Karlsruhe Institute of Technology

# Leveraging Neural Language Models for Knowledge Graph Completion

Master Thesis

by

Vjola Cili

Degree Course: Industrial Engineering and Management M.Sc.

Matriculation Number: 1810833

Institute of Applied Informatics and Formal Description Methods (AIFB)

KIT Department of Economics and Management

| | |
|---|---|
| First Supervisor: | Prof. Dr. Harald Sack |
| Second Supervisor: | Prof. Dr. Michael Färber |
| Advisor: | Dr. Russa Biswas |
| Submitted: | 04.08.2023 |

# Abstract

Knowledge Graphs (KGs) provide a valuable framework for encoding data and its relationships, facilitating efficient structuring and understanding of large amounts of information. Despite their potential, KGs' inherent incompleteness undermines the quality of KG-based applications, necessitating the development of Knowledge Graph Completion (KGC) techniques. Predominantly, Knowledge Graph Embedding (KGE) methods are employed, which focus on the KG structure yet disregard the linguistic context. Recently, pretrained Neural Language Models (NLMs) have gained momentum due to their advanced language processing capabilities. By leveraging their ability to capture complex linguistic context, pretrained NLMs have the potential to boost KGC performance.

This thesis explores the applicability of pretrained NLMs in KGC by using two distinct approaches. The first approach involves infusing knowledge from pretrained NLMs, specifically Word2Vec and BERT, into the learning process of KGE models. This is achieved by employing NLM-generated embeddings of entities and relations as initialization vectors for a KGE model, followed by an evaluation on link prediction. The second approach, termed KG-NLM, fine-tunes three pretrained NLMs, namely RoBERTa, DistilBERT, and BLOOM, on KG triples for triple classification. Both approaches are evaluated on two benchmark datasets. According to the experimental results in line with the first approach, initializing KGE models with pretrained NLM-generated embeddings does not significantly enhance the models' performance. Conversely, the KG-NLM approach, notably KG-RoBERTa and KG-DistilBERT, show substantial performance across both datasets, surpassing the baseline model for one dataset. Ultimately, these findings support the use of pretrained NLMs for KGC tasks, as demonstrated by the promising outcomes of the KG-NLM approach.

# Contents

# List of Abbreviations

|  |  |
|---|---|
| **AI** | Artificial Intelligence. |
| **BERT** | Bidirectional Encoder Representations from Transformers. |
| **CBOW** | Continuous Bag-of-Words. |
| **CWA** | Closed World Assumption. |
| **IRI** | Internalised Resource Identifier. |
| **KB** | Knowledge Base. |
| **KGC** | Knowledge Graph Completion. |
| **KGEs** | Knowledge Graph Embeddings. |
| **KGs** | Knowledge Graphs. |
| **KR** | Knowledge Representation. |
| **KRL** | Knowledge Representation Learning. |
| **MLM** | Masked Language Modeling. |
| **MLP** | Multi-Layer Perceptron. |
| **MR** | Mean Rank. |
| **MRR** | Mean Reciprocal Rank. |
| **NLMs** | Neural Language Models. |
| **NLP** | Natural Language Processing. |
| **NSP** | Next Sentence Prediction. |
| **OWA** | Open World Assumption. |
| **PCA** | Principal Component Analysis. |
| **RDF** | Resource Description Framework. |
| **RDFS** | Resource Description Framework Schema. |

# List of Figures

# List of Tables

# 1 Introduction

Knowledge Graphs (KGs) have emerged as a powerful tool in the field of Artificial Intelligence (AI) providing a structured and semantically rich representation of data. By encoding data and its relationships in a network of nodes and edges, KGs offer an interpretable and flexible framework that facilitates the discovery of hidden patterns and inference in data. Specifically, a KG is comprised of entities, which refer to real-world objects or concepts, and relations, that represent the connections between entities [1]. Paired with machine learning techniques, KGs can be used to develop real-world applications such as recommender systems or intelligent question answering [2]. In addition to open-source KGs like Freebase [3], DBpedia [4] or Wikidata [5], numerous corporations including Amazon [6], Siemens [7] and Bosch [8, 9] have developed company-specific KGs. Utilizing these proprietary KGs, companies strive to transform their operations and workforces and improve customer interaction [10].

## 1.1 Motivation

Most KGs are automatically constructed from various data sources such as websites, databases or text corpora, which can only cover a fraction of all the possible information about a specific domain. Therefore, the resulting KG can only reflect the information available in these resources. Furthermore, due to the dynamic nature of knowledge, KGs can quickly become outdated or incomplete once new information arises. As a result, they often face the challenge of incompleteness. Incomplete KGs can impair the extraction of valuable insights or reduce the accuracy of search and recommender systems. In an effort to tackle this issue, various machine learning techniques are applied. These fall under the topic of Knowledge Graph Completion (KGC), which aims at inferring knowledge from an existing KG by predicting missing links [11]. Figure 1 visualizes the idea behind KGC.

Knowledge Graph Embeddings (KGEs) are commonly employed to perform KGC. These aim at representing KG entities and relations in continuous vector space, such that the geometric relations in space reflect the semantic relationships in the graph. This vector representation allows machine learning techniques to infer knowledge from the KG [11]. Numerous KGE models such as TransE [12], ConvE [13] or TuckER [14] have been proposed, each employing different embedding strategies. Nevertheless, most KGE approaches capture only the relationships between entities, missing out on the word context within these relationships.

