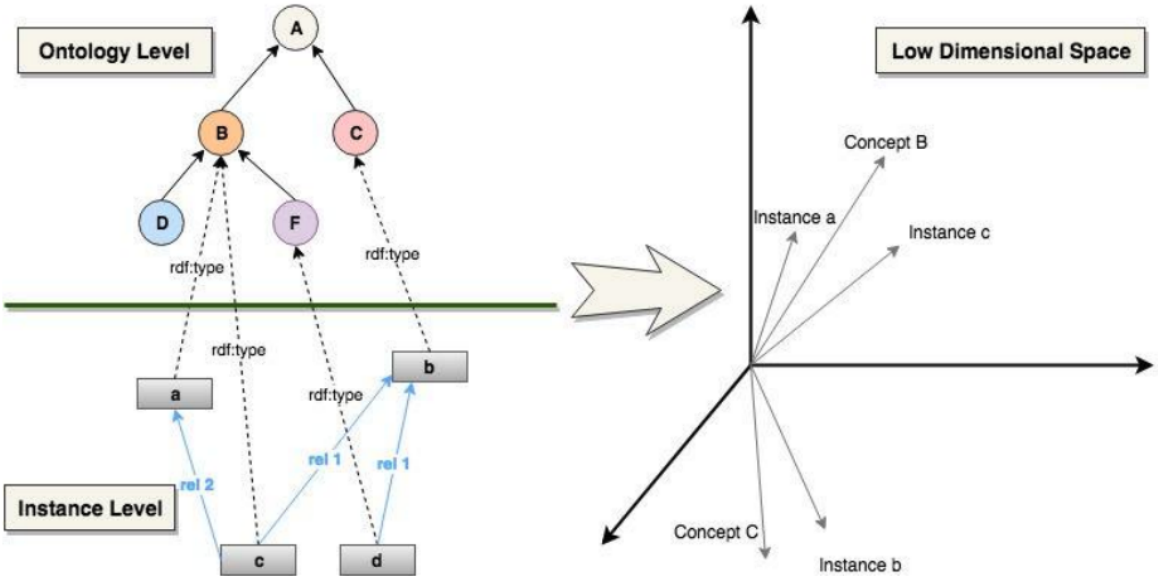


Figure 2.10: Representation of the vectorization of entities in the ontological and instance levels to a low-dimensional space, extracted from [1].

Meanwhile, as knowledge graph embeddings, and more specifically Node Embeddings, become also relevant for data mining purposes, new machine learning

datasets linked to knowledge graph entities arise. In 2016, Ristoski et al. [38] curated 22 datasets from different sources for machine learning tasks, covering classification, clustering, and regression. Subsequently, the GEval framework [29, 30] was introduced to establish a standardized evaluation protocol for these datasets, with DBpedia URIs. In this evaluation protocol, the embeddings of DBpedia entities are used to train and test different downstream algorithms, including not only classification, regression, and clustering, with their typical evaluation metrics such as accuracy or root mean squared error, but also semantic tasks that resemble those mentioned above. These semantic tasks are (i) document similarity, measured by Pearson and Spearman correlations and harmonic mean between, (ii) entity relatedness, measured by Kendall's Tau correlation, and (iii) semantic analogies, measured by accuracy@k. A complete list of GEval datasets is presented in Table 2.2.

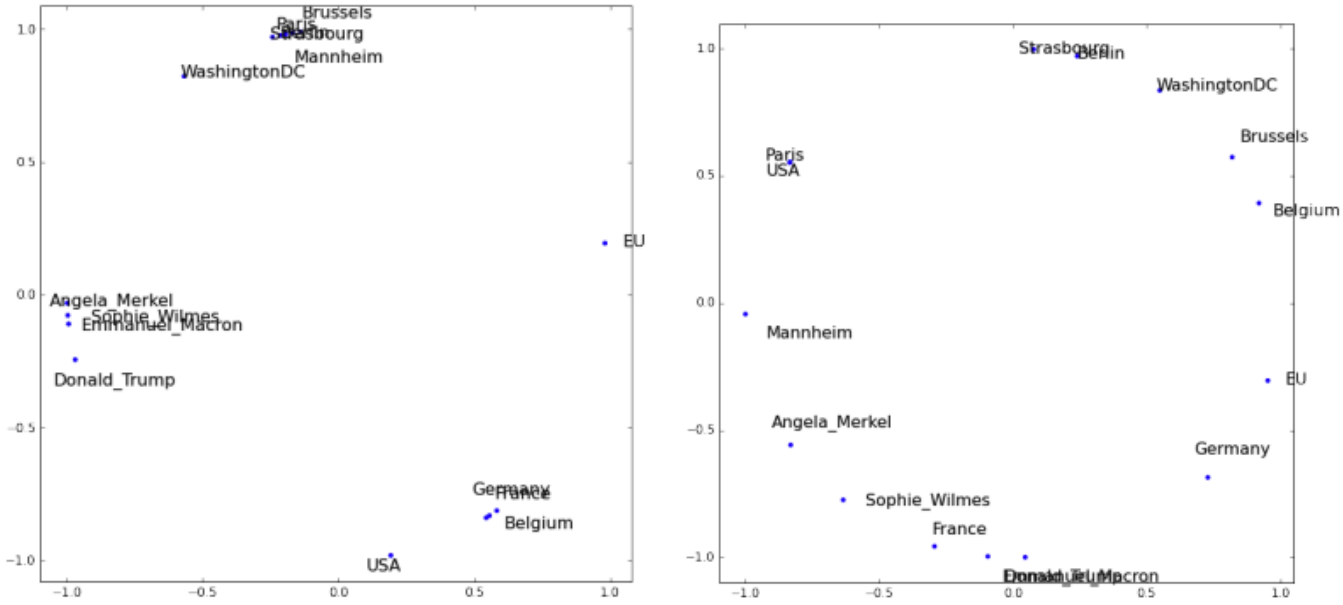
Task	Dataset	Unique entities	Target variable
Classification	Cities	212	3 classes $(67/106/39)$
	AAUP	960	3 classes $(236/527/197)$
	Forbes	1,585	3 classes $(738/781/66)$
	Metacritic Albums	1,600	2 classes $(800/800)$
	Metacritic Movies	2,000	2 classes $(1,000/1,000)$
Regression	Cities	212	numeric $[23, 106]$
	AAUP	960	numeric $[277, 1009]$
	Forbes	1,585	numeric$[0.0, 416.6]$
	Metacritic Albums	1,600	numeric $[15, 97]$
	Metacritic Movies	2,000	numeric $[1, 100]$
Clustering	Cities and Countries (2k)	4,344	2 clusters $(2,000/2,344)$
	Cities and Countries	11,182	2 clusters $(8,838/2,344)$
	Citiess, Movies, Albums, Companies, Uni	6,357	5 clusters $(2,000/960/1,600/212/1,585)$
	Teams	4,206	2 clusters $(4,185/21)$
Document Similarity	LP50	1,225	numeric similarity score $[1.0, 5.0]$
Entity Relatedness	KORE	400	ranking of entities
Semantic Analogies	(All) Capitals and Countries	4,523	entity prediction
	Capitals and Countries	505	entity prediction
	Cities and States	2,467	entity prediction
	Countries and Currencies	866	entity prediction

Table 2.2: List of GEval gold standards, adapted from [34].

Besides having its own datasets and tasks, the GEval Evaluation Framework can also be extended with new gold standard files, and even new tasks. This is relevant considering that its core datasets were produced based on a DBpedia version of 2016, and some target labels may lose consistency over time. In a similar direction, a method is proposed to generate new general-purpose benchmark datasets for the task of link prediction and entity typing [22]. This method uses existing knowledge graphs to create synthetic graphs with similar characteristics, such as the distribution of classes, relations, and instances. Furthermore, in the direction of establishing new node classification datasets with comprehensive tran-test split and expressive test sizes, Bloem et al. [3] introduce kgbench in 2021.

The evaluation methods mentioned above are suitable for evaluating KGEs in specific tasks, but still do not provide a better understanding of what these embeddings learned to represent. In this direction, Portisch et al. [33, 34] recently introduced a novel benchmark for node classification, namely Description Logic Class Constructors (DLCC). DLCC relies on the idea that class separation capabilities can better illustrate how well the model learned to represent vectors of the entities, as shown in Figure 2.11.

In this case, the meaning of class is not restricted to entity type, but as a binary result of a Description Logic constructor. For the generation of gold standards in DLCC, 12 different DL constructors are converted into SPARQL queries, which are used to create fixed gold standard files from DBpedia, as well as to allow the generation of new gold standards. A complete list of these constructors, as well as examples of each of them, are presented in Table 2.3.
In this work, both the GEval [29] and DLCC [33] frameworks are used with their core DBpedia golden standards for the evaluation procedure.



(a) Good class separation

(b) Bad class separation

Figure 2.11: Examples of good and bad entity embeddings regarding class separation capabilities, extracted from [33]. In the left example, cities, countries, and politicians form clear clusters, suggesting that this model successfully learned to differentiate them.

2.3 Compact Representations

The task of learning knowledge graph embeddings as latent representations of the KG itself is an attempt of compacting the semantics behind their sparse tensor in a dense matrix. The fewer dimensions and parameters used to learn its vectors, the more semantic information is condensed in them. But depending on the task they are supposed to solve and the size of the KG, keeping hundreds of dimensions with double precision for each entity can still be costly and inefficient. Two common approaches to achieve even more compact representations of embeddings, but preserving their original information, are dimensionality reduction (DR) and binarization, presented in Sections 2.3.1 and 2.3.2.

DL Type	Test Case	DL Expression	Example
1) Ingoing and Outgoing Relations	tc01	$r.\top$	*Everything that has a location*
	tc02	$r^{-1}.\top$	*Everything that is a location of something*
	tc03	$\exists r.\top \sqcup r^{-1}.\top$	*Everything that has or is a location of something*
2) Relations to particular Individuals	tc04	$\exists R.\{e\} \sqcup R^{-1}.\{e\}$	*Everything that is related to Mannheim*
	tc05	$\exists R_1.(\exists R_2.\{e\}) \sqcup \exists R_1^{-1}.(\exists R_2^{-1}.\{e\})$	*Everything to which Mannheim is related*
3) Particular Relations to Particular Individuals	tc06	$r.\{e\}$	*Movies directed by Steven Spielberg*
4) Qualified Restrictions	tc07	$\exists r.T$	*People married to soccer players*
	tc08	$\exists r^{-1}.T$	*Soccer players married to someone*
5) Cardinality Restrictions of Relations	tc09	$\geqslant 2r.\top$	*People who have at least two citizenship*
	tc10	$\geqslant 2r^{-1}.\top$	*Books written by at least two people*
6) Qualified Cardinality Restrictions	tc11	$\geqslant 2r.T$	*People who have published at least three bestsellers*
	tc12	$\geqslant 2r^{-1}.T$	*Books written by at least two female authors*

Table 2.3: DL class constructions with examples, adapted from [33].

2.3.1 Dimensionality Reduction

Dimensionality reduction techniques are crucial in machine learning and data analysis as they transform high-dimensional datasets into lower-dimensional representations while retaining important information. This is particularly important because high-dimensional datasets can lead to challenges such as increased computational complexity, overfitting, and difficulties in visualization, known as the curse of dimensionality. However, the definition of "important information" varies between different techniques. Principal Component Analysis (PCA), for example, is a widely used method in dimensionality reduction that has remained popular for over a century. It can identify linear combinations of the original features that capture the maximum variance in the data, assuming that this variance represents the essential information that should be preserved. PCA can effectively minimize in-

formation loss, making it a valuable tool for data preprocessing, feature extraction, and visualization of multidimensional data, including KGEs [37]. It can also be used to achieve more compact representations of word embeddings, with little loss in performance when the number of dimensions is reduced by half [35].

Two other prominent dimensionality reduction techniques in machine learning are t-Distributed Stochastic Neighbor Embedding (t-SNE) [48], known for its effectiveness in visualizing high-dimensional data in two or three dimensions, and Uniform Manifold Approximation and Projection (UMAP) [21]. Unlike PCA, t-SNE is nonlinear and focuses on preserving pairwise similarities between data points, assuming that their distances in the multidimensional space are the important information that should be preserved, rather than variance. This makes t-SNE particularly useful in revealing clusters and patterns that may not be apparent in higher-dimensional spaces, providing insight into the intrinsic structure of the data. UMAP, on the other hand, combines topological principles with manifold learning to create low-dimensional representations that preserve both global and local structure in the data. UMAP has gained popularity for its ability to capture complex relationships in high-dimensional datasets, often outperforming traditional methods like t-SNE. However, both are considerably slower than PCA.

2.3.2 Binarization

Aiming for compact representations of KGEs that preserve performance in data mining tasks with way less storage and memory requirements, DR techniques seem not as promising as binary embeddings, which can be better optimized for CPU register sizes and binary operations. Binarization of embeddings consists of converting the vectors of real numbers into vectors of booleans, drastically reducing their memory size. In the context of word embeddings, there are some works in this direction with promising results [24, 28], inspired by the autoencoder architecture proposed by Tissier et al. [45] in 2019.

Several naive approaches to convert real-valued vectors to binary embeddings are possible, such as mapping each real number to 0 if it is negative and to 1 otherwise, or computing the average or median across all dimensions, and mapping real numbers to 1 when they are above the threshold of their respective dimension. In all these cases, each bit could be interpreted as a high/low label for the original real number, and the number of dimensions persists. Although the implementation of these methods is simple, keeping the same number of dimensions can be a relevant drawback when the original dimensions do not match CPU register sizes (32 or 64 bits).

This problem can be tackled by binarization methods such as semantic hashing, which is used to generate concise binary codes with an arbitrary number of bits that effectively approximate nearest-neighbor search in the original space. In 2002, Charikar [8] introduced Locality Sensitive Hashing (LSH), which utilizes random projections to create binary codes that approximate the cosine similarity of the original vectors. Despite their effectiveness, these approaches often do not fully maintain semantic similarities, as highlighted by Xu et al. [51] in 2015.

With semantic preservation in mind, Tissier et al. in [45] proposed training an autoencoder capable of nearly reconstructing the original vectors. This method consists of training a deep learning architecture, that maps one vector x with a binary encoding function $\Phi(x)$ and reconstructs it with a decoding function f with minimal reconstruction loss ℓ_{rec}. The binary embedding $\Phi(x)$ can also have an arbitrary number of bits, but it is critical that it matches CPU register sizes. Its architecture is summarized in the illustration in Figure 2.12. Their results, obtained from experiments conducted on semantic similarity, text classification, and sentiment analysis tasks, showed a decrease of only 2% in accuracy. Meanwhile, this conversion also resulted in a significant reduction in vector size, approximately 97%, and the autoencoder outperformed LSH in semantic tasks.

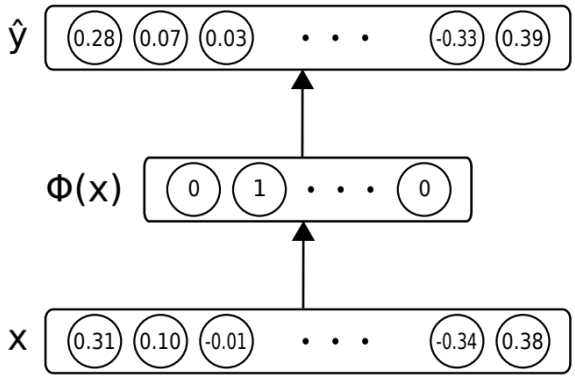


Figure 2.12: Autoencoder architecture, extracted from [45].

More recently, similar results were reported by Navali et al. [24] and Pan et al. [28], with some changes in the autoencoder architecture. Instead of learning to minimize the Euclidean distance (MSE) between the original embeddings and those reconstructed by the decoder, their approaches aimed at minimizing the recon-

struction error of word-to-word relations. For example, given the group of words $(x_\alpha, x_\beta, x_\gamma, x_\delta)$, with their respective continuous embeddings $(a_\alpha, a_\beta, a_\gamma, a_\delta)$, the autoencoder trains to binarize these embeddings into $(b_\alpha, b_\beta, b_\gamma, b_\delta)$. If the words x_α and x_β are semantically close to each other and distant to x_γ and x_δ, the cosine similarity between a_α and a_β in the real vector space is large. Therefore, the Hamming distance between b_α and b_β in the binary vector space should be small, compared to their distances to b_γ and b_δ. All these works achieved considerably compact representations with little loss in performance metrics.

2.4 Related Work

In this thesis, two different research directions converge: the systematic evaluation of knowledge graph embeddings beyond the task of Knowledge Graph Completion, as detailed in Section 2.2.3, and the binarization of word embeddings, detailed in Section 2.3.2.

On the one hand, GEval [29] and DLCC [33] establish themselves as evaluation frameworks to compare different KGE methods in several tasks and capabilities. Evaluations with these frameworks performed in [34, 30] can be considered a relevant baseline for the original KGEs. In [34], the embeddings of RDF2Vec [37], TransE [5], ComplEx [47], DistMult [53], RESCAL [26], RotatE [44], and TransR [17] were evaluated with GEval and DLCC, but exclusively with continuous vectors. In both evaluation frameworks, statistical tests are used to compare the top performers of each task with the remaining models. In GEval, RDF2vec$_{SG}$ and TransE-L2 frequently showed the best performance metrics, while in DLCC the best accuracy scores for the DBpedia gold standard most often belonged to TransE-L2. Still, this work reinforces that different embedding models are more suitable for each task and for each class-separation capabilities.

In the direction of compact representations, dimensionality reduction techniques are frequently applied with visualization purposes in mind, rather than preserving performance metrics in downstream tasks. The latter has been more of a focus for binarization techniques, but in the context of word embeddings. In this sense, the autoencoder architecture to binarize word embeddings proposed by [45] and its corresponding successors [28, 24] is the main inspiration for this work. In these three works, despite some changes in the autoencoding architecture, word embeddings of 300 dimensions were binarized into 64- to 512-bit vectors, sometimes even 640. The compression factor and the loss in performance metrics depended on the number of bits in the output vectors, but often ranged between 30 and 50

times smaller file sizes and between 2 and 5% performance loss. The difference in this case is to binarize the embeddings of URIs from a knowledge graph instead of words from text corpus in natural language and, therefore, to use different tasks and datasets. Furthermore, as in [34], performance in downstream tasks is not observed alone, but also in a comprehensive framework that is also concerned about *what* original vectors learned to represent, and consequently what persisted after the binarization procedure.

Chapter 3

Implementation

3.1 Implementation Overview

In this chapter, the methodological approach to investigate semantic preservation in different variants of binarized knowledge graph embeddings is discussed. To make a fair comparison with meaningful results, two publicly available Evaluation Frameworks (GEval [29] and DLCC [33]) with DBpedia gold standards were used to evaluate 11 flavors of KG embeddings of different natures, which were trained on the same version of DBpedia (2021-09). Each of these 11 original floating-point KG embeddings passed through the same pipeline to generate its binary variants, which is illustrated in Figure 3.1 and detailed in the next sections.

For each of the resulting 55 vector files, several Machine Learning models with different configurations were trained to run GEval's 6 tasks and DLCC's 20 classification test collections. Accuracy scores (in the case of clustering and classification tasks) and other performance metrics are used to assess semantic preservation, together with statistical testing. The aim is not only to conclude which binarization approach performs best, but also to determine whether some types of KG embeddings benefit more from this technique than others, and whether there is an impact on the size of the dataset, the nature of the task, and its complexity.

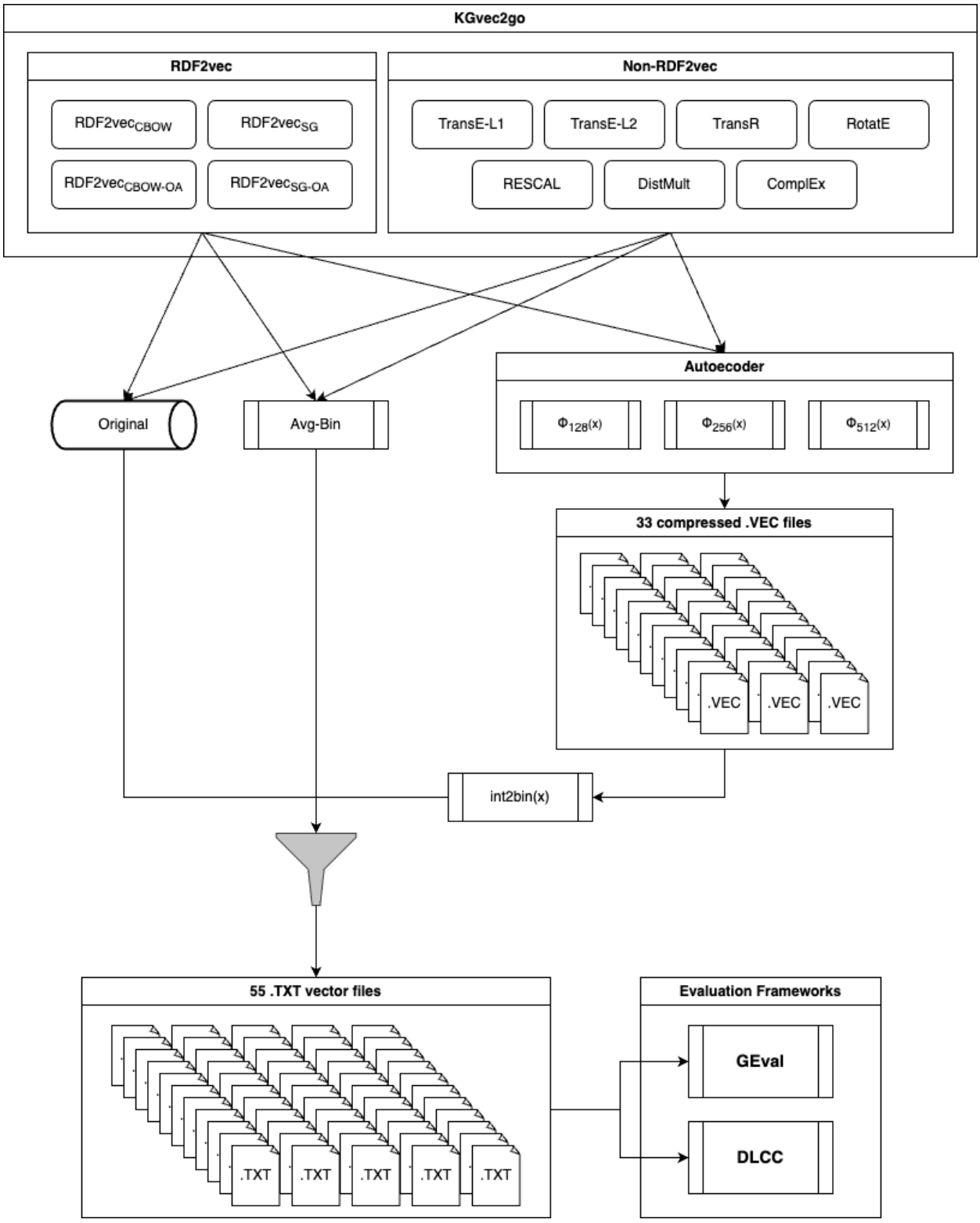


Figure 3.1: Implementation pipeline, summarized in a diagram.

3.2 DBpedia Original Embeddings

Knowledge graph embeddings of DBpedia entities of 11 different types were obtained from the publicly available KGvec2go repository [31][1]. All of them were pre-trained on the same version 2021-09 of DBpedia and contain 200 dimensions per entity. Although the total number of entities in each original file sometimes differs, and so does their file size, nearly 99% of the entities in GEval and DLCC's gold standard files are covered.

Four flavors of RDF2Vec [37] and two flavors (Norms L1 and L2) of TransE [5] embeddings are exploited, as well as ComplEx [47], DistMult [53], RESCAL [26], RotatE [44] and TransR [17] embeddings, all previously described in Section 2.2.2. The original vector files contain nearly 8 million DBpedia entities stored in nearly 18 GB TXT files and are the same as those used in the original DLCC paper [33]. These are downloaded directly from KGvec2go and used to generate the five variants used as input for GEval and DLCC. The so-called ORIGINAL-200 variant is obtained by simply filtering the entities present in GEval and DLCC's gold standard files from the original embeddings, which reduces its file size to approximately 1.6 GB.

A special focus is given to the RDF2vec family of Knowledge Graph Embeddings due to its Word2vec-based architecture. As the autoencoding binarizer proposed by Tissier et al. in [45] successfully preserved the performance of Word2vec embeddings in semantic tasks, RDF2vec embeddings with different flavors of the Word2vec implementation are good candidates for this binarization technique. Therefore, four variations are exploited in the experiments, namely RDF2vec$_{CBOW}$, RDF2vec$_{SG}$, RDF2vec$_{CBOW-OA}$, and RDF2vec$_{SG-OA}$. These four variants, available in KGvec2go, were previously trained for the DLCC paper in 5 epochs with 500 random, duplicate free walks per entity, with a depth of 4, a window size of 5 [33]. The results for orignal embeddings are the upper baseline for comparing binary vectors and should be comparable to those reported in [34] in similar experiments with the same vector files.

3.3 Binarization using Average

Different naive approaches can be used to binarize the original embeddings as a baseline. All positive values could be converted to 1 and all negative values to 0, as suggested by [45], but this method would not account for the distribution of

[1] Vector files in https://data.dws.informatik.uni-mannheim.de/kgvec2go/dbpedia/2021-09/

the embeddings in the vector space. The baseline approach used in this work is to binarize the original embeddings by computing the average of each dimension across all vectors of the full original embedding, and applying the greater-or-equal function to obtain binary vectors.

By doing this, the output binary vectors keep their 200 dimensions, their original distribution is taken into account, and the impact of binarizing with the same number of dimensions can be isolated in comparisons regarding storage and computation efficiency. Therefore, it facilitates the comparison between the original floating-point embeddings and the autoencoded binary variants described in Section 3.4, in which the number of dimensions differs from the original. In experiments, this baseline variant is called AVGBIN-200.

3.4 Binarization using Autoencoding

The main approach proposed in this work to binarize the KG embeddings, while still preserving its semantic information, is to train an autoencoder capable of nearly reconstructing the original vectors, as proposed by Tissier et al. in [45]. A key point of this method, in order to achieve higher semantic preservation and computational performance with minimal storage, is to generate vectors in adequacy with register sizes of a CPU (64, 128 or 256 bits). A C implementation of the *binarizer* proposed by the authors is publicly available on GitHub[2], and it was used with default hyperparameters to create binary vectors of 128, 256 and 512 bits for each embedding type, named AUTO-128, AUTO-256 and AUTO-512 in the experiments, respectively.

In this implementation, it should be noted that the compressed output file .VEC of the *binarizer* consists of each entity followed by 64-bit integers. To obtain TXT binary vectors in the format expected by the evaluation frameworks, these integers need to be converted to base 2, filled with zeros on the left when necessary, and then concatenated. For example, the following line in the .VEC file should be converted to the one below with its 128 bits.

```
http://dbpedia.org/resource/Rio_De_Janeiro 88893013957
2942997 9355426029144257952

http://dbpedia.org/resource/Rio_De_Janeiro 0 0 0 0 1 1
0 0 0 1 0 1 0 1 1 0 0 0 0 1 1 1 0 1 0 1 0 0 0 0 0 0 1
```

[2] See https://github.com/tca19/near-lossless-binarization

```
0 0 0 1 1 1 1 0 1 1 0 0 0 1 0 0 0 1 1 1 0 0 0 1 0 0 1
0 1 0 1 1 0 0 0 0 0 0 1 1 1 0 1 0 1 0 1 0 0 1 0 0 1 1
0 0 1 1 0 1 0 0 0 0 0 0 0 0 1 0 1 1 0 0 1 0 1 1 0 0 0
1 1 0 1 0 1 1 0 1 0 0 0 0 0
```

3.5 Evaluation Frameworks and Entity Filtering

To optimize storage during the pipeline, only the entities existing in the GEval and DLCC gold standard files were kept in the final embedding files. GEval contains 20 gold standard files that cover nearly 23 thousand entities. DLCC, on the other hand, covers more than 340 thousand entities in its 60 datasets. The union of both sets corresponds to 395,033 entities, slightly more than 4% of the entities in the original files downloaded from KGvec2go and the generated binary variants. The final .TXT files are then used as input for both evaluation frameworks in the final step of the pipeline.

The GEval evaluation framework, as proposed in 2020 [29], consists of evaluating KG embeddings on 3 data mining tasks (classification, clustering, and regression) and 3 semantic tasks (document similarity, entity-relatedness and semantic analogies), with the datasets listed in Table 2.2. The Python implementation and original DBpedia gold standards available on GitHub[3] were used, mostly with default parameters. For clustering and document similarity, the Manhattan distance was chosen instead of the cosine distance, because this operation can be computationally optimized for binary vectors in future applications. In the Semantic Analogies task, the number of k neighbors to search for the correct analogies was set to 10.

The Description Logic Class Constructors' evaluation framework [33], on the other hand, relies on the idea that class separation capabilities can better illustrate how good the vector representations of the entities are. Therefore, the framework proposed in 2022 is more focused on the task of binary classification, with 12 test collections and 6 different types of entity, each using a different description logic constructor. The embeddings were evaluated using the original Python implementation[4] and DBpedia gold standard files[5] available on GitHub. When finished, both evaluation frameworks produce several files with the resulting metrics for each model in each dataset. These files are analyzed and discussed in Chapter 4.

[3] See https://github.com/mariaangelapellegrino/Evaluation-Framework
[4] See https://github.com/janothan/dl-evaluation-framework
[5] See https://github.com/janothan/DL-TC-Generator

Chapter 4

Experimental Evaluation

4.1 Compression Factor

As described in Chapter 3, the first step of the experimental part consists of obtaining all 55 vector files in the same format expected by the GEval and DLCC Evaluation Frameworks. All these vector files cover nearly 99% of the DBpedia entities used in GEval and DLCC's gold standard datasets, and vector files of the same variant have rather similar file sizes, which is mostly determined by the number of dimensions and whether the vector is continuous or binary. A comparison of the file sizes of the .VEC and .TXT files of each embedding variant, as well as the compression factor relative to the original files, is presented in Table 4.1.

Embedding variant	Compact .VEC files		Full vector .TXT files	
	Average file size (MB)	Average compression factor	Average file size (MB)	Average compression factor
original-200	-	-	1607.5	1
avgbin-200	-	-	174.5	9
auto-128	32.9	**49**	112.6	**14**
auto-256	47.8	34	207.4	8
auto-512	78.3	21	396.6	4

Table 4.1: Average file sizes and compression factors relative to original-200 of the 33 .VEC files and 55 .TXT vector files, after entity filtering.

It can be seen that vector files of 200 dimensions become 9 times smaller when converted from continuous to binary and preserving dimensionality (as for AVGBIN-200). As expected, the compression factors of binary vectors obtained with autoen-

coding do not represent a great advantage in terms of TXT files, when compared to the naive approach, but even binary vectors with more than double dimensions than originally (AUTO-512) still reduce file size by 4 times.

It should also be noted that, in terms of storage, the great advantage of using binary vectors that match CPU register sizes is the usage of compact .VEC files. By encoding 64 bits in a single integer, these files achieve considerably higher compression factors, ranging from 20 to 50 times when using 128 to 512 bits. Similar compression factors were reported in [45, 24, 28].

4.2 GEval Evaluation Framework

The GEval Evaluation Framework covers six tasks, some of them with multiple datasets, as described in Table 2.2. Each task is analyzed separately in sections 4.2.1 to 4.2.6. For tasks with multiple datasets, dataset-specific results are available in Appendix B.1.

4.2.1 Classification Results

The classification task in GEval is performed in 5 gold standard files and measured by accuracy. When comparing the accuracy of each binary variant with their respective original vector, the relative accuracy loss is calculated, and a one-sided binomial test is conducted to check whether the difference is significant. Figure 4.1 shows a boxplot of the relative accuracy loss for each binarization technique in all data sets. It can be observed that the autoencoder with 512 bits was the technique that best preserved the accuracy, losing 3 to 5% on average. In contrast, the autoencoder with 128 bits showed a more expressive accuracy loss, of 4 to 8%, while the two remaining techniques were tied with similar distributions around 5%.

This behavior is also clear in Table 4.2, which indicates that AUTO-512 binary embeddings performed just as well as their original versions in half of the cases, with differences that are not statistically significant. It is also noted in Table 4.2 that both classic RDF2vec flavors (CBOW and Skipgram) benefit more from this technique than KGEs of other nature. In particular, the RDF2vec$_{SG}$ vectors binarized with autoencoding to 512 bits performed just as well as the original RDF2vec$_{SG}$ vectors in the 5 datasets.

It should also be noted that the best accuracy scores belonged to RDF2vec or TransE vectors for all classification datasets. A curious example occurred in the

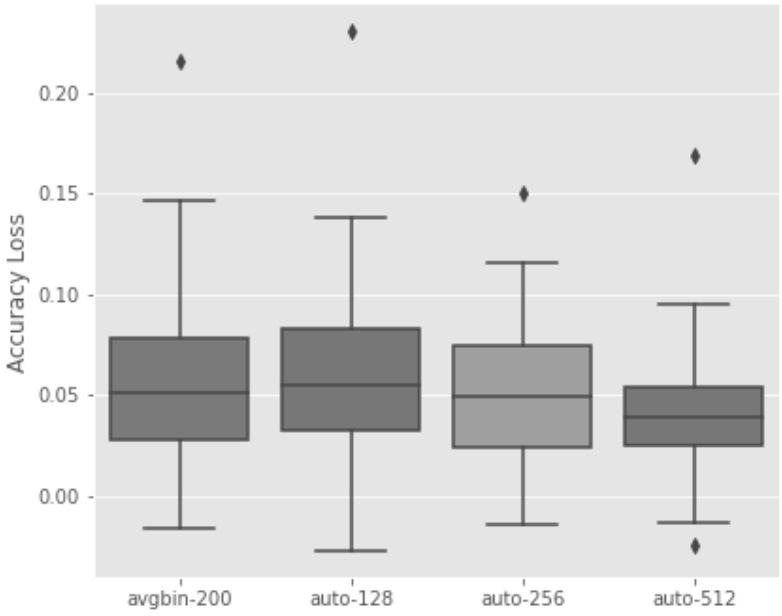


Figure 4.1: Relative accuracy loss by binarization technique in GEval classification.

Cities dataset (Table B.4), in which all binary variants of RDF2vec$_{SG}$ embeddings performed as well as the original, and the variant AUTO-512 scored even better. Figure 4.2 shows the class separation of four variants in this scenario. In all four cases, the principal component (PC-0) seems to explain the target variable to some extent, and the class separation between labels 'high' and 'low' is slightly clearer in the variants ORIGINAL-200 and AUTO-512. A complete list of accuracy scores and the distribution of accuracy loss in GEval's classification datasets is presented in Section B.1.1.

4.2.2 Regression Results

The regression task in GEval is performed in the same five gold standard files as in classification, but the target variable is numerical and the performance is measured by root mean square error (RMSE). When comparing the RMSE of each binary variant with their respective original vector, the relative gain of RMSE is calculated and a one-sided F test is conducted to check whether the increase in MSE

	avgbin-200	auto-128	auto-256	auto-512	**Average**
RDF2vec$_{CBOW}$	**4**	3	**4**	2	3.3
RDF2vec$_{CBOW-OA}$	**3**	1	1	**3**	2.0
RDF2vec$_{SG}$	3	3	3	**5**	**3.5**
RDF2vec$_{SG-OA}$	1	3	3	**4**	2.8
RESCAL	**2**	1	**2**	**2**	1.8
DistMult	**1**	0	**1**	**1**	0.8
ComplEx	0	**1**	0	0	0.3
TransE-L1	2	1	2	**3**	2.0
TransE-L2	1	**3**	2	2	2.0
TransR	1	1	1	**4**	1.8
RotatE	**1**	0	**1**	**1**	0.8
Average	1.7	1.5	1.8	**2.5**	1.9

Table 4.2: Count of GEval classification datasets in which the best classifier of each binary embedding variant did not significantly underperform the original one in accuracy. The closer to 5, the less performance loss. $\alpha = 0.05$.

is significant. In contrast to the classification results, the best regressors were often those trained on binary vectors autoencoded in 128 bits, as shown in Figure 4.3, while AUTO-512 performed worse. Furthermore, AVGBIN-200 vectors also appeared to preserve performance better than AUTO-256 ones. This can be partially explained by the sensitivity of regression tasks to high-dimensional inputs, which is critical in smaller datasets, such as the Cities gold standard. This particular dataset contains slightly more than 200 observations, so variants with 256 and 512 dimensions tend to always overfit.

Another breakdown is presented in Table 4.3, with the number of datasets where binary variants scored as good or better than the original real-numbered vectors. The binary variants with fewer dimensions consistently outperformed binary variants with more dimensions, and for RotatE and TransE with L1-norm regularization it was even the case that no gain in RMSE is observed in any dataset.

Notable cases were observed in the Forbes and MetacriticAlbums datasets, with scores available in Tables B.8 and B.9, respectively. In these cases, binary variants with 128 bits frequently achieved even better RMSE scores than the original vectors. This suggests that regressors may effectively benefit from dimensionality reduction in certain situations. A complete list of accuracy scores and the distribu-

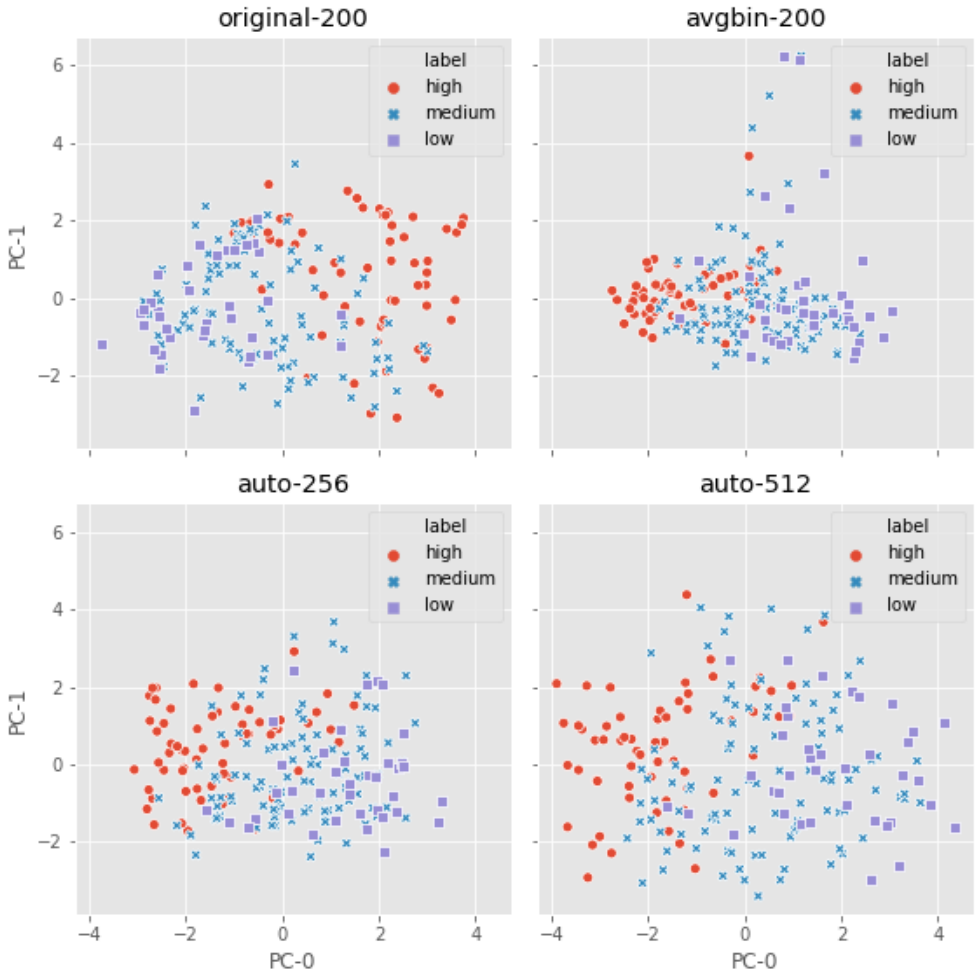


Figure 4.2: Class separation of RDF2vec$_{SG}$ variants in dataset Cities, visualized in 2 dimensions using PCA.

tion of RMSE gain in GEval's regression datasets is presented in Section B.1.2.