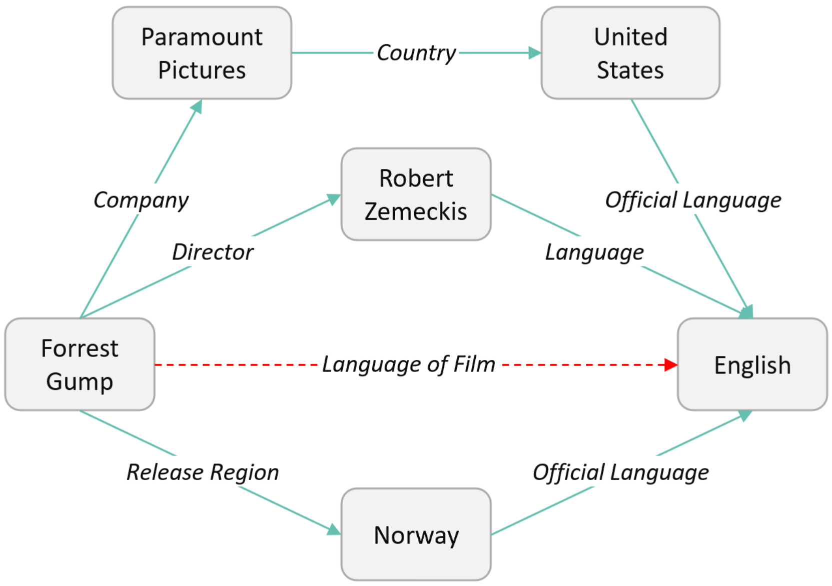


Figure 1: An example of KGC on a Freebase KG: Inferring a new relation (red) from existing ones (green), adapted from [15]

On the other hand, Neural Language Models (NLMs) have gained significant attention recently due to their superior performance in understanding, generating, and translating human language [16]. Their ability to capture complex linguistic contexts within large-scale data makes them essential for numerous applications, such as sentiment analysis, speech recognition and conversational AI. The pretrained NLMs such as BERT [17] or GPT-2 [18] are able to generate word representations that capture a word's various meanings depending on the context [19]. These pretrained NLMs can be used to generate rich, context-aware embeddings for entities and relationships. The embeddings can then be used as input for KGE models, providing a potentially superior starting point compared to random initialization. Furthermore, pretrained NLMs can be fine-tuned directly on a downstream KGC task by receiving as input KG triples, comprised of two entities linked by a relationship. This approach enables NLMs to learn the information contained in KGs' structure and assess the plausibility of a given triple. Although pretrained NLMs achieve state-of-the-art performance in many natural language processing tasks, their application in KGC remains relatively unexplored, presenting a significant opportunity for further research.

## 1.2 Objectives

Motivated by the great performance of pretrained NLMs in understanding and modelling human language and the need to handle KG incompleteness, this thesis explores the possibilities of employing pretrained NLMs to perform KGC. To accomplish this, the following research questions are to be answered:

- RQ1: How can the information encoded in pretrained NLMs be incorporated into the learning process of KGE models?
- RQ2: What is the impact of combining different pretrained NLMs and KGEs for KGC?
- RQ3: How do pretrained NLMs fine-tuned on a downstream KGC task perform?

To answer the research questions, this thesis considers two different approaches. The first approach exploits pretrained NLMs to generate entity and relation embeddings, which subsequently serve as initialization vectors for existing KGE models. The aim is to assess whether the information encapsulated in the pretrained embeddings enhances the performance of the KGE models on the task of link prediction. Alternatively, the second approach, named `KG-NLM`, evaluates the ability of pretrained NLMs to learn the information captured in a KG to perform KGC. Specifically, various pretrained NLMs are fine-tuned and evaluated on triple classification, as a subtask of KGC.

While the concept of initializing a KGE model with pretrained NLM-generated embeddings has been previously explored by Zhang et al. [20], this thesis builds upon this groundwork, extending the experimental analysis in several ways. This includes using both context-independent and context-aware pretrained NLMs, and the evaluation of two additional KGE models that have not been previously investigated in this context. Moreover, the second approach builds upon the work of Yao et al. [21], who introduced a BERT-based model for KGC. This thesis contributes to current research by fine-tuning and evaluating three pretrained NLMs, namely RoBERTa [22], DistilBERT [23] and BLOOM [24], on the task of triple classification across two datasets. As per the current understanding, such a combination of models, task and datasets has not been previously explored in literature.

## 1.3 Structure of the Thesis

The remainder of this thesis is organized as follows. Chapter 2 introduces the fundamental concepts, KGE models and pretrained NLMs needed for the understanding of this thesis. A review of relevant literature is provided in chapter 3. Furthermore, chapter 4 outlines the methodology employed in this thesis, including a description of the two approaches: `NLM-enhanced KGE` and `KG-NLM` for KGC. Subsequently, the used datasets, the conducted experiments and the respective results are discussed in chapter 5. To conclude, chapter 6 summarizes the principal findings and elaborates on limitations as well as opportunities for future research.

# 2 Theoretical Background

This chapter aims to create a shared understanding of the key concepts to this thesis. To facilitate this, a description of KGs accompanied by examples is provided in section 2.1. This section also includes an overview of KGC and several KGE methods employed in this thesis. Furthermore, section 2.2 explaines relevant NLMs. To conclude, a short description of the evaluation metrics used to assess the performance of both approaches is provided in section 2.3.

## 2.1 Knowledge Graphs

The concept of knowledge graphs can be traced back to the field of Knowledge Representation (KR), which emerged in the 1960s as a subfield of AI [2]. KR is concerned with finding a suitable way to represent knowledge so that machines are able to understand it and reason over it [25]. A central concept in KR is the Knowledge Base (KB), a collection of facts aiming to build a model of the world or of a slice of reality [26]. One early approach to KR are semantic networks, proposed by Richens in 1956 [27], which consist of nodes representing concepts and edges representing semantic relations between the concepts [28]. This graphical KR approach gained prominence with the introduction of the Semantic Web by Sir Tim Berners Lee [29]. The latter aimed at extending the World Wide Web with machine-readable descriptions of the web content enabling machines to process the information.

However, for machines to attain a true understanding of the added information, a standardized framework that conveys meaning is required [29]. This led to the development of the Resource Description Framework (RDF) as an abstract data model, used to represent and exchange data on the Web and the Resource Description Framework Schema (RDFS) providing meaning to the RDF data [29, 30]. RDF employs a triple structure, `<subject, predicate, object>`, also referred to as an RDF statement, to represent resources and the relationships that exist between them [31]. The `subject` denotes the resource the statement is about and is typically an Internalised Resource Identifier (IRI), but can also be a blank node representing an unnamed resource. An IRI consists of a sequence of characters that unambiguously identifies a resource [32]. The `predicate` is a property or a relationship attributed to the subject and is always an IRI. Lastly, the `object` refers to an entity the subject is related to and can be an IRI, a blank node or a literal. The latter is used for specific data types such as strings, numerical values and dates. As a result, knowledge is represented as a collection of RDF statements, forming an RDF Graph.

The term "Knowledge Graph" was initially introduced by Google in 2012 with the launch of their KB, the Google Knowledge Graph. Shortly after, the term gained great popularity in both academia and industry, becoming a widely accepted reference to any graph-based KB [33]. Ehrlinger et al. [34] provide a comprehensive analysis of the different KG definitions available in the literature. In the scope of this thesis, the following KG definition by Wang et al. is adopted [1]:

*A Knowledge Graph is a multi-relational graph, where entities (nodes) refer to real world objects or concepts and relations (edges) denote the relationship between entities.*

Therefore, the core structure of a KG can be formally described as a triple `<head, relation, tail>`, similar to the RDF statement, with `head` and `tail` denoting KG entities and `relation` denoting a KG relation. As a result, a KG provides structured information about a certain domain by connecting entities with each other. In addition to the entity and relation names, KGs often include textual entity and relation descriptions. Figure 2 provides an illustration of these descriptions, represented by the red framed boxes, accompanying the entity names. Throughout this work, the term "labels" is used for entity and relation names, while the corresponding textual descriptions are referred to as "descriptions."

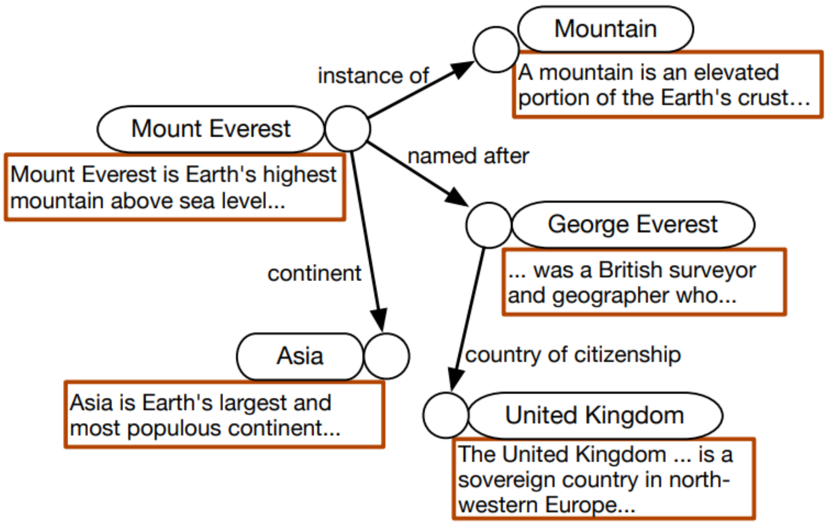

Mountain
A mountain is an elevated portion of the Earth's crust...
instance of
Mount Everest
Mount Everest is Earth's highest mountain above sea level...
named after
George Everest
... was a British surveyor and geographer who...
continent
Asia
Asia is Earth's largest and most populous continent...
country of citizenship
United Kingdom
The United Kingdom ... is a sovereign country in north-western Europe...

Figure 2: Example of a KG's entity names and descriptions, taken from [35]

In the following, two KBs, from which the KG datasets used in this thesis are extracted, are introduced.

**WordNet** [36] is a lexical KB of the English language that expresses semantic relations between words, as depicted in figure 3. It accomplishes this by organizing nouns, verbs, adverbs, and adjectives into groups called cognitive synonyms or synsets. These synsets are then interconnected through conceptual-semantic and lexical relationships. WordNet was established at Princeton University in the 1980s and has since been continuously developed and maintained [37].