4.2.3 Clustering Results

The clustering task in GEval is performed in 4 gold standard files and measured by clustering accuracy. These datasets resemble certain DLCC test collections, since they basically consist of separating types of entities (e.g., cities from countries and movies from albums), but here only unsupervised algorithms are used. When com-

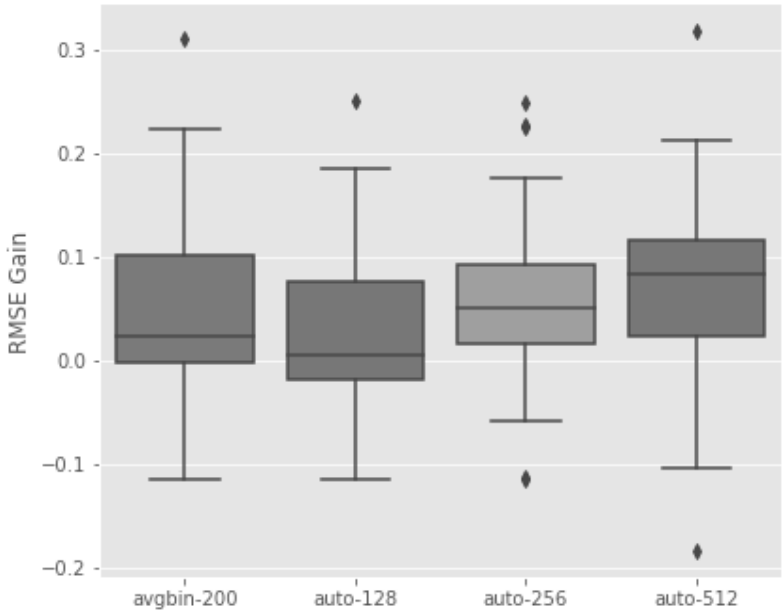


Figure 4.3: Relative RMSE gain by binarization technique in GEval regression.

paring the clustering accuracy of each binary variant with their respective original vector, the relative accuracy loss is calculated, and a one-sided binomial test is conducted to check whether the difference is significant, similarly to that for classification. Figure 4.4 shows a boxplot of the relative clutering accuracy loss for each binarization technique in all datasets.

For all binary variants, there is a considerable amount of outliers on both directions, and this can be partially explained by the nature of unsupervised learning, that may sometimes not converge at all. When this is the case with binary variants, their accuracy score tends to be extremely lower than the original vectors, and vice versa. Most of these extreme values come from the 5-cluster dataset Cities-Movies-Albums-Companies-Uni (see Figure B.12). Looking at Table 4.4, it can be seen that all binarization techniques performed similarly well, with scores that are not significantly less than the original ones in half of the cases.

Specifically for RDF2vec$_{CBOW-OA}$ and RDF2vec$_{SG}$, binary vectors performed just as well and sometimes even better than their respective original variants in all datasets. An example in which binary variants of RDF2vec$_{CBOW-OA}$ embeddings

	avgbin-200	auto-128	auto-256	auto-512	**Average**
RDF2vec$_{CBOW}$	**4**	**4**	3	1	3.0
RDF2vec$_{CBOW-OA}$	**4**	**4**	2	1	2.8
RDF2vec$_{SG}$	**4**	**4**	1	2	2.8
RDF2vec$_{SG-OA}$	3	**4**	1	2	2.5
RESCAL	**4**	2	1	1	2.0
DistMult	1	**4**	1	0	1.5
ComplEx	2	**3**	1	0	1.5
TransE-L1	**5**	**5**	3	0	**3.3**
TransE-L2	**3**	**3**	0	0	1.5
TransR	**4**	**4**	3	1	3.0
RotatE	**5**	**5**	3	0	**3.3**
Average	3.5	**3.8**	1.7	0.7	2.4

Table 4.3: Count of GEval regression datasets in which the best regressor of each binary embedding variant did not significantly underperform the original one in RMSE. The closer to 5, the less performance loss. $\alpha = 0.05$.

	avgbin-200	auto-128	auto-256	auto-512	**Average**
RDF2vec$_{CBOW}$	**2**	**2**	**2**	**2**	2.0
RDF2vec$_{CBOW-OA}$	**4**	**4**	**4**	**4**	**4.0**
RDF2vec$_{SG}$	3	**4**	**4**	**4**	3.8
RDF2vec$_{SG-OA}$	2	2	**3**	**3**	2.5
RESCAL	**1**	0	0	**1**	0.5
DistMult	1	2	2	**3**	2.0
ComplEx	**2**	1	**2**	1	1.5
TransE-L1	**2**	**2**	**2**	**2**	2.0
TransE-L2	**3**	2	2	2	2.3
TransR	**3**	2	2	2	2.3
RotatE	**2**	0	1	**2**	1.3
Average	2.3	1.9	2.2	**2.4**	2.2

Table 4.4: Count of GEval clustering datasets in which the best clusterer of each binary embedding variant did not significantly underperform the original one in clustering accuracy. The closer to 4, the less performance loss. $\alpha = 0.05$.

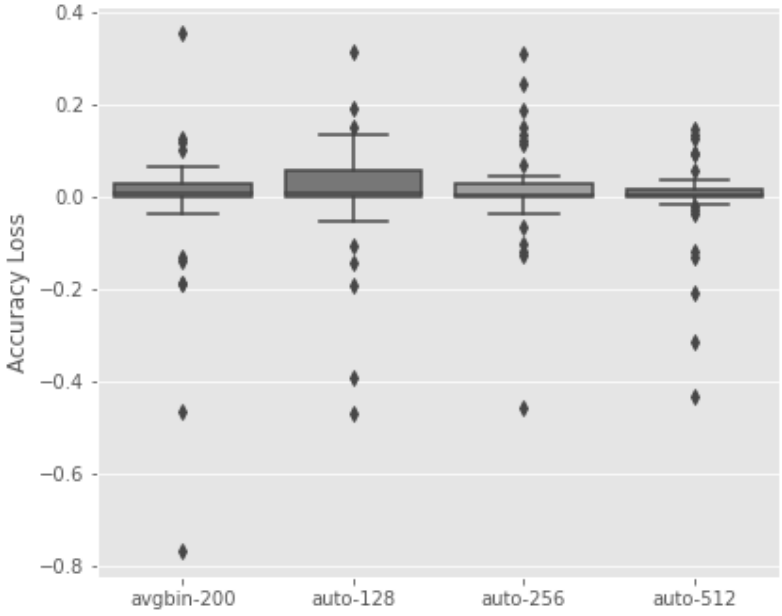


Figure 4.4: Relative accuracy loss by binarization technique in GEval clustering.

performed considerably better than originals was in the Cities-Countries dataset (Table B.13). In this dataset, the variant AUTO-512 achieved the best score among all embeddings, surpassing even the dataset-champion TransE-L2 [34]. The clear class separation of RDF2vec$_{CBOW-OA}$ variants in this particular dataset is shown in Figure 4.5. The complete list of accuracy scores and the distribution of accuracy loss in GEval's clustering datasets is presented in Section B.1.3.

4.2.4 Document Similarity Results

The document similarity task in GEval is performed on a single gold standard dataset, the LP50, and is measured by the Pearson and Spearman correlations, which can be summarized with their harmonic mean. Figure 4.6 shows that the Pearson correlation is frequently higher than Spearman's, and binary variants autoencoded in 128 and 512 bits generally show higher correlations.

Granular results for the harmonic mean between Pearson and Spearman correlations are presented in Table 4.5. There, it is more visible that variants AUTO-128

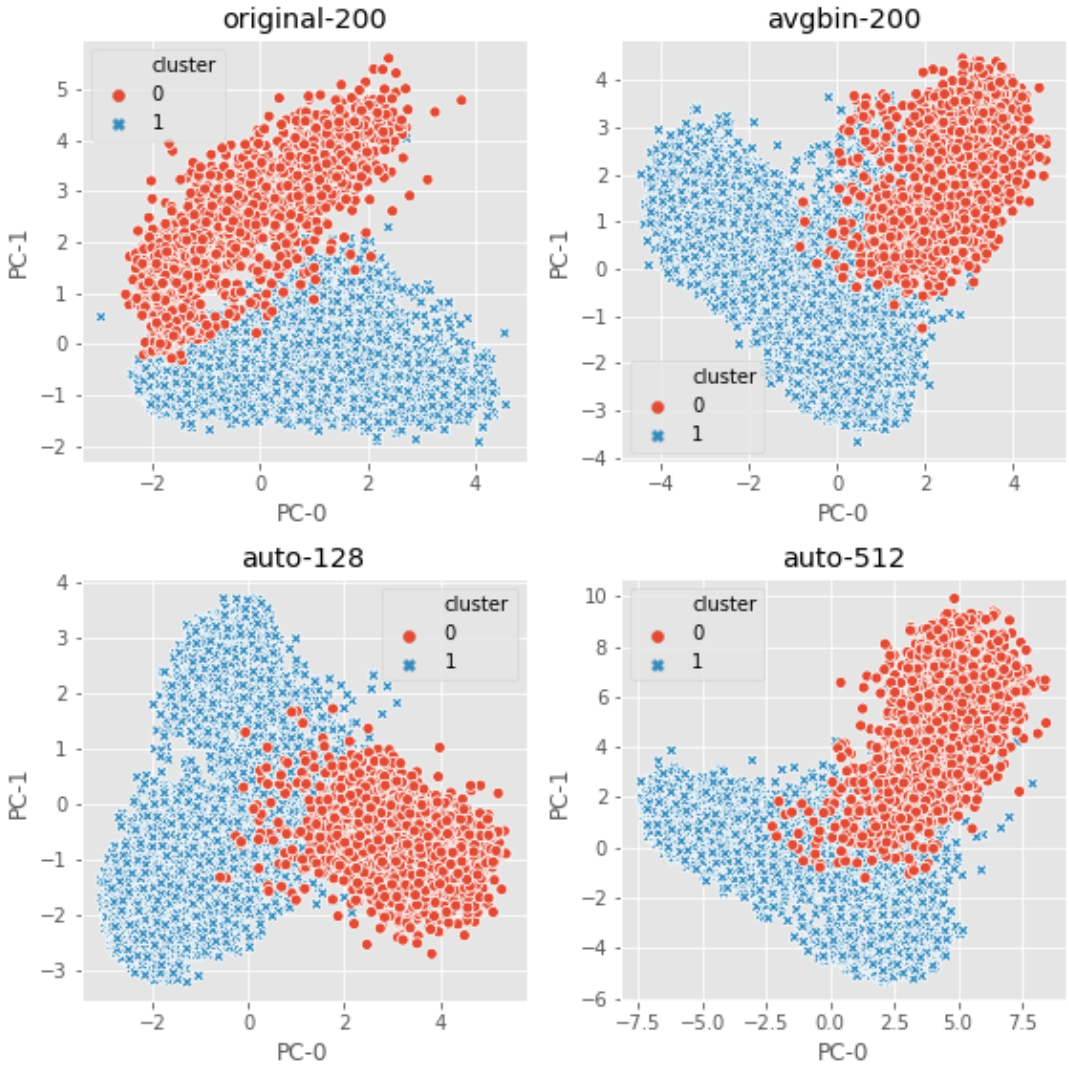


Figure 4.5: Class separation of RDF2vec$_{CBOW-OA}$ variants in dataset Cities and Countries, visualized in 2 dimensions using PCA.

achieved higher correlations in all flavors of RDF2vec, while variants AUTO-512 achieved it in edge reconstruction-based techniques.

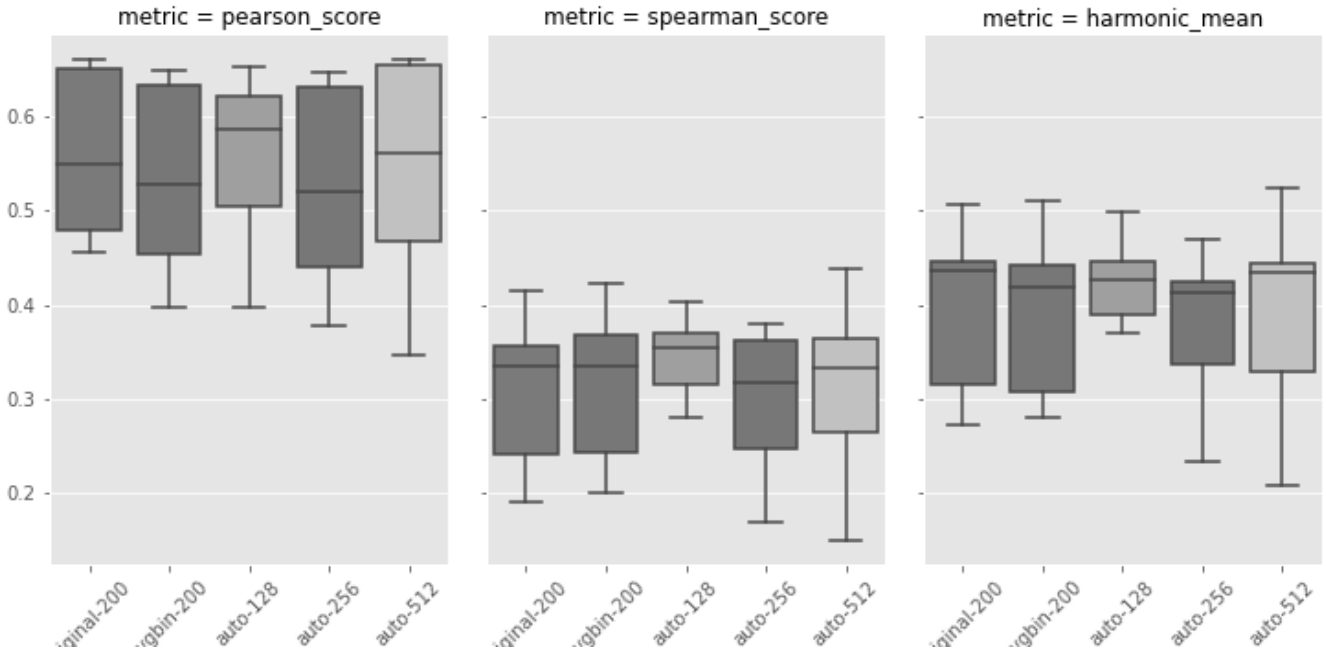


Figure 4.6: Pearson and Spearman correlations and harmonic mean between them by binarization technique in GEval document similarity.

	original-200	avgbin-200	auto-128	auto-256	auto-512
RDF2vec$_{CBOW}$	0.352	0.316	**0.374**	0.373	0.349
RDF2vec$_{CBOW-OA}$	0.273	0.280	**0.370**	0.234	0.209
RDF2vec$_{SG}$	0.278	0.281	**0.499**	0.301	0.286
RDF2vec$_{SG-OA}$	0.275	0.297	**0.462**	0.246	0.308
RESCAL	0.408	**0.484**	0.427	0.469	0.427
DistMult	**0.445**	0.380	0.384	0.378	0.435
ComplEx	**0.442**	**0.442**	0.396	0.414	**0.442**
TransE-L1	0.435	0.418	0.426	0.420	**0.445**
TransE-L2	**0.447**	0.443	0.418	0.426	0.434
TransR	0.506	0.511	0.450	0.453	**0.525**
RotatE	0.458	0.441	0.440	0.424	**0.464**

Table 4.5: Harmonic mean scores for best model of each embedding variant in the LP50 dataset.

4.2.5 Entity Relatedness Results

Similar as for document similarity, the GEval task of entity relatedness is performed in a single gold standard, the KORE dataset, and is measured by the Kendall's Tau correlation between the nearest neighbors of certain entities in the embedding vector space and the actual ones in the dataset. This task is particularly sensible to the number of entities covered in the vector files and to missing entities, but they contain a similar number of entities, and all vector files achieved complete coverage in the KORE dataset. Although the results are not directly comparable to those presented in [34], Table 4.6 also shows RDF2vec embeddings as best performers in this task. It can also be noted that binary variants achieved higher correlations in most cases, but without a visible pattern.

	original-200	avgbin-200	auto-128	auto-256	auto-512
RDF2vec$_{CBOW}$	0.290	0.311	0.287	**0.315**	**0.315**
RDF2vec$_{CBOW-OA}$	0.121	0.128	**0.155**	0.152	0.095
RDF2vec$_{SG}$	**0.504**	0.468	0.417	0.459	0.478
RDF2vec$_{SG-OA}$	**0.419**	0.348	0.339	0.384	0.373
RESCAL	**0.188**	0.183	0.123	0.128	0.123
DistMult	0.130	0.138	0.164	**0.178**	0.153
ComplEx	**0.234**	0.218	0.208	0.208	0.223
TransE-L1	0.095	0.128	0.160	0.137	**0.189**
TransE-L2	0.061	**0.128**	0.047	0.069	0.063
TransR	0.227	0.213	0.249	0.217	**0.270**
RotatE	0.061	0.149	**0.195**	0.170	0.182

Table 4.6: Kendall's Tau correlation scores for each embedding variant in the KORE dataset.

4.2.6 Semantic Analogies Results

The task of semantic analogies in GEval is performed in 4 gold standard files, where analogies of type *"A is to B as C is to..."* are used to find *D*, in a list of *k* candidates. Then it is measured by precision@k, although it is sometimes also referred to as accuracy [29, 30, 34]. This task is typically best solved by KGEs that are trained to preserve translational properties, such as RDF2vec and TransE. Similarly to classification and clustering, precision@10 scores for each binary variant are compared with their respective original vector, the relative precision loss is

calculated, and one-sided binomial tests are performed to check whether the difference is significant.

Figure 4.7 shows the loss of precision@10 for each binarization technique, with AVGBIN-200 and AUTO-512 as the methods that better preserve performance in this semantic task. However, semantic preservation was not as expressive as in previous tasks, with frequent losses of 10% or more. This suggests that, although single bits of binary vectors may convey relevant information for separating classes, many of the translational properties from the continuous vectors are lost.

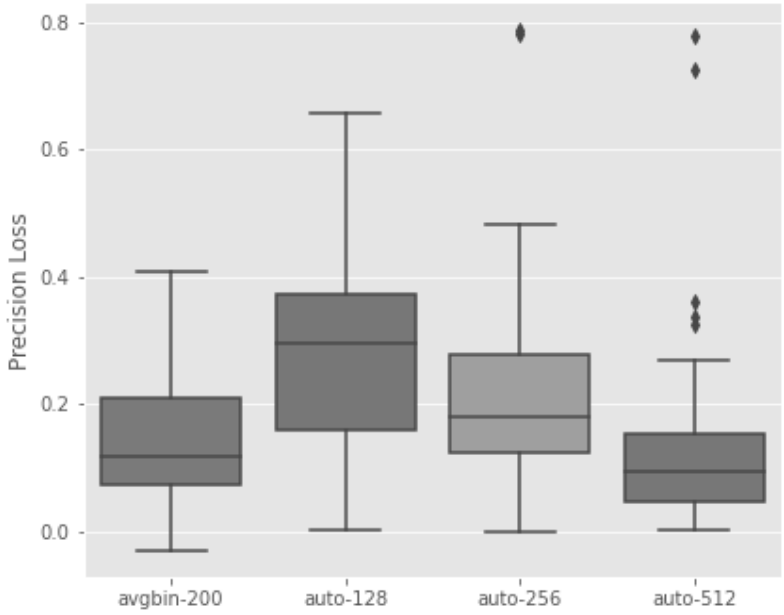


Figure 4.7: Relative precision loss by binarization technique in GEval semantic analogies.

This behavior is also visible in Table 4.7, which shows that in the vast majority of cases there is a significant loss in precision@10. The only embedding that often preserved its original performance is the matrix-factorization-based RESCAL, but it is notably the easiest baseline to surpass in most datasets. In contrast, binary variants of RDF2vec$_{SG}$, TransE-L2 and TransR surprisingly achieved high scores in the All-Capital-Countries dataset (see Table B.18).

	avgbin-200	auto-128	auto-256	auto-512	**Average**
RDF2vec$_{CBOW}$	0	0	0	0	0
RDF2vec$_{CBOW-OA}$	0	0	0	0	0
RDF2vec$_{SG}$	**1**	0	0	**1**	0.5
RDF2vec$_{SG-OA}$	**1**	0	0	**1**	0.5
RESCAL	3	2	2	**4**	**2.8**
DistMult	1	1	1	**2**	1.3
ComplEx	0	0	**1**	**1**	0.5
TransE-L1	0	0	0	0	0
TransE-L2	**1**	0	0	0	0.3
TransR	1	0	0	**2**	0.8
RotatE	**1**	**1**	**1**	**1**	1
Average	0.8	0.4	0.5	**1.1**	0.7

Table 4.7: Count of GEval Semantic Analogies datasets in which the best model of each binary embedding variant did not significantly underperform the original one in precision@10. The closer to 4, the less performance loss. $\alpha = 0.05$.

To better illustrate how the translational properties of the embeddings are affected after binarization, the embeddings of certain capitals and countries from two semantic analogy datasets are presented in two dimensions in Figure 4.8. This figure shows variants of TransE with L2-norm regularization, which is originally able to solve the Cities and Countries dataset with perfect precision@10.

In this particular example, binary variants performed well, but it is still possible to see that the original embeddings show a clearer pattern of translations, which was only decently preserved by embeddings binarized in 512 bits. The complete list of scores and the distribution of precision loss in GEval's semantic analogy datasets is presented in Section B.1.4.