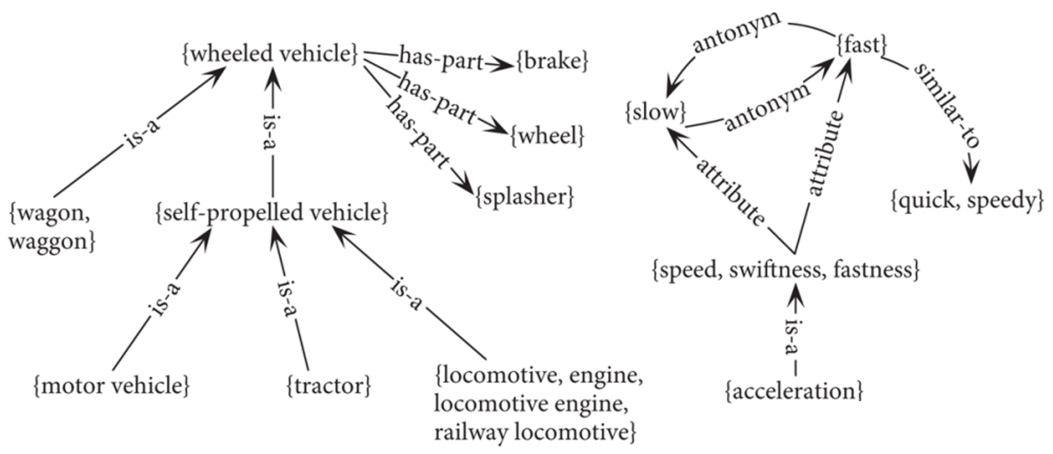


Figure 3: Visualization of a subgraph from WordNet, taken from [38]

**Freebase** [3] was an open, collaborative KB aimed at facilitating the development of web-based data-driven applications [39]. It included data from various sources such as Wikipedia, MusicBrainz and community contributions over the years. After being acquired by Google, the entire Freebase KB was eventually transferred to Wikidata before it was ultimately shut down in 2016 [40]. Figure 4 demonstrates a fragment of the Freebase KB.

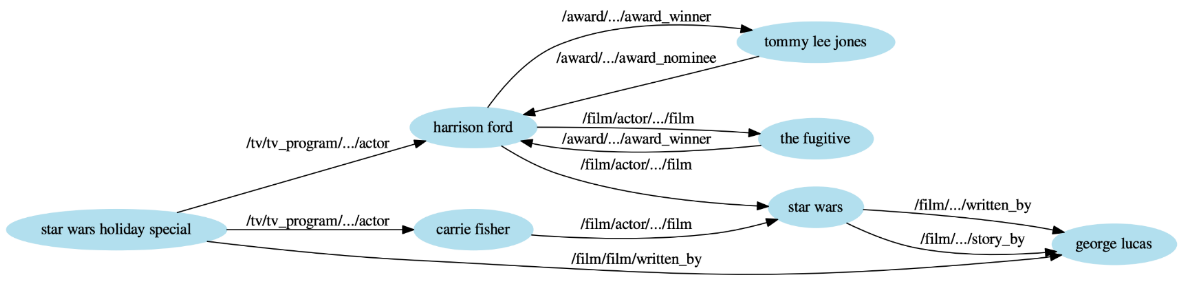


Figure 4: Visualization of a subgraph from Freebase, taken from [41]

### 2.1.1 Knowledge Graph Completion

Real-life KGs often suffer from incomplete knowledge due to missing information. Knowledge Graph Completion (KGC) addresses the incompleteness and sparsity of KGs by predicting missing links or discovering new facts, thereby augmenting the overall structure of the graph [11, 42]. In their comprehensive analysis of KGC methods, Chen et al. [11] distinguish between traditional KGC methods and Knowledge Representation Learning (KRL) methods. Traditional KGC methods infer knowledge by applying rule reasoning, probabilistic graph models or graph calculation. Alternatively, KRL based methods rely on machine learning techniques to transform a KG's entities and relations into low-dimensional vectors in a continuous vector space, also referred to as Knowledge Graph Embeddings (KGE) [43]. Being a central concept in this thesis, KGEs are discussed in detail in the following section. Furthermore, depending on the way of treating unknown facts, literature differentiates between the Closed World Assumption (CWA) and the Open World Assumption (OWA) [44, 11]. CWA considers only existing KG triples as true, thus treating non-existing triples directly as false. Hence, KGC relies solely on predicting relationships between the existing entities and relations. On the other hand, OWA considers non-existing KG triples as unknown, indicating that the absence of a triple in a KG does not necessarily imply its falsehood.

KGC tasks include link prediction and triple classification. Link prediction exploits existing facts in a KG to predict missing links. It does so by inferring the missing instance (head, relation, or tail) of a KG triple given the other two instances [45]. Triple classification refers to the task of determining if a given triple is part of a given KG. Hence, it is a binary classification problem, assessing whether a triple is true or not [2].

### 2.1.2 Knowledge Graph Embedding Methods

As previously explained, KGEs encode the entities and relations of a KG into low-dimensional vectors that can be processed by machine learning algorithms while capturing the semantic relationships and structure of the KG. For instance, if two entities often have similar relationships in the KG, their embeddings should be close together in the embedding space [12]. Typical KGE methods usually consist of the following three steps [1]:

1. Representing entities and relations in a continuous space. Entities are commonly represented as deterministic points in the vector space while relations are viewed as operations in the vector space.
2. Defining a scoring function that assesses the likelihood of a triple formation.
3. Learning entitiy and relation representations by solving an optimization problem that maximizes the total plausibility of the correct triples.

In their extensive analysis of KGE approaches, Ferrari et al. [45] differentiate between three primary embedding techniques: *translational models*, *semantic matching models* and *neural network models*.

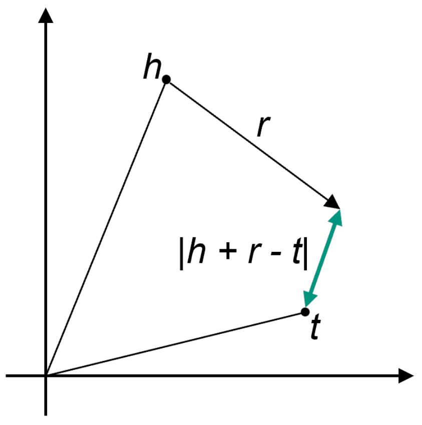
**Translational models** employ distance-based scoring functions to assess the plausibility of a triple. This is determined by the distance between two entities, typically after a translation performed by the relation is applied [1]. TransE [12] is the most prominent translational KGE model, whereby entities and relations are represented as vectors into the same embedding space. Given a KG triple *(h, r, t)*, the relation $r$ is modelled as a translation vector between the head $h$ and tail $t$ entities, such that the assumption $h + r \approx t$ holds. The scoring function is defined as follows, whereby $l_1/l_2$ denote the norm constraint:

$$f_r(h,t) = - \parallel h + r - t \parallel_{l1/l2} \tag{2.1}$$

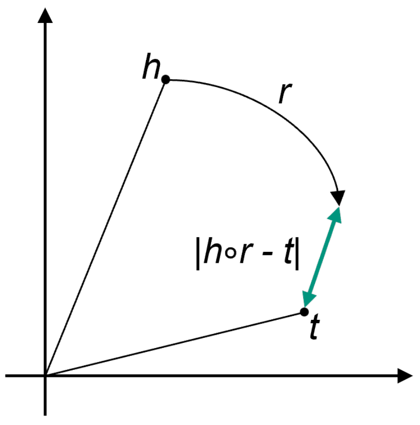
Sun et al. [46] suggest that the overarching goal of KGE methods is to model and deduce a KG's connectivity patterns based on the observed knowledge. Such connectivity patterns include symmetry/antisymmetry, inversion and composition. To achieve this, they introduce the RotatE model, which represents entities and relations in the complex vector space and models each relation $r$ as an element-wise rotation from the head entity $h$ to the tail entity $t$, as illustrated in figure 5 (b). Therefore, the scoring function can be defined as:

$$f_r(h,t) = - \parallel h \circ r - t \parallel \tag{2.2}$$

Overall, the idea behind both TransE and RotatE models depicted in figure 5 (a) and (b), is to minimize the distance $|h + r - t|$ or $|h \circ r - t|$ respectively for the observed KG triples (true triples) and maximize it for false triples.



(a)



(b)

Figure 5: Geometric visualization of TransE (a) from [45] and RotatE (b) from [46], for an embedding dimension equal to one.

**Semantic matching models** rely on a similarity-based scoring function to estimate the similarity between different entities and relations [45, 1]. The scoring function is calculated as a bilinear product, where the relation embedding is a bi-dimensional matrix $r$ [45]. It can be generally defined as follows:

$$f_r(h,t) = h \times r \times t \tag{2.3}$$

The intuition behind the matrix $r$ is that it captures how a head entity is related to the tail entity under the considered relation. RESCAL [47] is a bilinear model that follows the described logic to capture interactions between a KG's entities. The DistMult model [48] is introduced in an effort to reduce the complexity of RESCAL by restricting the relation matrix $r$ from a general asymmetric matrix to a diagonal square matrix [1]. Furthermore, the ComplEX model [49] extends DistMult by representing entities and relations in the complex space.

Unlike the previous semantic matching models, which use a bilinear product, the TuckER model [14] relies on the Tucker decomposition [50], a popular method in machine learning used for dealing with multidimensional data [51]. Specifically, TuckER considers each triple as a 3-way tensor that is factorized into a core tensor multiplied by a matrix along each mode. Entity and relation embeddings are represented in the rows of the matrices, while the core tensor captures the level of interaction between entities and relations [14]. The scoring function can be defined as follows:

$$f_r(h,t) = W \times_1 h \times_2 r \times_3 t \tag{2.4}$$

where $W$ denotes the core tensor, $h$ and $t$ refer to the rows of the entity embedding matrix, $r$ refers to the rows of the relation embedding matrix and $x_i$ indicates the tensor product along the i-th mode.