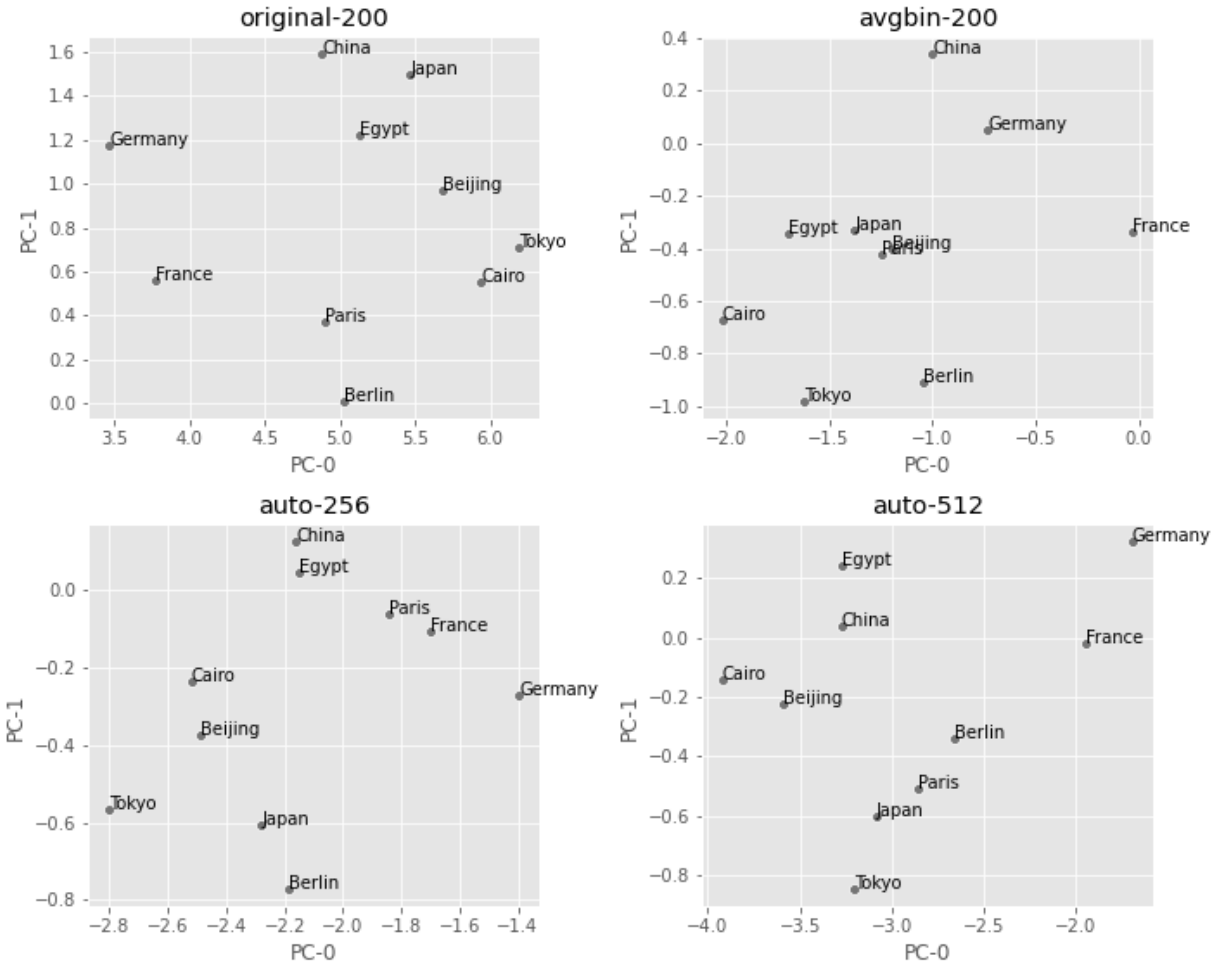


Figure 4.8: TransE-L2 embedding variants of selected capitals and countries from Semantic Analogies' dataset Cities and Countries, visualized in 2 dimensions using PCA.

4.3 DLCC Evaluation Framework

In contrast to GEval, the DLCC framework focuses exclusively on the task of classification, separating entities based on several heuristics. Here, the DLCC results are presented similarly as in Sections 4.2.1 and 4.2.3, with relative accuracy loss and one-sided binomial tests as the main metrics, but with special attention to the class constructors listed in Table 2.3 to understand what is learned and what is lost.

The DBpedia benchmark consists of 20 test collections of 12 different class constructors, and datasets of sizes 50, 500 and 5000. Therefore, there are 60 datasets in total.

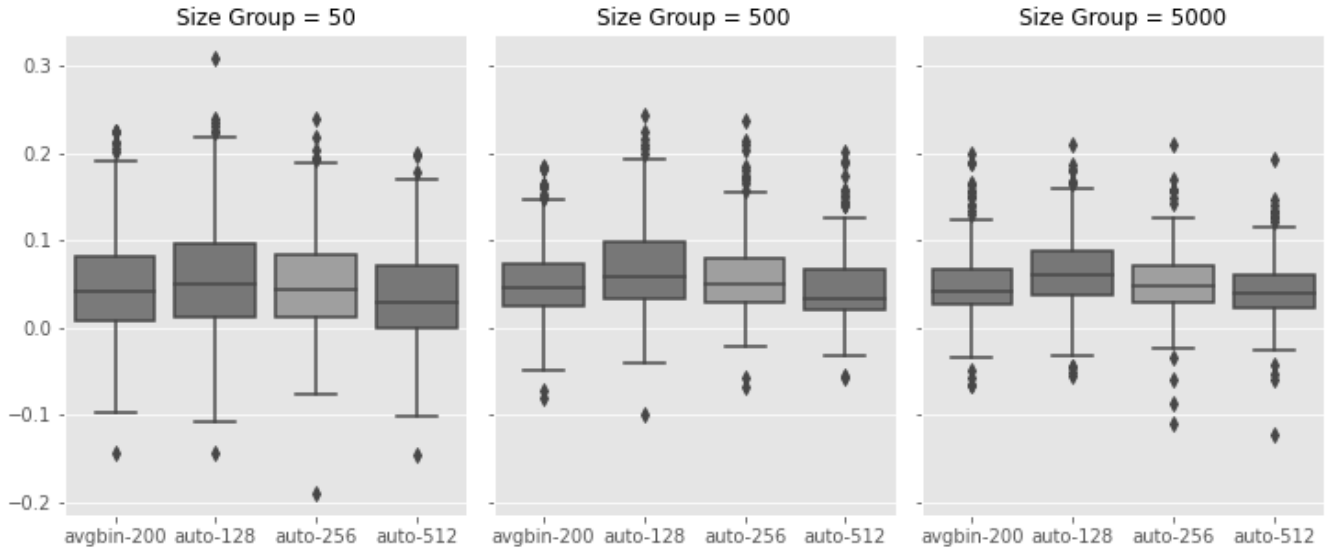


Figure 4.9: Relative accuracy loss by size group and embedding variant in DLCC.

Figure 4.9 shows the relative accuracy loss for each binarization method compared to their respective original vectors. Interestingly, the size of the dataset does not affect how much accuracy is lost when binarizing embeddings, but rather the variance of its distribution around the mean. The general behavior is similar to GEval's classification datasets (Section 4.2.1), in the sense that AUTO-512 also achieves better semantic preservation (3 to 6%), while AUTO-128 is again the worst (4 to 9%). Situations where binary embeddings outperformed their original versions were also observed for all binarization methods, although not very often.

In terms of observing insignificant accuracy losses, Table 4.8 confirms that AUTO-512 achieved expressive semantic preservation, with losses that are not significant in almost half of the cases. It was also consistently the binarization method that best preserved accuracy for 10 of 11 embedding models.

It should be noted that most of the cases where binary vectors performed as well as

	avgbin-200	auto-128	auto-256	auto-512	**Average**
RDF2vec$_{CBOW}$	**37**	22	27	29	28.8
RDF2vec$_{CBOW-OA}$	33	29	28	**34**	**31.0**
RDF2vec$_{SG}$	27	22	30	**34**	29.8
RDF2vec$_{SG-OA}$	23	23	25	**27**	24.5
RESCAL	21	**27**	21	**27**	24.0
DistMult	13	17	15	**19**	16.0
ComplEx	14	18	23	**24**	19.8
TransE-L1	22	17	20	**25**	21.0
TransE-L2	27	22	26	**31**	26.5
TransR	21	20	17	**28**	21.5
RotatE	28	24	25	**30**	26.8
Average	24.2	21.9	23.4	**28.0**	24.4

Table 4.8: Count of DLCC Test Collections of all size groups (out of 60) in which the best classifier of each binary embedding variant did not significantly underperform the original one in accuracy. $\alpha = 0.05$.

their original versions occurred in smaller datasets because the size of the dataset influences the statistical test. The binomial test becomes more powerful with an increase in the number of observations, and, consequently, more capable of distinguishing small differences. Therefore, datasets of size 5000 provide a better understanding of what the embeddings learned to represent and what is preserved after binarized. The accuracy scores for individual test collections of size 5000 are presented in Section B.2.

4.3.1 Ingoing and Outgoing Relations

The class separation in terms of the ingoing and outgoing relations corresponds to the DLCC test collections tc01, tc02 and tc03. Classifiers should be able to differentiate, for example, people who have children from people who do not have children (at least not in DBpedia). The relative accuracy loss for these test collections is presented side by side in Figure 4.10.

Figure 4.10 shows similar behavior in the three test collections, with AUTO-512 being the only binarization method that achieves, on average, a loss of accuracy of less than 5%. The binary variant with fewer dimensions also consistently underperformed other binarization methods. It should be noted that the accuracy losses

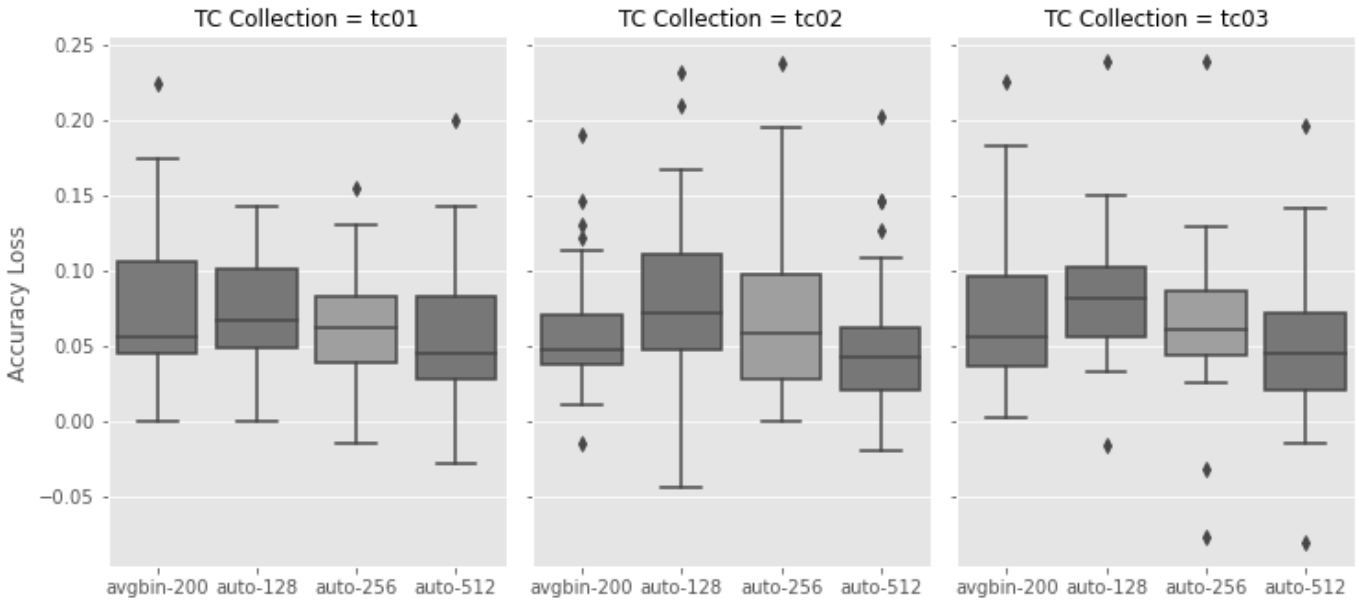


Figure 4.10: Relative accuracy loss for DLCC Test Collections tc01, tc02 and tc03.

are significant in all cases for group size 5000, except for RotatE binarized with AVGBIN-200 in tc02, which also had the worst performance among original embeddings. Therefore, it is possible to say that binarization with AVGBIN-200 and AUTO-512 significantly sacrifices accuracy regarding both ingoing and outgoing relations, but still performs reasonably well.

4.3.2 Relations to Particular Individuals

The class separation in terms of the relationships with particular individuals corresponds to the DLCC test collections tc04 and tc05. Classifiers should be able to differentiate, for example, cities that have some sort of relationship to the United States, such as being located in this country, from cities that do not have any relationship with this particular entity. The relative accuracy loss for both test collections is presented side by side in Figure 4.11.

Compared to previous test collections, binary embeddings seem to preserve this type of knowledge better, with accuracy losses that frequently range between 0 and 5%. Binarization methods that increase the number of dimensions achieved better results, AUTO-512 being again the best performer and having competitive

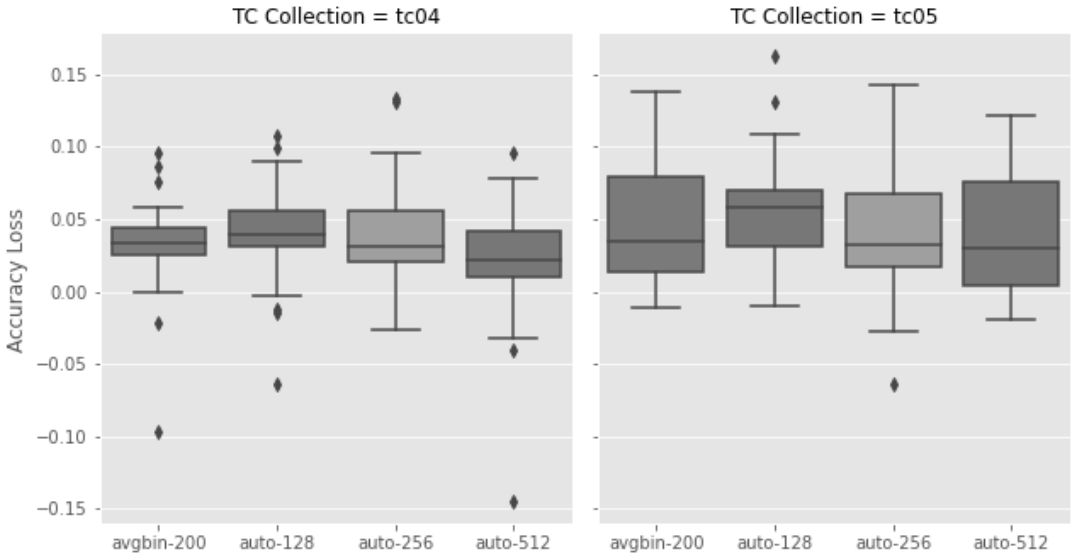


Figure 4.11: Relative accuracy loss for DLCC Test Collections tc04 and tc05.

scores (0.978 with ComplEx in Table B.22 and 0.982 with RDF2vec$_{SG-OA}$ in Table B.23). It can also be seen that tc05 showed a larger variance than tc04, which can be explained by the wider scope of its task. For example, while in tc04 the positive instances for cities are directly related to the United States, in tc05 the positive instances for cities are related to some intermediary entity, which is directly related to New York City. Although the differences are statistically significant in most cases, it can be said that binarization methods that increase the number of dimensions preserve both kinds of information well.

4.3.3 Particular Relations to Particular Individuals

Particular relations to particular individuals correspond to the DLCC test collection tc06, and can be exemplified by separating movies that are produced by Universal Studios from movies that are not. The relative accuracy loss for this specific test collection is presented in Figure 4.12.

The behavior and conclusion for this test collection are similar to that observed for tc04 and tc05, in which most losses in accuracy range between 0 and 5%, and more dimensions helps to better preserve this kind of information. When using AUTO-512, for example, only 3% of accuracy is lost on average. For the tc06 dataset

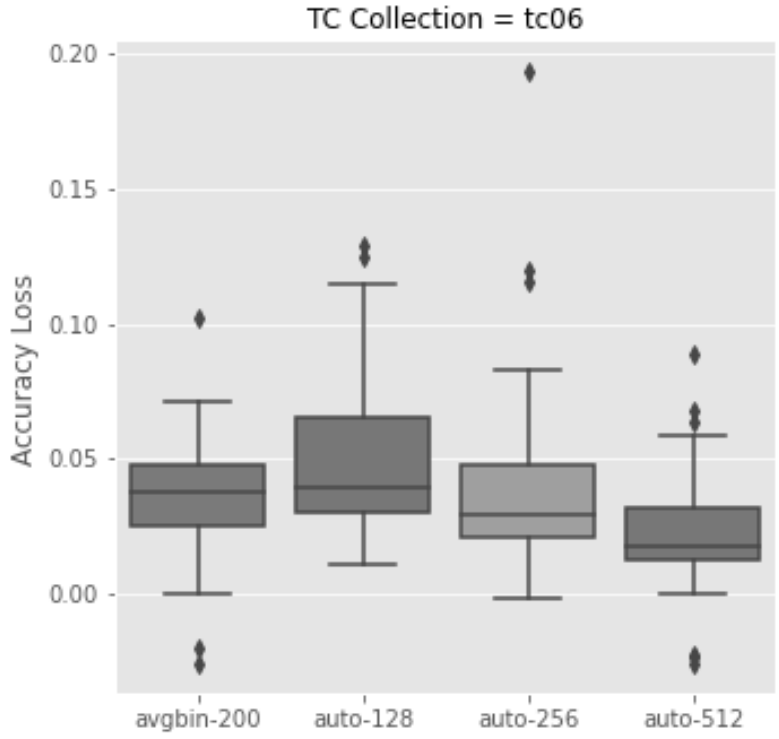


Figure 4.12: Relative accuracy loss for DLCC Test Collection tc06.

of size 5000 (Table B.24), similarly as in [37], the best models were the matrix-factorization based RESCAL and ComplEx, both with 0.99 accuracy, and all their binary variants scored between 0.96 and 0.98.

4.3.4 Qualified Restrictions

The class separation in terms of qualified restrictions corresponds to the DLCC test collections tc07 and tc08, where tc08 is the inverse case of tc07. For example, in tc07, the classifiers should be able to differentiate people who played on any basketball team from those who did not, while in tc08, they should separate people who starred in any television show from those who did not. The basic difference between them is the direction of the edges in intermediary entities (*someone played on some basketball team* vs. *some television show starred someone*). The relative accuracy loss for these test collections is presented side by side in Figure 4.13.

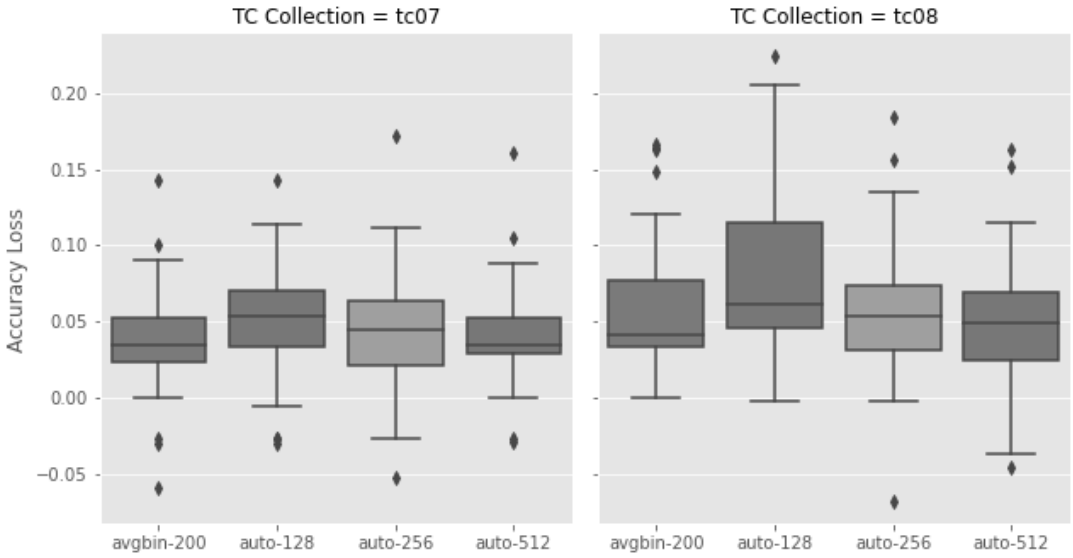


Figure 4.13: Relative accuracy loss for DLCC Test Collections tc07 and tc08.