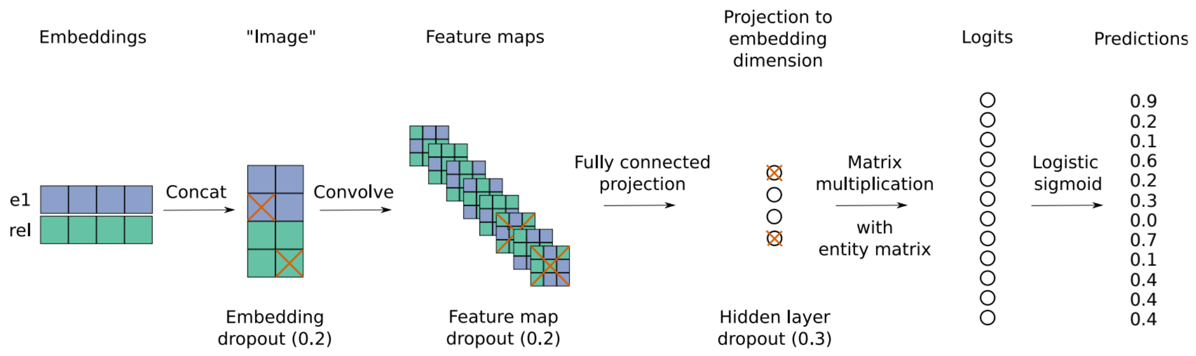

Embeddings
"Image"
Feature maps
Projection to embedding dimension
Logits
Predictions
e1
rel
Concat
Convolve
Fully connected projection
Matrix multiplication with entity matrix
Logistic sigmoid
Embedding dropout (0.2)
Feature map dropout (0.2)
Hidden layer dropout (0.3)
0.9
0.2
0.1
0.6
0.2
0.3
0.0
0.7
0.1
0.4
0.4
0.4

Figure 6: Visualization of the ConvE approach, taken from [13]

**Neural network models** utilize deep learning architectures to model and learn KG embeddings. ConvE [13] applies a multi-layer convolutional architecture to learn KG embeddings and perform link prediction. The approach is illustrated in figure 6. In particular, the head $h$ and relation $r$ embeddings are initially reshaped into a 2D representation (denoted as $\bar{h}, \bar{r}$) and concatenated. Afterwards, the resulting matrix is passed through a 2D convolution layer equipped with several filters $w$ that yield a feature map tensor. The tensor is vectorized and projected into a k-dimensional space using linear transformation. To conclude, it is compared with the embeddings of all potential tail entities. The scoring function of ConvE can be defined as follows:

$$f_r(h,t) = g(vec(g(concat(\bar{h},\bar{r}) * w))W)t \tag{2.5}$$

where $g$ refers to a nonlinear function, $vec(\cdot)$ denotes the tensor vectiorization, $*$ represents convolution and $W$ refers to the linear transformation matrix [45].

Other neural network-based KGE methods include R-GCN [52] and CompGCN [53] which apply Graph Convolution Networks (GCN) to learn representations for entities and relations considering their neighbour connections in the KG [54].

## 2.2 Neural Language Models

Natural Language Processing (NLP) is a subfield of computer science that is concerned with enabling computers to understand and represent human languages [55]. Converting human language into numerical form, while capturing its meaning, is essential to applying machine learning algorithms to textual data. This numerical vector representation is known as word embedding. Word embeddings can be defined as fixed-length numerical vectors in a predefined vector space representing individual words [56]. Within the scope of this thesis, the terms `word embedding`, `KG embedding` and `vector representation` are used interchangeably. According to Almedia et al. [56], word embedding methods can be broadly categorized into *count-based models* and *prediction-based models* depending on the embedding generation strategy. Count-based models analyze word co-occurrence statistics, whereas predictive models leverage a word's context. This thesis focuses on predictive methods, specifically on neural network-based language models.

Neural Language Models (NLM) employ neural networks to learn the distributed representation of words as low-dimensional vectors and use these to estimate the probability of word sequences [57]. NLMs are widely used across many NLP applications such as speech recognition, question answering and sentiment analysis [58]. NLM-generaled word embeddings can be categorized into non-contextual and contextual embeddings [59]. Non-contextual embeddings are static in nature, maintaining the same representation regardless of the context in which the word is used. In contrast, contextual embeddings dynamically adjust the representations based on the surrounding context, allowing multiple embeddings for the same word. NLMs are typically trained on vast amounts of unlabeled text data, substantially increasing the number of the model's parameters. This can often lead to overfitting or limited generalization capabilities. To address these challenges, pre-training of NLMs is carried out on extensive text corpora, focusing on general tasks. The knowledge captured in pretrained NLMs has proven to be beneficial for a variety of downstream tasks. Therefore, it is a common practice to fine-tune a pretrained NLM for a specific task, enabling the model to adapt and perform effectively in task-specific scenarios [60].

### Word2Vec

Word2Vec [61] is a prominent technique that utilizes a shallow neural network to learn non-contextual word embeddings. It employs two different approaches, illustrated in figure 7, namely the Continuous Bag-of-Words (CBOW) and the Continuous Skip-gram. Both approaches share a similar structure, consisting of an input layer, an output layer, and a single hidden layer. The input layer represents the one-hot encoded word vectors. The hidden layer performs a linear transformation by computing the dot product between the input vectors and the weight matrix connecting the input and hidden layer. The output layer is a softmax layer that calculates the probabilities of predicting each word in the vocabulary. The CBOW model predicts the target word $w_t$ by considering the surrounding context words within a fixed-length window. The Continuous Skip-gram model reverses the task by predicting the context words within a fixed-length window given a target word.

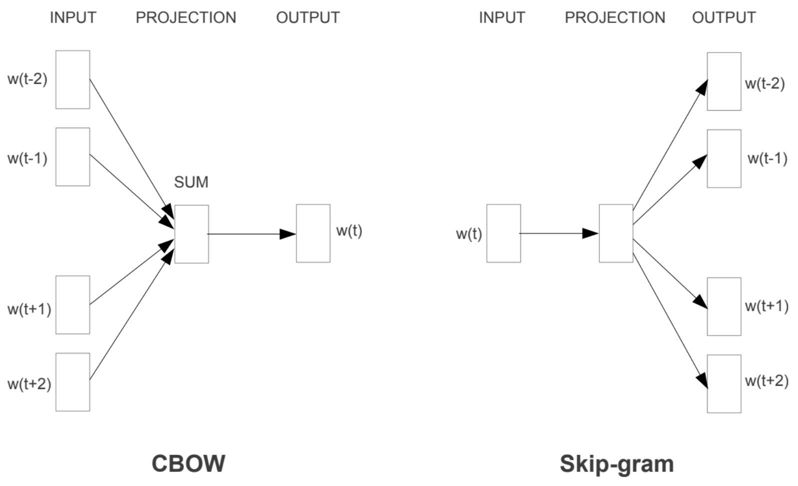


Figure 7: The Word2Vec model architecture, taken from [61]

### BERT

Bidirectional Encoder Representations from Transformers (BERT) is a context-aware NLM introduced by Devlin et al. [17]. It employs a transformer architecture that relies on self-attention mechanisms to model dependencies between words in a sentence. The attention mechanism was introduced by Vaswani et al. [62] and enables a neural network to weigh the importance of different input elements when processing sequential data, allowing the model to focus more on relevant information. BERT$_{base}$ consists of 12 Transformer encoder layers with a hidden dimension of 768 and a total of 110 million parameters [17]. The key novelty of BERT lies in its bidirectional training approach, allowing it to capture contextual information from both left and right contexts simultaneously, leading to a deeper understanding of language.

BERT is able to represent as input both single sentences and pairs of sentences within a single token sequence. In the context of BERT, Devlin et al. define a *sentence* as any continuous span of text and *sequence* as the token sequence given as input to BERT, which can be a single sentence or two sentences combined. BERT uses WordPiece embeddings [63] including a 30.000 token vocabulary. The model's input is processed as follows. Firstly, the input text is divided into individual tokens or subwords using WordPiece tokenization. A special classification [CLS] token is inserted at the beginning of every sequence. Sentence pairs are combined into a single sequence. A [SEP] token is inserted between the two sentences to separate them. Furthermore, every token of the input sequence is associated with a so-called segment embedding that indicates to which sentence the token belongs. Additionally, a positional embedding is included with each token to indicate its position in the sequence. To conclude, the input embeddings are calculated as the sum of the token embeddings, the segment embeddings and the position embeddings, as shown in figure 8.

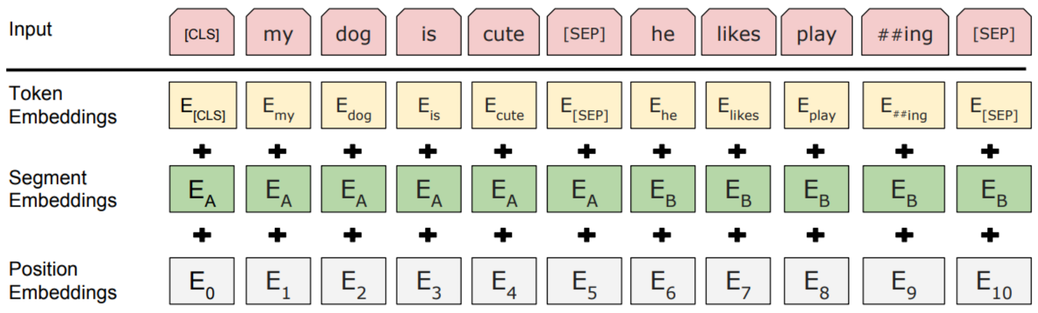

Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ $E_{my}$ $E_{dog}$ $E_{is}$ $E_{cute}$ $E_{[SEP]}$ $E_{he}$ $E_{likes}$ $E_{play}$ $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
$E_A$ $E_A$ $E_A$ $E_A$ $E_A$ $E_A$ $E_B$ $E_B$ $E_B$ $E_B$ $E_B$
Position Embeddings
$E_0$ $E_1$ $E_2$ $E_3$ $E_4$ $E_5$ $E_6$ $E_7$ $E_8$ $E_9$ $E_{10}$

Figure 8: BERT input representation, taken from [17]

The BERT model is pretrained on large amounts of unlabeled data sourced from the BooksCorpus and the English Wikipedia on two unsupervised tasks: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). The idea behind MLM is to randomly mask a percentage of the input tokens and then predict these masked tokens based on the context provided by the other, non-masked words in the sequence. For BERT specifically, 15% of all tokens in each sequence at random are masked. In NSP, BERT learns to understand relationships between sentences by predicting whether the second sentence in the pair is the subsequent sentence in the original document.

RoBERTa is an optimized variant of the BERT model introduced by Liu et al [22]. It employs a similar architecture to BERT but incorporates modifications in the training process. RoBERTa is trained longer, over more data and longer sequences, and it utilizes dynamic masking during pre-training. DistilBERT, developed by Sanh et al. [23], is a distilled version of the BERT model. It aims to retain most of the performance of BERT while reducing the model's size and computational requirements. DistilBERT achieves this by applying a knowledge distillation technique, where it is trained to mimic the behaviour of the larger BERT model.

### BLOOM

BLOOM [24] is an open-access multilingual language model that employs a causal decoder-only Transformer architecture. It is trained with the objective of predicting the next token that will follow in a sentence. BLOOM is trained on the ROOTS corpus [64], a collection of datasets covering 46 natural languages and 13 programming languages. The BLOOM tokenizer is learned using byte pair encoding, whereby frequent words are tokenized as single tokens, while less frequent words are split into subwords and represented by multiple tokens. The tokenizer comprises of a vocabulary of 250680 words.