It can be seen that accuracy is better preserved in the tc07 test collection than in tc08, probably because tc07 consists of a more direct task that can be solved accurately by most KGEs. In these tests, AVGBIN-200 and AUTO-512 were considered the best binarization approaches. In both test collections with size 5000 (Tables B.25 and B.26), the TransE-L2 embeddings are the top among the original vectors, with 0.986 in tc07 and 0.967 in tc08, and also among the binary variants, with 0.970 (AVGBIN-200) in tc07 and 0.945 (AUTO-512) in tc08.

4.3.5 Cardinality Restrictions of Relations

The class separation in terms of cardinality restrictions of relations corresponds to the DLCC test collections tc09 and tc10, where tc10 is the inverse case of tc09. For example, in tc09, the classifiers should be able to differentiate movies that have at least two directors from those that do not, while in tc10, they should separate movies for which at least two entities are known from movies that do not own this restriction. Similarly to tc07 and tc08, the basic difference between them is the direction of the edges (*some movie has two or more directors* vs. *two or more entities are known for some movie*). Although tc10 may seem more complex, it can generally be better learned by KGEs than tc09 [34]. The relative accuracy loss for these test collections is presented side by side in Figure 4.14.

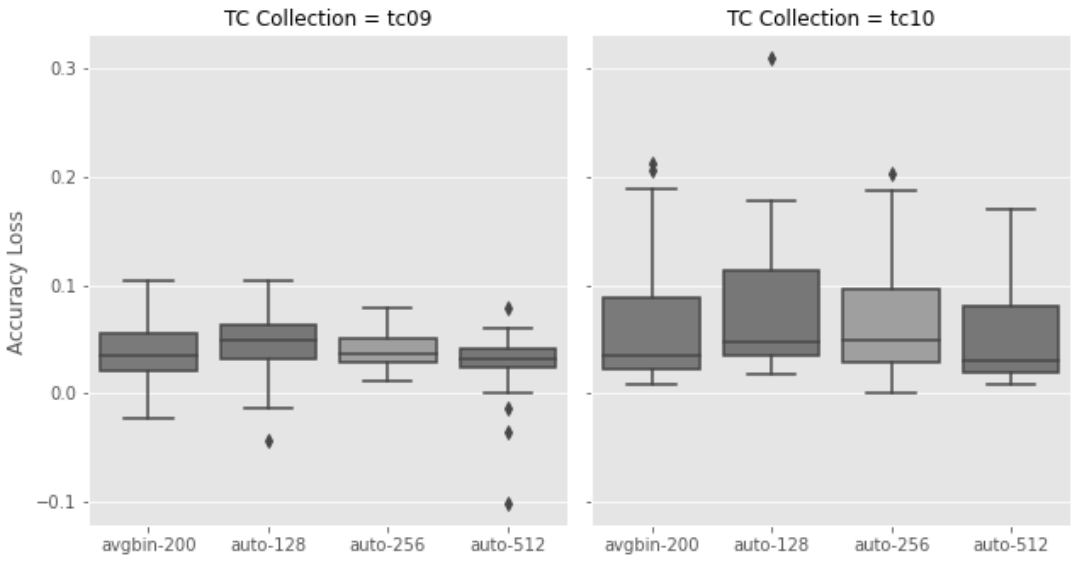


Figure 4.14: Relative accuracy loss for DLCC Test Collections tc09 and tc10.

It can be seen that accuracy is better preserved in the tc09 test collection, with losses often below 6%, than in tc10, where all distributions are spread with the third quartile closer to the mark of 10%. The different binarization methods performed similarly for each test collection. Consistent with the results in Section 4.3.4, it can be observed that knowledge about these types of restrictions is better preserved by binary variants for direct/outgoing edges than for inverse/ingoing edges.

4.3.6 Qualified Cardinality Restrictions

Last but not least, the class separation in terms of qualified cardinality restrictions corresponds to the DLCC test collections tc11 and tc12, where tc12 is the inverse case of tc11. For example, in tc11, the classifiers should be able to differentiate movies starring at least two people from those that do not, while in tc12, they should separate movies for which at least two people are known from movies that do not own this restriction. The basic difference from tc11 and tc12 to tc09 and tc10 is the qualification of the cardinality (in examples involving movies, entities should be instances of people). The relative accuracy loss for these test collections is presented side by side in Figure 4.15.

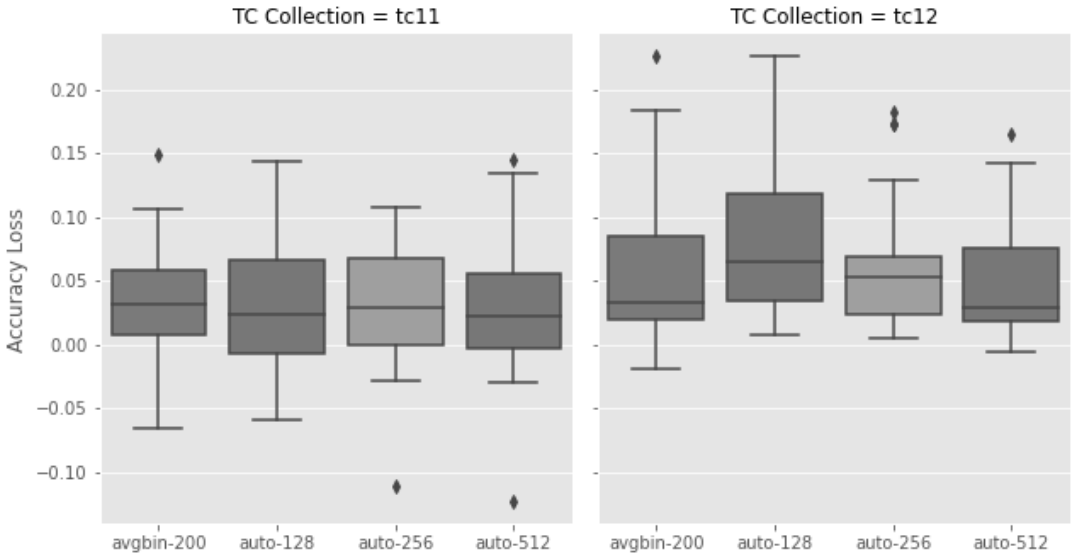


Figure 4.15: Relative accuracy loss for DLCC Test Collections tc11 and tc12.

Again, binary variants preserved accuracy better in the direct test collection (tc11), with losses ranging more often between 0 and 5%, than in the inverse test collection (tc12). For tc11, in particular, all distributions are similar and there were multiple cases in which the accuracy loss was insignificant, sometimes even higher scores than their original versions (see Table B.29). A positive example is RDF2vec$_{SG}$, whose binary variants AVGBIN-200 and AUTO-256 achieved the best scores of the test collection (0.974 and 0.975, respectively), surpassing its original 0.954.

Chapter 5

Conclusion

5.1 Summary

In this research, knowledge graph embeddings were converted to binary vectors that are considerably more compact, with file sizes that are 10 to 50 times smaller. The semantic preservation of these binarized KGEs was experimentally assessed using the evaluation frameworks GEval [29] and DLCC [33]. The experiments covered a variety of tasks including classification, regression, clustering, document similarity, entity-relatedness, and semantic analogies to evaluate the same eleven knowledge graph embeddings of three different natures used in [34]. These KGEs were converted to binary vectors of different sizes using the autoencoder architecture proposed in [45] and evaluated under the same procedure. A naive binarization approach using averages of the original vectors is also used as a baseline for a full comparison.

For GEval, in the classification task, the embeddings autoencoded in 512 bits generally performed well, with only a modest loss in accuracy (3-6%). In particular, certain binary variants even outperformed their original counterparts, showcasing the potential of binary embeddings to preserve semantic information. However, the performance varied between different models and datasets. Similar results were observed in clustering tasks, but with greater variability, probably due to the nature of unsupervised learning tasks, where convergence may not always occur. In contrast, regression tasks revealed that embeddings autoencoded in 128 bits often achieved better performance than their respective continuous versions, indicating that dimensionality reduction can be beneficial in these scenarios. The sensitivity of regression to high-dimensional input was evident, especially in smaller datasets, where lower-dimensional binary embeddings exhibited better performance and less

overfitting.

The document similarity task and the entity-relatedness task showed that autoencoded binary embeddings generally achieved higher correlations in their respective datasets, even when compared to the original vectors. This suggests that the autoencoding technique seems to preserve well the semantic information required for these tasks, but further investigation with more datasets should be conducted. However, semantic analogy tasks indicated that while some binary variants could preserve semantic relationships in very particular cases, there was a noticeable loss of precision, suggesting that much of the translational properties from continuous vectors were lost during binarization.

In the DLCC framework, focusing on classification tasks, the binary vectors autoencoded in 512 bits consistently provided expressive semantic preservation, with insignificant accuracy losses in half of the datasets, surpassing other binarization methods. Furthermore, specific class separation tasks, such as relations to particular individuals, qualified cardinality restrictions, and particular relations to particular individuals, showed reasonable semantic preservation, often ranging from 0 to 5% of loss of accuracy. Other class-separation tasks, such as qualified restrictions and cardinality restrictions, suggested that the loss of information in binary embeddings is higher for inverse relations (tc08 and tc10) than for direct ones (tc07 and tc09). Particularly for tc11, which corresponds to qualified cardinality restrictions in outgoing edges, binary embeddings often scored even better than their original versions.

In general, models that typically figure as the top performers in these evaluation frameworks, such as RDF2vec$_{SG}$ and TransE with L2 regularization [34], also had their performance well-preserved, especially when binarized using 512-bit autoencoding. However, the increase in the number of dimensions is only beneficial if the downstream task in mind is or can be optimized for binary operations. Otherwise, these operations might take even longer with binary vectors than originally with continuous vectors of fewer dimensions.

5.2 Remarks

The experimental results presented in this study highlight the trade-offs and nuances associated with binarizing knowledge graph embeddings. Although certain tasks and embeddings demonstrated promising performance with binary variants, there were instances of significant accuracy loss, especially in semantic analogy

tasks. The impact of binarization varied between different embeddings and tasks, emphasizing the need for careful consideration depending on the requirements of the final application and the kind of information relevant to solve it. For example, the sensitivity of regression tasks to dimensionality, the convergence in unsupervised learning tasks, and the translational properties assumed for tasks such as semantic analogies, all contribute to the complexity of evaluating and adopting binary embeddings. In this direction, an important remark is that changes can be made in the autoencoding architecture to better preserve information about the distances between entities in the continuous vector space, as proposed in [24] and [28].

In terms of reproducibility of the results obtained in this research, a critical remark is that the binarizer open-sourced by [45] does not yet allow random seeds, so when a vector file with continuous embeddings is binarized with autoencoding multiple times, the output tends to be slightly different, even when the hyperparameters are kept constant. As soon as newer implementations with the possibility of setting a random seed become available, future research in the direction of binarization with autoencoders should adopt this functionality.

Another important remark is that different applications may tolerate different levels of semantic preservation. In cases where high accuracy is absolutely necessary, losses of 3 to 6% may not be acceptable, but expressive savings in the memory footprint may decently compensate for this loss in many other applications. In this study, all statistical tests to detect significant preservation in performance metrics confronted the null hypothesis of *no loss*, but if a certain application tolerates 1 or 2% loss, these tests would produce completely different results.

It should also be mentioned that the datasets in GEval and DLCC relate to specific snapshots of DBpedia and the real world. For some datasets, such as the Cities and Countries in the clustering task, significant changes in the target variable over time are unlikely, but in the classification and regression dataset Cities, for example, the target variable (quality of living in each city) may get obsolete over decades. Investigating whether there was an impact of this time sensitivity of the datasets covered by GEval and DLCC is beyond the scope of this research.

5.3 Future Work

The findings of this study open avenues for future research in several directions. Relative to the remarks mentioned in Section 5.2, different autoencoding architectures that better preserve pairwise distances in the continuous vector space could

be tested against the architecture used in this study, as promising candidates for semantic analogies. Implementations of Tissier's binarizer in other programming languages and of these newer approaches would also be beneficial, especially if providing a mechanism for reproducibility, such as random seeds. Additionally, the DLCC evaluation framework allows the generation of synthetic datasets, which could be used together with DBpedia gold standards for a deeper understanding of what information is preserved in each case.

In another direction, given the positive result of binary variants in experiments, current KGE models could be adapted to optionally train binary embeddings by design. This applies directly to RDF2vec models, in the training step where Word2vec is used on generated sequences to learn numerical representations. Changes in its neural network could be made towards a binary output, which could eventually outperform the binary vectors post-processed with autoencoding. Similarly, to make the best use of binary vectors, distance-based machine learning models and vector search engines could also be optimized for them, with binary operations such as XOR, which are performed with a single CPU cycle due to hardware optimizations in modern processors. In cases where the pairwise Hamming distance between entities in the binary vector space strongly correlates with the pairwise cosine distance between the same entities in the continuous vector space, the gains in efficiency and scalability would be considerable.

Furthermore, another advantage of binary vectors that has yet to be mentioned is the interpretability they may be able to provide. For example, it is possible that a certain bit alone or just a few bits together encode relevant information about the entities and are more decisive in class separation tasks. If that happens, a decision tree could be used to visualize this pattern, and important conclusions about what KGEs learn to represent about their entities could be driven.

By addressing these areas, future research can contribute to refining and extending the applicability of binary embeddings in the context of knowledge graph representation learning.

Bibliography

[1] Faisal Alshargi, Saeedeh Shekarpour, Tommaso Soru, and Amit Sheth. Concept2vec: Metrics for Evaluating Quality of Embeddings for Ontological Concepts, May 2020. arXiv:1803.04488 [cs].

[2] Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web: A New Form of Web Content That is Meaningful to Computers Will Unleash a Revolution of New Possibilities. *ScientificAmerican.com*, May 2001.

[3] Peter Bloem, Xander Wilcke, Lucas Berkel, and Victor Boer. kgbench: A Collection of Knowledge Graph Datasets for Evaluating Relational and Multimodal Machine Learning. pages 614–630. May 2021.

[4] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250, Vancouver Canada, June 2008. ACM.

[5] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating Embeddings for Modeling Multi-relational Data. 2013, December 2013.

[6] Hongyun Cai, Vincent W. Zheng, and Kevin Chen-Chuan Chang. A Comprehensive Survey of Graph Embedding: Problems, Techniques and Applications, February 2018. arXiv:1709.07604 [cs].

[7] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam Hruschka, and Tom Mitchell. Toward an Architecture for Never-Ending Language Learning. volume 3, January 2010.

[8] Moses S. Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388, Montreal Quebec Canada, May 2002. ACM.

[9] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2D Knowledge Graph Embeddings, July 2018. arXiv:1707.01476 [cs].

[10] David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, Nico Schlaefer, and Christopher Welty. Building Watson: An Overview of the DeepQA Project. *AI Magazine*, 31:59–79, September 2010.

[11] Aditya Grover and Jure Leskovec. node2vec: Scalable Feature Learning for Networks, July 2016. arXiv:1607.00653 [cs, stat].

[12] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, José Emilio Labra Gayo, Sabrina Kirrane, Sebastian Neumaier, Axel Polleres, Roberto Navigli, Axel-Cyrille Ngonga Ngomo, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan Sequeda, Steffen Staab, and Antoine Zimmermann. Knowledge Graphs. *ACM Computing Surveys*, 54(4):1–37, May 2022. arXiv:2003.02320 [cs].

[13] Arun Krishnan. Making search easier, August 2018. Section: Innovation at Amazon.

[14] Denis Krompass, Maximilian Nickel, and Volker Tresp. Querying Factorized Probabilistic Triple Databases. October 2014.

[15] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, and Christian Bizer. DBpedia – A large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web*, 6(2):167–195, 2015.

[16] Douglas Lenat. CYC: A Large-Scale Investment in Knowledge Infrastructure. *Communications of the ACM*, 38, December 1998.

[17] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning Entity and Relation Embeddings for Knowledge Graph Completion. *Proceedings of AAAI*, 29:2181–2187, February 2015.

[18] Wang Ling, Chris Dyer, Alan Black, and Isabel Trancoso. Two/Too Simple Adaptations of Word2Vec for Syntax Problems. May 2015.

[19] Jue Liu, Zhuocheng Lu, and Wei Du. Combining Enterprise Knowledge Graph and News Sentiment Analysis for Stock Price Prediction. 2019.

[20] F. Mahdisoltani, J. Biega, and Fabian M. Suchanek. YAGO3: A Knowledge Base from Multilingual Wikipedias. 2015.

[21] Leland McInnes, John Healy, and James Melville. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction, September 2020. arXiv:1802.03426 [cs, stat].

[22] André Meló and Heiko Paulheim. Synthesizing Knowledge Graphs for Link and Type Prediction Benchmarking. pages 136–151, May 2017.

[23] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space, September 2013. arXiv:1301.3781 [cs].

[24] Samarth Navali, Praneet Sherki, Ramesh Inturi, and Vanraj Vala. Word Embedding Binarization with Semantic Information Preservation. In Donia Scott, Nuria Bel, and Chengqing Zong, editors, *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1256–1265, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics.

[25] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A Review of Relational Machine Learning for Knowledge Graphs. *Proceedings of the IEEE*, 104(1):11–33, January 2016. arXiv:1503.00759 [cs, stat].

[26] Maximilian Nickel, Volker Tresp, and Peer Kröger. A Three-Way Model for Collective Learning on Multi-Relational Data. pages 809–816, January 2011.

[27] Natasha Noy, Yuqing Gao, Anshu Jain, Anant Narayanan, Alan Patterson, and Jamie Taylor. Industry-scale Knowledge Graphs: Lessons and Challenges: Five diverse technology companies show how it's done. *Queue*, 17:48–75, April 2019.

[28] Feiyang Pan, Shuokai Li, Xiang Ao, and Qing He. Relation Reconstructive Binarization of word embeddings. *Frontiers of Computer Science*, 16(2):162307, September 2021.

[29] Maria Pellegrino, Abdulrahman Altabba, Martina Garofalo, Petar Ristoski, and Michael Cochez. GEval: A Modular and Extensible Evaluation Framework for Graph Embedding Techniques. pages 565–582. May 2020.

[30] Maria Pellegrino, Michael Cochez, Martina Garofalo, and Petar Ristoski. A Configurable Evaluation Framework for Node Embedding Techniques. pages 156–160. October 2019.

[31] Jan Portisch, Michael Hladik, and Heiko Paulheim. KGvec2go – Knowledge Graph Embeddings as a Service, March 2020. arXiv:2003.05809 [cs].

[32] Jan Portisch and Heiko Paulheim. Putting RDF2vec in Order, August 2021. arXiv:2108.05280 [cs].

[33] Jan Portisch and Heiko Paulheim. The DLCC Node Classification Benchmark for Analyzing Knowledge Graph Embeddings, July 2022. arXiv:2207.06014 [cs].

[34] Jan Portisch and Heiko Paulheim. The RDF2vec Family of Knowledge Graph Embedding Methods | www.semantic-web-journal.net, August 2023.

[35] Vikas Raunak. Simple and Effective Dimensionality Reduction for Word Embeddings, November 2017. arXiv:1708.03629 [cs].

[36] Daniel Ringler and Heiko Paulheim. One Knowledge Graph to Rule Them All? Analyzing the Differences Between DBpedia, YAGO, Wikidata & co. volume 10505, pages 366–372, Cham, 2017. Springer International Publishing. Book Title: KI 2017: Advances in Artificial Intelligence Series Title: Lecture Notes in Computer Science.

[37] Petar Ristoski, Jessica Rosati, Tommaso Di Noia, Renato De Leone, and Heiko Paulheim. RDF2Vec: RDF graph embeddings and their applications. *Semantic Web*, 10(4):721–752, May 2019.

[38] Petar Ristoski, Gerben Klaas Dirk Vries, and Heiko Paulheim. A Collection of Benchmark Datasets for Systematic Evaluations of Machine Learning on the Semantic Web. pages 186–194, October 2016.

[39] Baoxu Shi and Tim Weninger. Open-World Knowledge Graph Completion, November 2017. arXiv:1711.03438 [cs].

[40] Saurabh Shrivastava. Bring rich knowledge of people, places, things and local businesses to your apps, July 2017.

[41] Amit Singhal. Introducing the Knowledge Graph: things, not strings, May 2012.

[42] Dagobert Soergel. WordNet. An Electronic Lexical Database. October 1998.

[43] Suchanek, Marián Fabian, Kasneci, Gjergji, Weikum, and Gerhard. Yago: a core of semantic knowledge. January 2007.

[44] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space, February 2019. arXiv:1902.10197 [cs, stat].

[45] Julien Tissier, Christophe Gravier, and Amaury Habrard. Near-lossless Binarization of Word Embeddings. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):7104–7111, July 2019. arXiv:1803.09065 [cs].

[46] Kristina Toutanova and Danqi Chen. Observed Versus Latent Features for Knowledge Base and Text Inference. July 2015.

[47] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex Embeddings for Simple Link Prediction, June 2016. arXiv:1606.06357 [cs, stat].

[48] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, November 2008.

[49] Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, September 2014.

[50] Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. Knowledge base completion via search-based question answering. pages 515–526, April 2014.

[51] Jiaming Xu, Peng Wang, Guanhua Tian, Bo Xu, Jun Zhao, Fangyuan Wang, and Hongwei Hao. Convolutional neural networks for text hashing. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 1369–1375, Buenos Aires, Argentina, July 2015. AAAI Press.

[52] Mengjia Xu. Understanding graph embedding methods and their applications, December 2020. arXiv:2012.08019 [cs, math].

[53] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding Entities and Relations for Learning and Inference in Knowledge Bases, December 2014.

[54] Xiaohan Zou. A Survey on Application of Knowledge Graph. *Journal of Physics: Conference Series*, 1487(1):012016, March 2020. Publisher: IOP Publishing.

Appendix A

Program Code / Resources

The source code used to obtain all 55 vector files, run the experiments, and analyze the results, as well as additional test results, is available on GitHub[1]. The documentation of the repository instructs how to reproduce the experiments described in the implementation chapter, and Jupyter notebooks used to analyze the GEval and DLCC results are saved with the input and output of each cell for better transparency.

Due to large file sizes, the TXT vector files used in the experiments are not directly available in the repository, but can be produced with the existing code. For the sake of reproducibility, the 33 compact VEC files filtered with GEval and DLCC entities are stored in the resource folder of the GitHub repository and can be easily converted to the TXT vector files of the autoencoded variants. These can also be obtained by running the pipeline, but knowing that the binarizer step is non-deterministic. The remaining variants ORIGINAL-200 and AVGBIN-200 can be obtained by downloading the original vector files from KGvec2go and following the documented steps.

[1] GitHub repository: https://github.com/vitor-faria/kgembeddings-binarization

Appendix B

Further Experimental Results

In this Appendix, results specific to GEval datasets and DLCC test collections are shown. For GEval, the granular results for the four of six tasks that cover multiple datasets are presented in Section B.1, ordered by task. For DLCC, tables of accuracy scores for the 12 test collections of size 5000 are presented in ascending order in Section B.2.

B.1 GEval Results per Dataset

B.1.1 Classification Results

MetacriticMovies Dataset

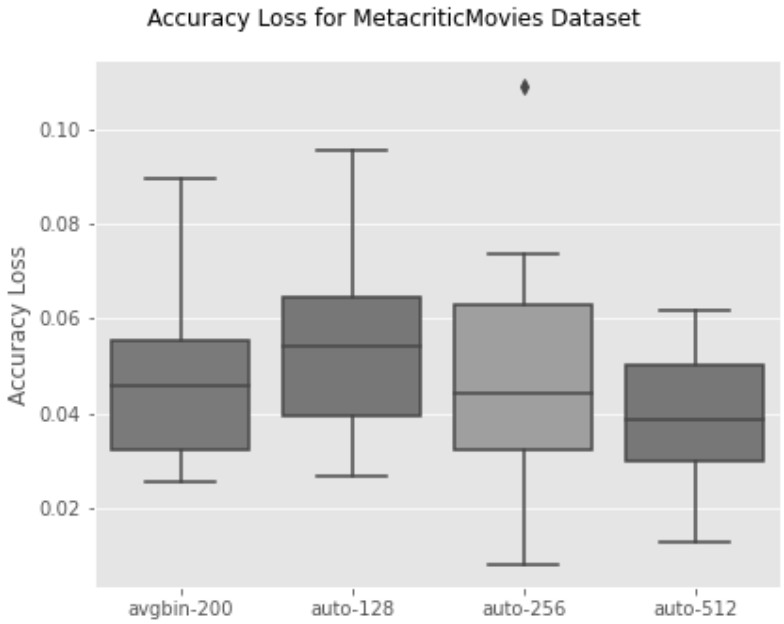


Figure B.1: Relative accuracy loss in the Metacritic Movies dataset.

	original-200	avgbin-200	auto-128	auto-256	auto-512
RDF2vec$_{CBOW}$	**0.538**	0.523	0.523	0.533	0.507
RDF2vec$_{CBOW-OA}$	**0.620**	0.604	0.580	0.581	0.597
RDF2vec$_{SG}$	**0.714**	0.688	0.686	0.695	0.705
RDF2vec$_{SG-OA}$	**0.712**	0.677	0.684	0.690	0.680
RESCAL	**0.683**	0.664	0.663	0.658	0.662
DistMult	**0.673**	0.613	0.637	0.644	0.654
ComplEx	**0.695**	0.646	0.658	0.658	0.666
TransE-L1	**0.640**	0.610	0.579	0.570	0.602
TransE-L2	**0.754**	0.712	0.690	0.706	0.724
TransR	**0.710**	0.671	0.681	0.686	0.693
RotatE	**0.564**	0.540	0.528	0.523	0.529

Table B.1: Accuracy scores for best classifiers in dataset Metacritic Movies.

Metacritic Albums Dataset

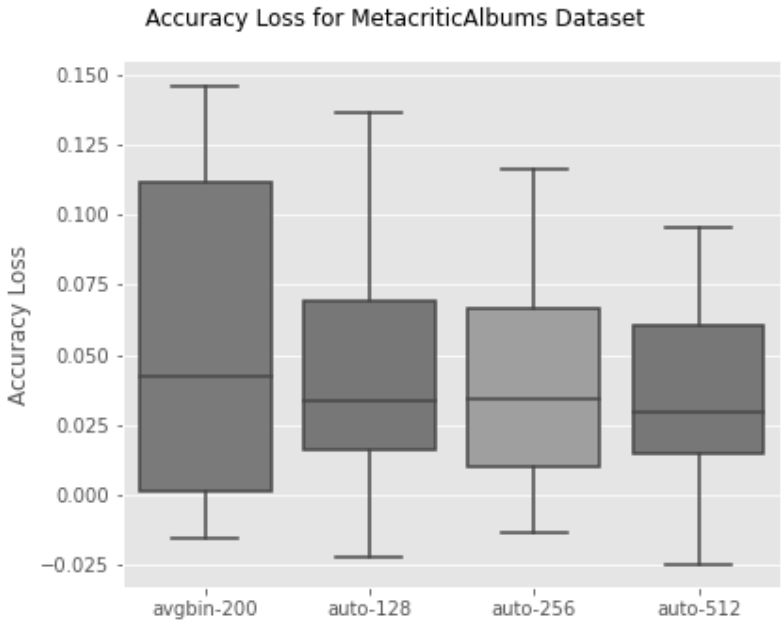


Figure B.2: Relative accuracy loss in the Metacritic Albums dataset.

	original-200	avgbin-200	auto-128	auto-256	auto-512
RDF2vec$_{CBOW}$	0.530	0.539	0.529	0.538	**0.543**
RDF2vec$_{CBOW-OA}$	**0.519**	**0.519**	0.506	0.511	0.507
RDF2vec$_{SG}$	0.582	0.584	**0.595**	0.581	0.583
RDF2vec$_{SG-OA}$	**0.581**	0.580	0.576	0.579	0.578
RESCAL	**0.620**	0.561	0.572	0.567	0.577
DistMult	**0.630**	0.538	0.568	0.578	0.597
ComplEx	**0.628**	0.548	0.614	0.597	0.599
TransE-L1	**0.622**	0.596	0.585	0.603	0.606
TransE-L2	**0.660**	0.635	0.638	0.637	0.640
TransR	**0.616**	0.537	0.532	0.545	0.558
RotatE	**0.569**	0.525	0.534	0.540	0.526

Table B.2: Accuracy scores for best classifiers in dataset Metacritic Albums.

Forbes Dataset

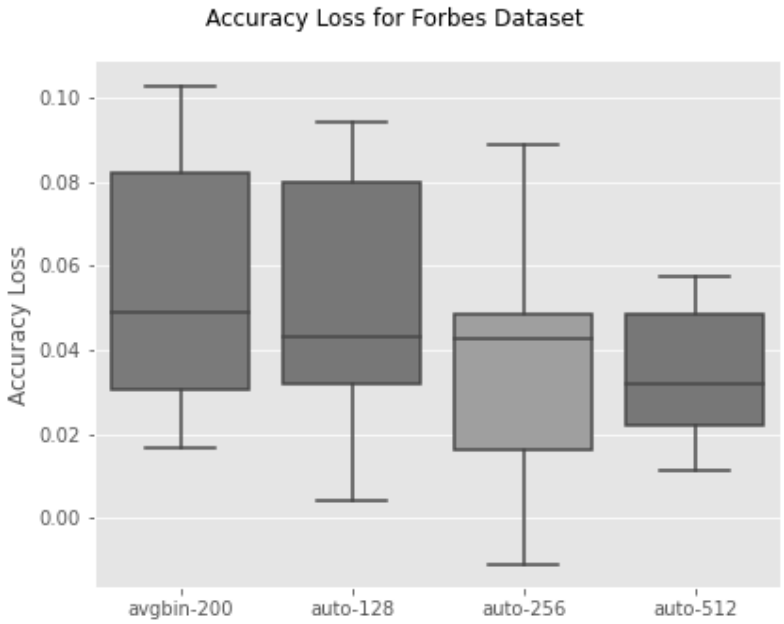


Figure B.3: Relative accuracy loss in the Forbes dataset.

	original-200	avgbin-200	auto-128	auto-256	auto-512
RDF2vec$_{CBOW}$	**0.565**	0.551	0.546	0.560	0.559
RDF2vec$_{CBOW-OA}$	**0.591**	0.555	0.565	0.565	0.572
RDF2vec$_{SG}$	**0.612**	0.596	0.609	0.607	0.605
RDF2vec$_{SG-OA}$	0.595	0.570	0.581	**0.602**	0.583
RESCAL	**0.594**	0.546	0.538	0.545	0.565
DistMult	**0.562**	0.509	0.514	0.548	0.532
ComplEx	**0.569**	0.541	0.525	0.545	0.536
TransE-L1	**0.565**	0.545	0.543	0.552	0.540
TransE-L2	**0.603**	0.593	0.585	0.574	0.588
TransR	**0.568**	0.509	0.536	0.517	0.550
RotatE	**0.531**	0.487	0.487	0.506	0.506

Table B.3: Accuracy scores for best classifiers in dataset Forbes.

Cities Dataset

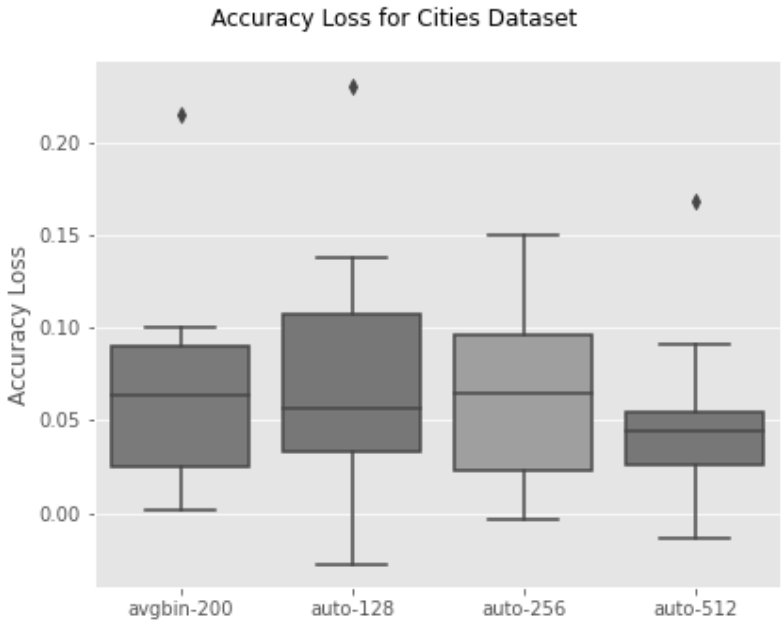


Figure B.4: Relative accuracy loss in the Cities dataset.

	original-200	avgbin-200	auto-128	auto-256	auto-512
RDF2vec$_{CBOW}$	**0.703**	0.686	0.630	0.657	0.639
RDF2vec$_{CBOW-OA}$	**0.716**	0.678	0.653	0.637	0.683
RDF2vec$_{SG}$	0.783	0.782	0.763	0.765	**0.793**
RDF2vec$_{SG-OA}$	**0.785**	0.717	0.754	0.769	0.763
RESCAL	0.745	0.732	**0.763**	0.747	0.751
DistMult	**0.672**	0.629	0.579	0.600	0.631
ComplEx	**0.735**	0.577	0.565	0.625	0.611
TransE-L1	**0.679**	0.611	0.640	0.622	0.660
TransE-L2	**0.806**	0.751	0.764	0.765	0.767
TransR	0.748	0.728	**0.769**	0.744	0.715
RotatE	**0.632**	0.574	0.562	0.578	0.616

Table B.4: Accuracy scores for best classifiers in dataset Cities.

AAUP Dataset

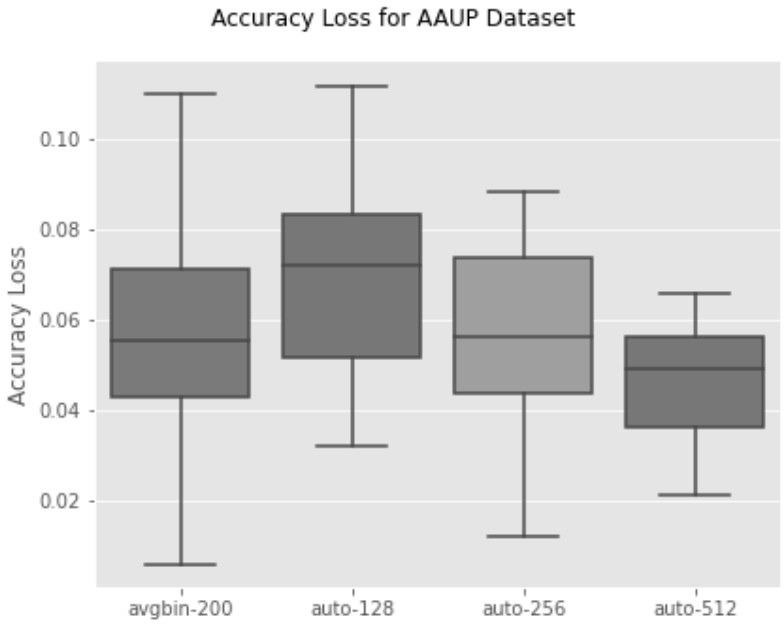


Figure B.5: Relative accuracy loss in the AAUP dataset.

	original-200	avgbin-200	auto-128	auto-256	auto-512
RDF2vec$_{CBOW}$	**0.630**	0.597	0.560	0.574	0.588
RDF2vec$_{CBOW-OA}$	**0.688**	0.638	0.630	0.632	0.648
RDF2vec$_{SG}$	**0.696**	0.670	0.663	0.661	0.677
RDF2vec$_{SG-OA}$	**0.705**	0.666	0.656	0.679	0.681
RESCAL	**0.643**	0.639	0.607	0.635	0.619
DistMult	**0.628**	0.559	0.572	0.584	0.594
ComplEx	**0.614**	0.571	0.563	0.568	0.582
TransE-L1	**0.630**	0.609	0.602	0.594	0.616
TransE-L2	**0.658**	0.626	0.637	0.636	0.626
TransR	**0.629**	0.582	0.584	0.595	0.605
RotatE	**0.621**	0.580	0.573	0.576	0.581

Table B.5: Accuracy scores for best classifiers in dataset AAUP.

B.1.2 Regression Results

AAUP Dataset

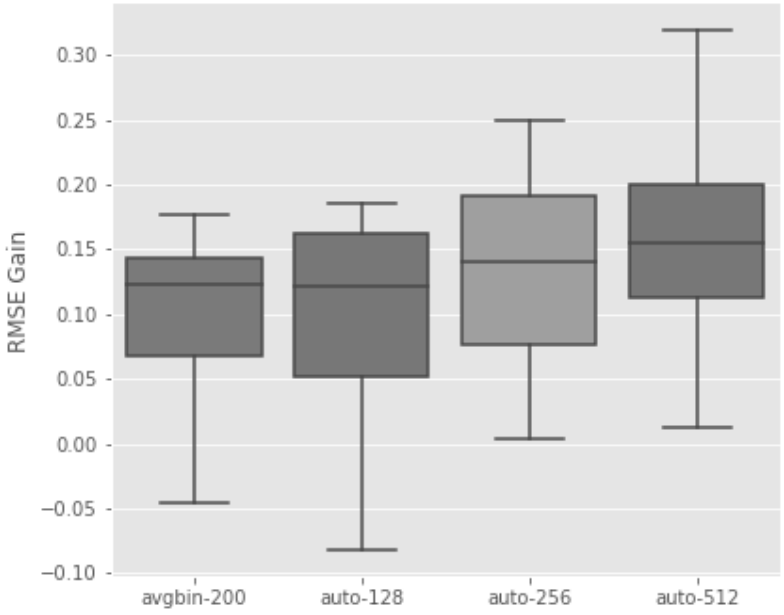


Figure B.6: Relative RMSE gain in the AAUP dataset.

	original-200	avgbin-200	auto-128	auto-256	auto-512
RDF2vec$_{CBOW}$	**78.37**	87.96	91.04	88.88	87.14
RDF2vec$_{CBOW-OA}$	**67.45**	79.40	77.82	82.69	81.86
RDF2vec$_{SG}$	**65.07**	72.99	76.65	74.17	72.79
RDF2vec$_{SG-OA}$	**65.15**	73.55	77.27	79.98	75.51
RESCAL	**69.36**	78.41	75.62	79.36	75.93
DistMult	**73.82**	86.43	82.81	92.26	88.89
ComplEx	**76.76**	88.78	85.91	88.85	91.74
TransE-L1	83.19	81.55	**79.93**	85.12	109.80
TransE-L2	**65.41**	72.99	76.10	71.90	72.76
TransR	87.29	83.36	**80.08**	87.56	88.36
RotatE	**84.23**	85.80	85.32	88.74	97.24

Table B.6: RMSE scores for best regressors in dataset AAUP.

Cities Dataset

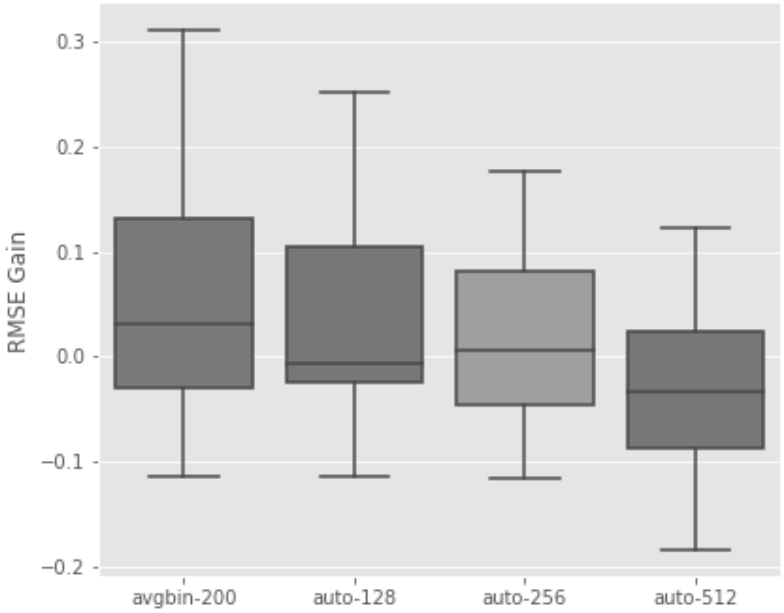


Figure B.7: Relative RMSE gain in the Cities dataset.

	original-200	avgbin-200	auto-128	auto-256	auto-512
RDF2vec$_{CBOW}$	19.64	19.15	18.12	**17.38**	19.49
RDF2vec$_{CBOW-OA}$	19.96	20.58	**19.20**	19.66	19.39
RDF2vec$_{SG}$	15.64	13.88	13.86	13.87	**12.76**
RDF2vec$_{SG-OA}$	13.38	15.31	13.77	14.42	**12.25**
RESCAL	16.72	**14.81**	16.56	16.17	14.98
DistMult	**18.16**	22.24	20.76	19.37	19.15
ComplEx	**15.85**	20.80	19.85	18.66	17.80
TransE-L1	**17.12**	19.17	20.03	18.61	18.08
TransE-L2	12.75	12.46	13.60	12.84	**11.63**
TransR	13.60	14.47	13.51	15.37	**13.14**
RotatE	21.50	20.76	21.25	**20.24**	20.46

Table B.7: RMSE scores for best regressors in dataset Cities.

Forbes Dataset

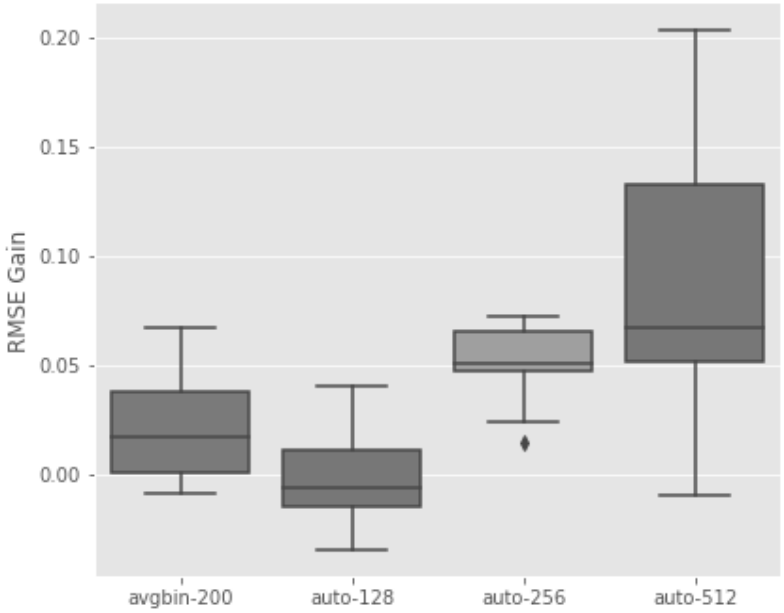


Figure B.8: Relative RMSE gain in the Forbes dataset.

	original-200	avgbin-200	auto-128	auto-256	auto-512
RDF2vec$_{CBOW}$	39.63	39.29	**38.55**	41.40	41.91
RDF2vec$_{CBOW-OA}$	37.81	37.85	**37.58**	39.90	40.76
RDF2vec$_{SG}$	**36.70**	37.97	37.27	39.07	38.34
RDF2vec$_{SG-OA}$	37.06	37.83	**36.61**	38.89	38.41
RESCAL	36.84	38.37	37.05	39.26	**36.47**
DistMult	**37.31**	39.81	38.29	39.89	39.81
ComplEx	**36.26**	37.91	37.71	38.88	39.53
TransE-L1	37.93	38.03	**37.27**	38.82	45.02
TransE-L2	36.95	36.95	**36.71**	38.84	44.48
TransR	38.93	38.96	**37.57**	39.49	45.78
RotatE	39.30	39.98	**38.83**	41.30	41.89

Table B.8: RMSE scores for best regressors in dataset Forbes.

MetacriticAlbums Dataset

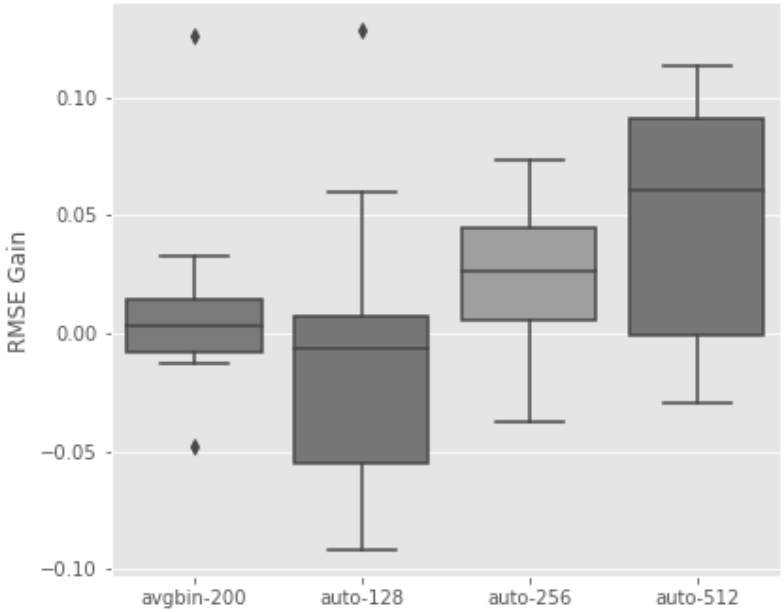


Figure B.9: Relative RMSE gain in the Metacritic Albums dataset.

	original-200	avgbin-200	auto-128	auto-256	auto-512
RDF2vec$_{CBOW}$	15.96	15.76	**14.59**	15.38	15.72
RDF2vec$_{CBOW-OA}$	15.99	15.22	**14.74**	15.96	16.00
RDF2vec$_{SG}$	15.63	15.69	**14.19**	15.84	15.60
RDF2vec$_{SG-OA}$	16.02	15.82	15.97	**15.42**	15.54
RESCAL	**14.75**	14.83	15.64	15.21	15.94
DistMult	14.29	16.09	**14.20**	14.94	15.70
ComplEx	**14.41**	14.75	16.26	15.05	16.04
TransE-L1	14.83	14.79	**14.41**	15.07	15.74
TransE-L2	**13.88**	14.34	14.12	14.91	15.32
TransR	14.66	14.70	**14.58**	15.31	15.42
RotatE	15.03	15.04	**14.56**	15.43	16.29

Table B.9: RMSE scores for best regressors in dataset Metacritic Albums.

MetacriticMovies Dataset

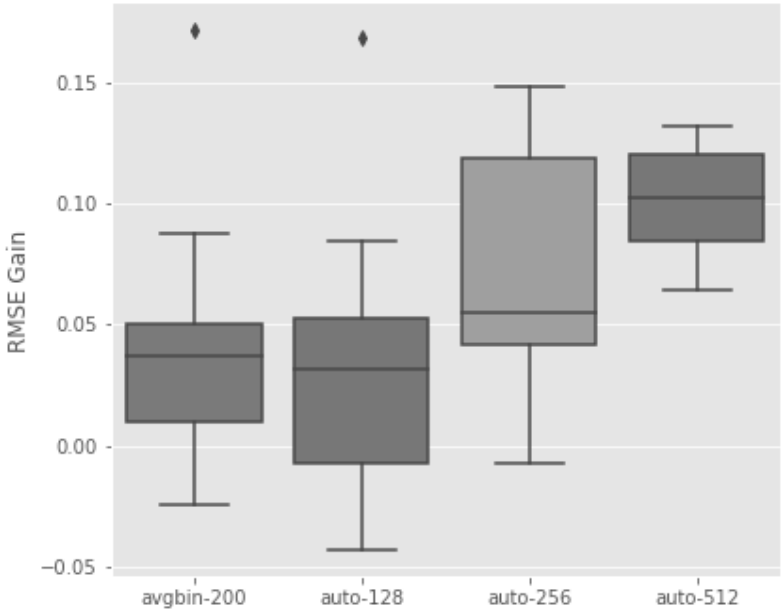


Figure B.10: Relative RMSE gain in the Metacritic Movies dataset.

	original-200	avgbin-200	auto-128	auto-256	auto-512
RDF2vec$_{CBOW}$	24.96	24.37	**23.88**	24.78	26.56
RDF2vec$_{CBOW-OA}$	23.57	23.26	**23.12**	23.95	25.55
RDF2vec$_{SG}$	**20.46**	21.22	21.01	21.80	23.12
RDF2vec$_{SG-OA}$	**20.57**	21.60	21.33	23.63	22.85
RESCAL	**21.67**	22.23	23.51	22.86	23.72
DistMult	**21.40**	25.08	22.09	24.08	24.22
ComplEx	**21.16**	23.02	21.86	23.98	23.33
TransE-L1	22.99	23.16	**22.92**	24.01	25.99
TransE-L2	**19.88**	20.89	21.24	22.11	21.57
TransR	**20.80**	21.66	24.31	21.85	23.01
RotatE	24.00	24.31	**23.74**	24.92	25.69

Table B.10: RMSE scores for best regressors in dataset Metacritic Movies.

B.1.3 Clustering Results

teams-cluster Dataset

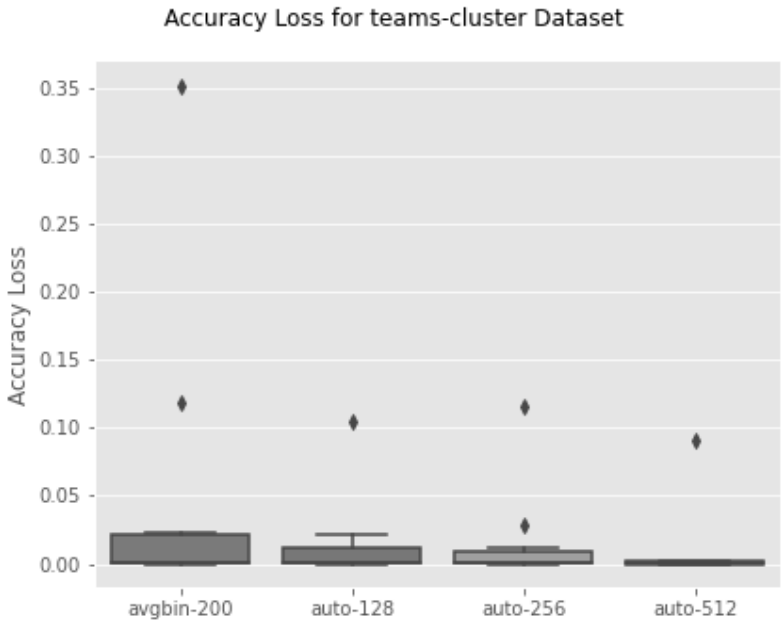


Figure B.11: Relative accuracy loss in the Teams dataset.

	original-200	avgbin-200	auto-128	auto-256	auto-512
RDF2vec$_{CBOW}$	**0.942**	0.610	0.843	0.916	0.856
RDF2vec$_{CBOW-OA}$	**0.941**	**0.941**	**0.941**	0.940	**0.941**
RDF2vec$_{SG}$	**0.852**	0.805	**0.852**	**0.852**	0.851
RDF2vec$_{SG-OA}$	**0.941**	**0.941**	**0.941**	**0.941**	**0.941**
RESCAL	**0.941**	0.921	0.920	0.930	0.940
DistMult	**0.942**	0.830	**0.942**	**0.942**	**0.942**
ComplEx	**0.942**	0.941	0.941	0.940	**0.942**
TransE-L1	**0.941**	**0.941**	0.940	**0.941**	0.940
TransE-L2	**0.942**	**0.942**	**0.942**	**0.942**	**0.942**
TransR	**0.942**	0.941	0.941	0.936	0.940
RotatE	**0.942**	0.920	0.922	0.833	0.940

Table B.11: Accuracy scores for best clusterers in dataset Teams.

Cities-Movies-Albums-Companies-Uni Dataset

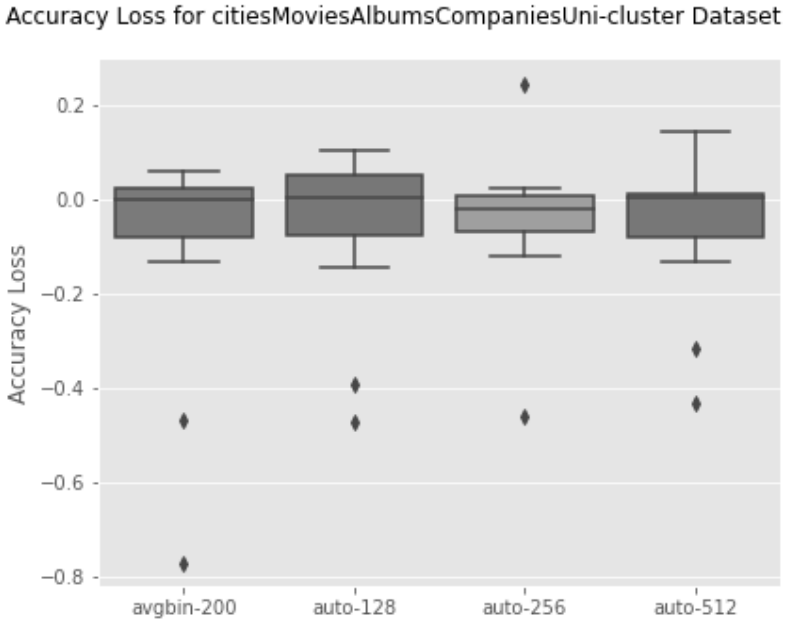


Figure B.12: Relative accuracy loss in Cities-Movies-Albums-Companies-Uni.

	original-200	avgbin-200	auto-128	auto-256	auto-512
RDF2vec$_{CBOW}$	0.433	**0.767**	0.637	0.632	0.570
RDF2vec$_{CBOW-OA}$	0.548	**0.805**	0.763	0.605	0.786
RDF2vec$_{SG}$	**0.749**	0.742	0.738	0.742	0.746
RDF2vec$_{SG-OA}$	0.854	0.819	0.765	**0.884**	0.828
RESCAL	**0.894**	0.871	0.874	0.872	0.883
DistMult	0.861	0.862	0.868	0.885	**0.886**
ComplEx	0.859	0.860	0.791	**0.864**	0.849
TransE-L1	**0.901**	0.885	0.895	0.885	0.890
TransE-L2	0.906	0.907	0.902	**0.914**	0.902
TransR	0.739	0.838	**0.846**	0.829	0.836
RotatE	**0.762**	0.718	0.688	0.575	0.652

Table B.12: Accuracy scores for best clusterers in dataset Cities-Movies-Albums-Companies-Uni.

Cities-Countries Dataset

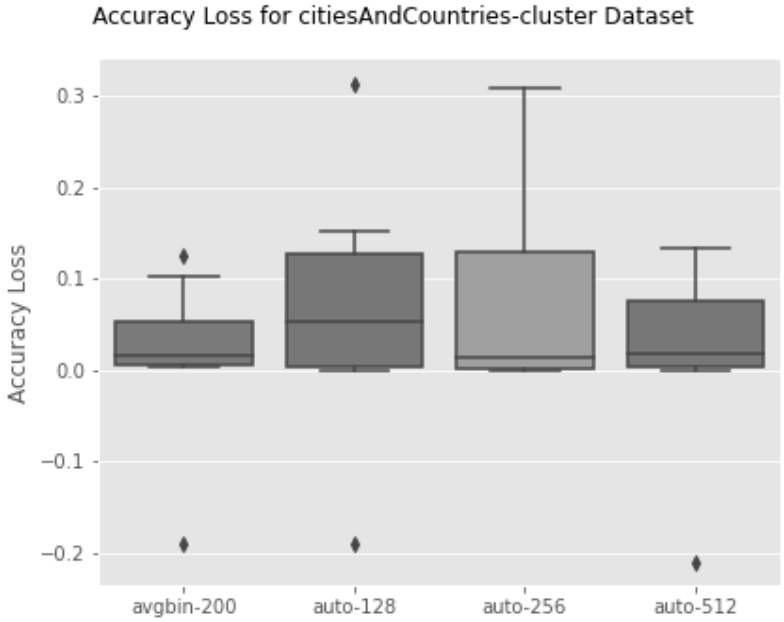


Figure B.13: Relative accuracy loss in the Cities-Countries dataset.

	original-200	avgbin-200	auto-128	auto-256	auto-512
RDF2vec$_{CBOW}$	**0.785**	0.705	0.539	0.543	0.741
RDF2vec$_{CBOW-OA}$	0.785	0.935	0.935	0.785	**0.950**
RDF2vec$_{SG}$	**0.742**	0.723	0.740	**0.742**	**0.742**
RDF2vec$_{SG-OA}$	**0.785**	0.773	**0.785**	**0.785**	**0.785**
RESCAL	**0.928**	0.867	0.787	0.787	0.841
DistMult	**0.896**	0.783	0.787	0.786	0.785
ComplEx	**0.909**	0.874	0.787	0.787	0.787
TransE-L1	**0.930**	0.925	0.872	0.922	0.910
TransE-L2	**0.939**	0.936	0.921	0.925	0.929
TransR	**0.917**	0.907	0.868	0.853	0.910
RotatE	**0.787**	0.782	0.780	0.785	0.773

Table B.13: Accuracy scores for best clusterers in dataset Cities-Countries.

Cities-Countries 2k Dataset

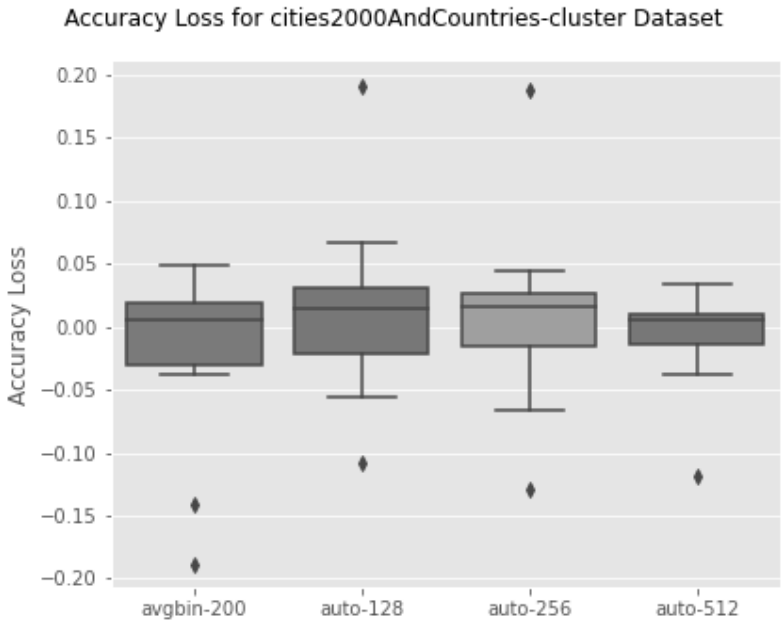


Figure B.14: Relative accuracy loss in the Cities-Countries 2k dataset.

	original-200	avgbin-200	auto-128	auto-256	auto-512
RDF2vec$_{CBOW}$	0.521	**0.619**	0.550	0.556	0.518
RDF2vec$_{CBOW-OA}$	0.894	0.928	**0.939**	0.927	0.928
RDF2vec$_{SG}$	0.581	**0.662**	0.581	0.581	0.581
RDF2vec$_{SG-OA}$	0.900	**0.922**	0.728	0.731	0.916
RESCAL	**0.933**	0.928	0.925	0.924	0.925
DistMult	0.868	0.841	0.855	0.854	**0.878**
ComplEx	**0.897**	0.861	0.837	0.857	0.866
TransE-L1	**0.932**	0.927	0.916	0.928	0.931
TransE-L2	**0.940**	0.894	0.906	0.918	0.931
TransR	**0.921**	0.916	0.908	0.894	0.896
RotatE	0.825	**0.831**	0.803	0.808	0.817

Table B.14: Accuracy scores for best clusterers in dataset Cities-Countries 2k.

B.1.4 Semantic Analogies Results

Country-Currency Dataset

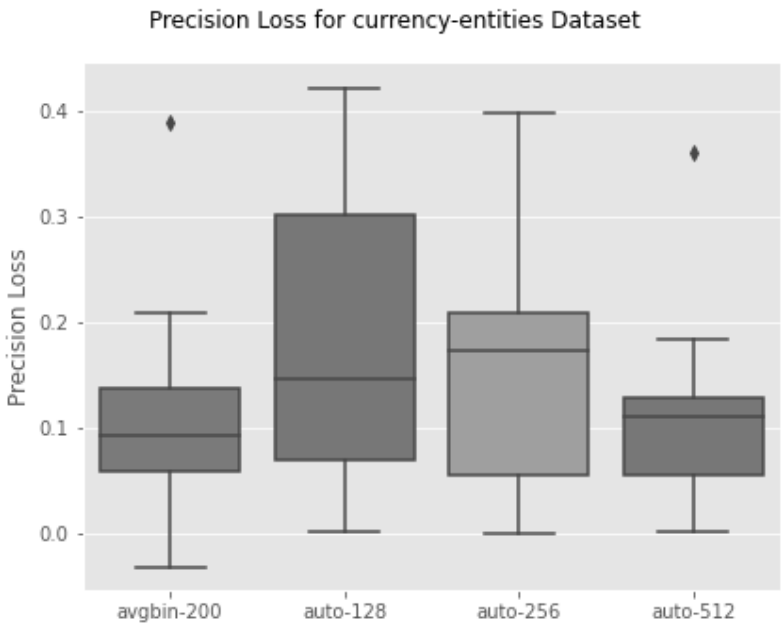


Figure B.15: Relative precision loss in the Country-Currency dataset.

	original-200	avgbin-200	auto-128	auto-256	auto-512
RDF2vec$_{CBOW}$	**0.488**	0.338	0.215	0.272	0.304
RDF2vec$_{CBOW-OA}$	**0.582**	0.374	0.185	0.184	0.222
RDF2vec$_{SG-OA}$	**0.637**	0.544	0.491	0.435	0.553
RESCAL	0.079	**0.111**	0.043	0.042	0.031
DistMult	**0.045**	0.005	0.004	0.009	0.019
ComplEx	**0.101**	0.017	0.006	0.030	0.040
TransE-L1	**0.504**	0.114	0.082	0.134	0.363
TransE-L2	**0.743**	0.642	0.411	0.597	0.626
TransR	**0.308**	0.231	0.118	0.136	0.198
RotatE	0.001	**0.002**	0.000	0.001	0.000

Table B.15: Precision @ 10 scores in dataset Country-Currency.

City-State Dataset

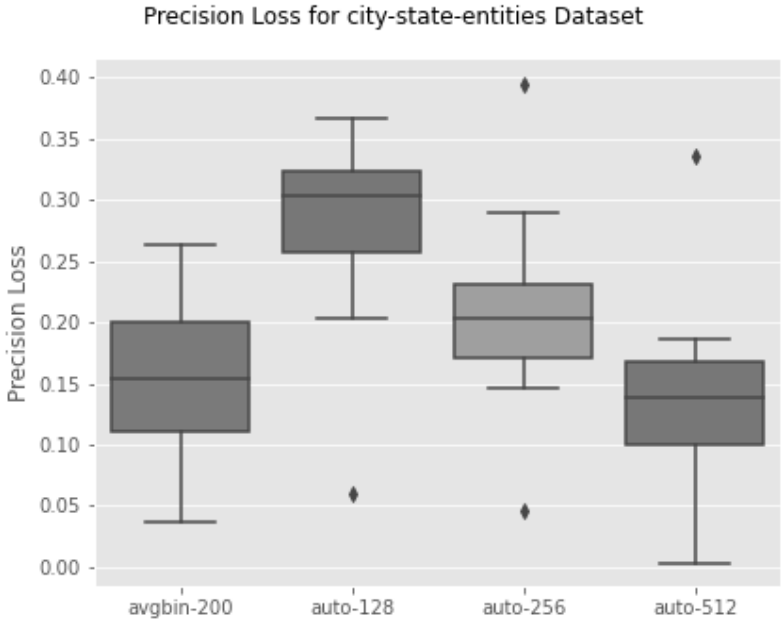


Figure B.16: Relative precision loss in the City-State dataset.

	original-200	avgbin-200	auto-128	auto-256	auto-512
RDF2vec$_{CBOW}$	**0.596**	0.485	0.255	0.381	0.409
RDF2vec$_{CBOW-OA}$	**0.568**	0.369	0.263	0.173	0.233
RDF2vec$_{SG-OA}$	**0.738**	0.603	0.474	0.449	0.587
RESCAL	**0.314**	0.277	0.254	0.269	0.310
DistMult	**0.541**	0.278	0.234	0.327	0.403
ComplEx	**0.578**	0.315	0.228	0.376	0.430
TransE-L1	**0.610**	0.432	0.306	0.454	0.483
TransE-L2	**0.590**	0.480	0.223	0.387	0.477
TransR	**0.620**	0.419	0.333	0.434	0.569
RotatE	**0.411**	0.303	0.160	0.265	0.324

Table B.16: Precision @ 10 scores in dataset City-State.

Capital-Country Dataset

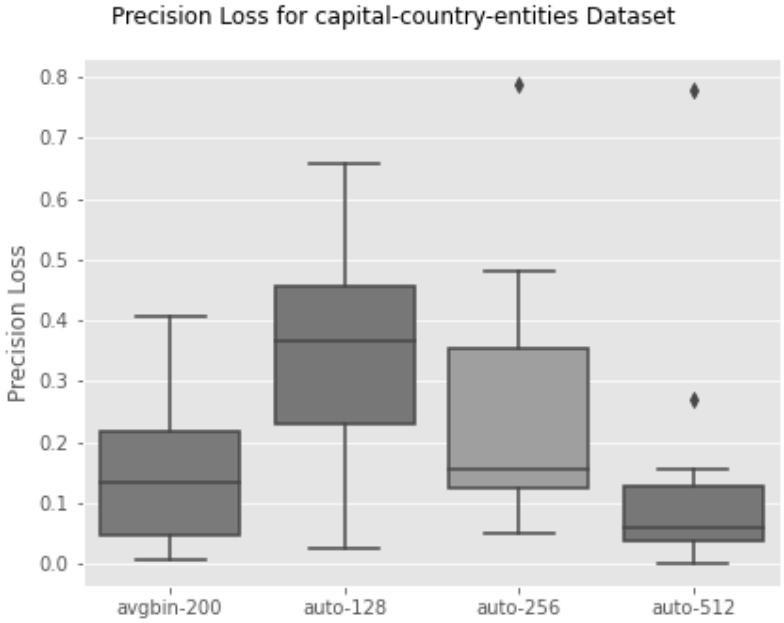


Figure B.17: Relative precision loss in the Capital-Country dataset.

	original-200	avgbin-200	auto-128	auto-256	auto-512
RDF2vec$_{CBOW}$	**0.909**	0.690	0.253	0.427	0.640
RDF2vec$_{CBOW-OA}$	**0.907**	0.500	0.403	0.119	0.128
RDF2vec$_{SG-OA}$	**0.986**	0.854	0.883	0.593	0.830
RESCAL	**0.872**	0.864	0.601	0.761	0.870
DistMult	**0.988**	0.775	0.617	0.848	0.953
ComplEx	**0.994**	0.725	0.561	0.840	0.923
TransE-L1	**0.994**	0.943	0.769	0.830	0.935
TransE-L2	**1.000**	0.957	0.634	0.951	0.962
TransR	**1.000**	0.915	0.763	0.919	0.966
RotatE	**0.885**	0.690	0.403	0.573	0.785

Table B.17: Precision @ 10 scores in dataset Capital-Country.

All-Capital-Country Dataset

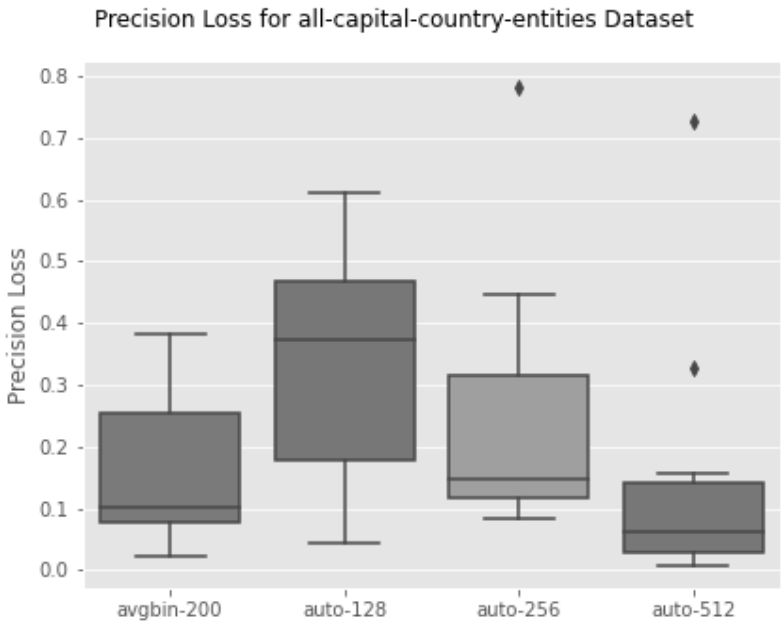


Figure B.18: Relative precision loss in the All-Capital-Country dataset.

	original-200	avgbin-200	auto-128	auto-256	auto-512
RDF2vec$_{CBOW}$	**0.735**	0.407	0.126	0.288	0.409
RDF2vec$_{CBOW-OA}$	**0.852**	0.471	0.242	0.070	0.126
RDF2vec$_{SG-OA}$	**0.944**	0.844	0.836	0.670	0.816
RESCAL	**0.688**	0.588	0.494	0.561	0.670
DistMult	**0.960**	0.740	0.586	0.851	0.931
ComplEx	**0.968**	0.782	0.620	0.871	0.937
TransE-L1	**0.961**	0.863	0.589	0.750	0.897
TransE-L2	**0.968**	0.919	0.575	0.821	0.913
TransR	**0.971**	0.914	0.810	0.887	0.963
RotatE	**0.808**	0.518	0.266	0.453	0.651

Table B.18: Precision @ 10 scores in dataset All-Capital-Country.

B.2 DLCC Results per Test Collection

B.2.1 Ingoing and Outgoing Relations

	original-200	avgbin-200	auto-128	auto-256	auto-512
RDF2vec$_{CBOW}$	**0.778**	0.757	0.741	0.751	0.759
RDF2vec$_{CBOW-OA}$	**0.873**	0.834	0.818	0.816	0.828
RDF2vec$_{SG}$	**0.918**	0.892	0.876	0.888	0.893
RDF2vec$_{SG-OA}$	**0.939**	0.899	0.868	0.879	0.902
RESCAL	**0.968**	0.913	0.906	0.920	0.929
DistMult	**0.875**	0.752	0.758	0.773	0.797
ComplEx	**0.860**	0.768	0.772	0.799	0.800
TransE-L1	**0.842**	0.794	0.778	0.788	0.797
TransE-L2	**0.947**	0.896	0.871	0.899	0.911
TransR	**0.862**	0.808	0.804	0.796	0.825
RotatE	**0.768**	0.674	0.670	0.680	0.678

Table B.19: Accuracy scores for best classifier of each embedding variant in each test collection tc01 of size 5000.

	original-200	avgbin-200	auto-128	auto-256	auto-512
RDF2vec$_{CBOW}$	**0.865**	0.776	0.796	0.854	0.778
RDF2vec$_{CBOW-OA}$	**0.956**	0.917	0.891	0.893	0.916
RDF2vec$_{SG}$	**0.953**	0.933	0.923	0.925	0.929
RDF2vec$_{SG-OA}$	**0.961**	0.936	0.801	0.927	0.936
RESCAL	**0.904**	0.858	0.856	0.864	0.852
DistMult	**0.855**	0.760	0.742	0.721	0.763
ComplEx	**0.846**	0.750	0.752	0.782	0.780
TransE-L1	**0.855**	0.816	0.791	0.801	0.813
TransE-L2	**0.972**	0.929	0.886	0.914	0.938
TransR	**0.833**	0.794	0.778	0.802	0.792
RotatE	**0.712**	0.704	0.694	0.698	0.688

Table B.20: Accuracy scores for best classifier of each embedding variant in each test collection tc02 of size 5000.

	original-200	avgbin-200	auto-128	auto-256	auto-512
RDF2vec$_{CBOW}$	**0.849**	0.828	0.791	0.809	0.810
RDF2vec$_{CBOW-OA}$	**0.902**	0.854	0.820	0.823	0.846
RDF2vec$_{SG}$	**0.950**	0.923	0.908	0.909	0.937
RDF2vec$_{SG-OA}$	**0.960**	0.928	0.909	0.899	0.931
RESCAL	**0.945**	0.907	0.902	0.913	0.925
DistMult	**0.895**	0.755	0.774	0.800	0.811
ComplEx	**0.873**	0.784	0.781	0.806	0.811
TransE-L1	**0.823**	0.786	0.774	0.772	0.777
TransE-L2	**0.933**	0.880	0.854	0.869	0.883
TransR	**0.854**	0.823	0.810	0.814	0.835
RotatE	**0.780**	0.705	0.700	0.692	0.704

Table B.21: Accuracy scores for best classifier of each embedding variant in each test collection tc03 of size 5000.

B.2.2 Relations to Particular Individuals

	original-200	avgbin-200	auto-128	auto-256	auto-512
RDF2vec$_{CBOW}$	**0.698**	0.668	0.643	0.653	0.661
RDF2vec$_{CBOW-OA}$	**0.872**	0.864	0.863	0.859	0.864
RDF2vec$_{SG}$	**0.960**	0.930	0.916	0.918	0.932
RDF2vec$_{SG-OA}$	**0.969**	0.945	0.937	0.935	0.948
RESCAL	**0.990**	0.956	0.955	0.967	0.965
DistMult	**0.984**	0.937	0.946	0.962	0.966
ComplEx	**0.989**	0.956	0.961	0.970	0.978
TransE-L1	**0.932**	0.903	0.872	0.878	0.896
TransE-L2	**0.987**	0.969	0.950	0.964	0.974
TransR	**0.973**	0.935	0.931	0.936	0.952
RotatE	**0.862**	0.827	0.795	0.813	0.817

Table B.22: Accuracy scores for best classifier of each embedding variant in each test collection tc04 of size 5000.

	original-200	avgbin-200	auto-128	auto-256	auto-512
RDF2vec$_{CBOW}$	**0.768**	0.767	0.696	0.658	0.699
RDF2vec$_{CBOW-OA}$	**0.904**	0.885	0.869	0.888	0.891
RDF2vec$_{SG}$	**0.987**	0.966	0.956	0.968	0.975
RDF2vec$_{SG-OA}$	**0.992**	0.979	0.934	0.975	0.982
RESCAL	**0.909**	0.878	0.878	0.880	0.883
DistMult	**0.904**	0.806	0.852	0.861	0.822
ComplEx	**0.908**	0.829	0.844	0.897	0.871
TransE-L1	**0.868**	0.834	0.817	0.820	0.798
TransE-L2	**0.947**	0.916	0.906	0.915	0.919
TransR	**0.881**	0.821	0.849	0.863	0.834
RotatE	**0.802**	0.729	0.733	0.706	0.705

Table B.23: Accuracy scores for best classifier of each embedding variant in each test collection tc05 of size 5000.

B.2.3 Particular Relations to Particular Individuals

	original-200	avgbin-200	auto-128	auto-256	auto-512
RDF2vec$_{CBOW}$	**0.693**	0.659	0.636	0.645	0.648
RDF2vec$_{CBOW-OA}$	**0.850**	0.841	0.835	0.825	0.835
RDF2vec$_{SG}$	**0.956**	0.932	0.918	0.919	0.931
RDF2vec$_{SG-OA}$	**0.963**	0.938	0.930	0.931	0.937
RESCAL	**0.990**	0.961	0.961	0.968	0.969
DistMult	**0.984**	0.938	0.950	0.958	0.968
ComplEx	**0.990**	0.962	0.960	0.971	0.978
TransE-L1	**0.928**	0.891	0.844	0.853	0.881
TransE-L2	**0.984**	0.965	0.945	0.961	0.968
TransR	**0.975**	0.938	0.932	0.948	0.955
RotatE	**0.866**	0.819	0.791	0.815	0.815

Table B.24: Accuracy scores for best classifier of each embedding variant in each test collection tc06 of size 5000.

B.2.4 Qualified Restrictions

	original-200	avgbin-200	auto-128	auto-256	auto-512
RDF2vec$_{CBOW}$	**0.746**	0.734	0.707	0.715	0.709
RDF2vec$_{CBOW-OA}$	**0.784**	0.756	0.739	0.729	0.746
RDF2vec$_{SG}$	**0.936**	0.900	0.896	0.879	0.898
RDF2vec$_{SG-OA}$	**0.953**	0.920	0.921	0.892	0.922
RESCAL	**0.944**	0.906	0.911	0.907	0.916
DistMult	**0.932**	0.847	0.842	0.887	0.900
ComplEx	**0.964**	0.879	0.873	0.918	0.931
TransE-L1	**0.935**	0.887	0.884	0.883	0.886
TransE-L2	**0.986**	0.970	0.950	0.965	0.969
TransR	**0.976**	0.936	0.920	0.933	0.948
RotatE	**0.845**	0.809	0.784	0.791	0.800

Table B.25: Accuracy scores for best classifier of each embedding variant in each test collection tc07 of size 5000.

	original-200	avgbin-200	auto-128	auto-256	auto-512
RDF2vec$_{CBOW}$	**0.889**	0.853	0.806	0.833	0.842
RDF2vec$_{CBOW-OA}$	**0.900**	0.863	0.848	0.839	0.851
RDF2vec$_{SG}$	**0.962**	0.934	0.920	0.919	0.938
RDF2vec$_{SG-OA}$	**0.966**	0.924	0.912	0.930	0.941
RESCAL	**0.880**	0.848	0.856	0.843	0.851
DistMult	**0.853**	0.726	0.732	0.770	0.768
ComplEx	**0.881**	0.799	0.776	0.793	0.821
TransE-L1	**0.898**	0.854	0.824	0.840	0.840
TransE-L2	**0.967**	0.930	0.913	0.936	0.945
TransR	**0.870**	0.839	0.829	0.840	0.862
RotatE	**0.831**	0.731	0.732	0.732	0.739

Table B.26: Accuracy scores for best classifier of each embedding variant in each test collection tc08 of size 5000.

B.2.5 Cardinality Restrictions of Relations

	original-200	avgbin-200	auto-128	auto-256	auto-512
RDF2vec$_{CBOW}$	**0.769**	0.743	0.723	0.737	0.734
RDF2vec$_{CBOW-OA}$	**0.853**	0.840	0.820	0.811	0.821
RDF2vec$_{SG}$	**0.898**	0.865	0.869	0.867	0.874
RDF2vec$_{SG-OA}$	**0.907**	0.883	0.875	0.881	0.886
RESCAL	**0.922**	0.881	0.877	0.889	0.889
DistMult	**0.880**	0.789	0.811	0.826	0.837
ComplEx	**0.887**	0.817	0.827	0.849	0.858
TransE-L1	**0.884**	0.850	0.828	0.854	0.861
TransE-L2	**0.935**	0.909	0.890	0.905	0.912
TransR	**0.876**	0.854	0.844	0.855	0.856
RotatE	**0.780**	0.735	0.732	0.746	0.744

Table B.27: Accuracy scores for best classifier of each embedding variant in each test collection tc09 of size 5000.

	original-200	avgbin-200	auto-128	auto-256	auto-512
RDF2vec$_{CBOW}$	**0.918**	0.875	0.843	0.858	0.874
RDF2vec$_{CBOW-OA}$	**0.909**	0.883	0.875	0.864	0.883
RDF2vec$_{SG}$	**0.950**	0.923	0.916	0.923	0.930
RDF2vec$_{SG-OA}$	**0.960**	0.943	0.913	0.921	0.941
RESCAL	**0.930**	0.908	0.906	0.908	0.907
DistMult	**0.918**	0.744	0.763	0.817	0.837
ComplEx	**0.935**	0.781	0.786	0.831	0.867
TransE-L1	**0.957**	0.930	0.929	0.929	0.939
TransE-L2	**0.985**	0.973	0.964	0.966	0.974
TransR	**0.894**	0.831	0.815	0.829	0.859
RotatE	**0.878**	0.791	0.748	0.773	0.786

Table B.28: Accuracy scores for best classifier of each embedding variant in each test collection tc10 of size 5000.

B.2.6 Qualified Cardinality Restrictions

	original-200	avgbin-200	auto-128	auto-256	auto-512
RDF2vec$_{CBOW}$	0.840	**0.868**	0.720	0.779	0.727
RDF2vec$_{CBOW-OA}$	0.773	0.823	0.797	0.858	**0.867**
RDF2vec$_{SG}$	0.954	0.974	0.891	**0.975**	0.915
RDF2vec$_{SG-OA}$	0.895	0.876	**0.935**	0.896	0.885
RESCAL	**0.954**	0.923	0.938	0.926	0.923
DistMult	**0.854**	0.772	0.787	0.771	0.805
ComplEx	**0.967**	0.911	0.903	0.890	0.839
TransE-L1	**0.918**	0.889	0.840	0.847	0.910
TransE-L2	**0.960**	0.946	0.954	0.953	0.952
TransR	0.925	0.925	0.896	0.890	**0.939**
RotatE	0.836	0.837	**0.858**	0.745	0.838

Table B.29: Accuracy scores for best classifier of each embedding variant in each test collection tc11 of size 5000.

	original-200	avgbin-200	auto-128	auto-256	auto-512
RDF2vec$_{CBOW}$	**0.885**	0.878	0.847	0.879	0.870
RDF2vec$_{CBOW-OA}$	**0.910**	0.880	0.876	0.873	0.884
RDF2vec$_{SG}$	**0.958**	0.927	0.907	0.907	0.932
RDF2vec$_{SG-OA}$	**0.938**	0.909	0.915	0.919	0.936
RESCAL	**0.931**	0.917	0.899	0.914	0.920
DistMult	**0.895**	0.770	0.746	0.838	0.817
ComplEx	**0.912**	0.775	0.798	0.820	0.814
TransE-L1	**0.961**	0.948	0.892	0.922	0.939
TransE-L2	**0.985**	0.973	0.951	0.969	0.971
TransR	**0.881**	0.826	0.816	0.827	0.865
RotatE	**0.834**	0.769	0.736	0.761	0.769

Table B.30: Accuracy scores for best classifier of each embedding variant in each test collection tc12 of size 5000.

Ehrenwörtliche Erklärung

Ich versichere, dass ich die beiliegende Masterarbeit ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen. Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

Mannheim, den 31.01.2024 Unterschrift