## 2.3 Evaluation Metrics

Commonly applied evaluation metrics for link prediction are `Hits@k` and `Mean Reciprocal Rank (MRR)` [11]. `Hits@k` measures the proportion of correct triples within the top $k$ ranked predictions and is calculated as follows:

$$H@k = \frac{|\{q \in Q : q < k\}|}{|Q|}, \tag{2.6}$$

where $q$ refers to the current prediction and $Q$ denotes all the predictions made by the model. It indicates the accuracy of the model by considering if a true prediction is among the top $k$ predictions. Typically, Hits@k is reported for $k = 1, 3, 10$. `MRR` grades the predicted triples depending on whether these are true or not and is computed as follows:

$$MRR = \frac{1}{|Q|} \sum_{q \in Q} \frac{1}{q} \tag{2.7}$$

To evaluate the performance of AI models on classification tasks, metrics such as `Accuracy`, `F1-Score`, `Precision` and `Recall` are used [65]. `Accuracy` measures the proportion of correct predictions (both true positives and true negatives) among the total number of instances evaluated. It gives a general measure of the model's overall performance. `Precision` denotes the ratio of correctly predicted positive instances to the total predicted positives. `Recall` refers to the ratio of correctly predicted positive instances to the total positive instances, indicating how well the positive class is predicted. The `F1-Score` is the harmonic mean of precision and recall.

# 3 Related Work

To handle the incompleteness of KGs, research exploits machine learning techniques and low-dimensional representation of KG entities and relations. For this purpose, numerous approaches, such as TransE [12], DistMult [48] or ComplEx [49] have been proposed. These models rely predominantly on the structural information inherent in the KG to perform KGC tasks such as link prediction and triple classification.

With the emergence of pretrained NLMs such as BERT [17], RoBERTa [22] or GPT-2 [18] research is gradually shifting towards leveraging the knowledge present in these language models for KGC. Aiming to integrate the contextual information captured in word embeddings generated by pretrained NLMs, Ma et. al [66] suggest a novel KGE approach that forms KG embeddings as a linear combination of NLM-derived word embeddings, achieving promising results in detecting unknown facts, link prediction and triple classification. Alam et. al [67] propose integrating NLM-generated word embeddings into the loss function of an existing KGE model. The plausibility of triples based on the word embeddings serves as guidance during the learning process of the KGE. Specifically, the loss function is modified in such way that it not only penalizes the KGE model for predicting incorrect relationships between entities but also for generating entity and relation embeddings that are too dissimilar from the respective NLM embeddings. In contrast, Zhang et al. [20] present a different strategy: they initialize a KGE model using word embeddings of KG entities and relations generated by a pretrained NLM, thereby exploring a novel way to enhance the performance of KGEs in KGC tasks. Their approach is evaluated for the KGE models TransE, DistMult, ComplEX, QuatE, RotatE on the task of link prediction. Through numerous experiments on multiple benchmark datasets such WN18RR and FB15K237, Zhang et al. demonstrate that the proposed approach outperforms traditional KGE models that use random initialization.

Furthermore, Yao et al. [21] propose a novel framework that relies solely on the pretrained BERT model to model triples' plausibility, named KG-BERT. It is fine-tuned with KG triples to perform triple classification, link prediction and relation prediction tasks. In particular, each triple `<head, relation, tail>` is treated as a separate input sequence and the model is trained with entity and relation labels and descriptions. When evaluated on three benchmark datasets for link prediction, KG-BERT notably improves Mean Rank (MR) and achieves comparable Hits@10 results to conventional KGE models such as TransE. Similar to KG-BERT, Biswas et al. [68] introduce a GPT -2-based approach for KGC. GPT-2 is fine-tuned on KG triples and finalized with a sigmoid scoring function on top of the model's final layer to enable sequence classification. The approach is evaluated on the task of triple classification for two benchmark datasets.

# 4 Methodology

This chapter outlines the methodology used to investigate the applicability of pretrained NLMs for KGC. Two distinct approaches are explored: (1) utilizing pretrained NLM embeddings of entities and relations as initialization vectors for a given KGE model, and (2) fine-tuning a pretrained NLM with KG triples for triple classification. Both approaches are applied in the context of KGC tasks.

## 4.1 NLM-enhanced Knowledge Graph Embedding Models

The first approach consists of using pretrained NLM-generated embeddings of KG entities and relations as initialization vectors for a given KGE model, rather than the commonly applied random initialization. The core idea of this approach suggests that the contextual information drawn from pretrained NLMs, as reflected in the word embeddings, could potentially result in superior KG embeddings when compared to randomly initialized vectors. In broad terms, the approach can be summarized as follows. For a given KG, the entity and relation labels together with the respective textual descriptions are extracted. Entity and relation embeddings are generated for two different scenarios: `labels only` and `labels & descriptions`. For this purpose, the context-independent Word2Vec model and the context-aware BERT model are used. The resulting entity and relation embeddings are used as initialization vectors to train a KGE model. To conclude, the KGE model is evaluated on the task of link prediction. Table 1 summarizes the explored embedding initialization strategies.

| **Method** | **Labels** | **Labels & descriptions** |
|---|---|---|
| Word2Vec | yes | no |
| BERT | yes | yes |
| BERT PCA | yes | no |

Table 1: Overview of embedding strategies for KGE model initialization

In the following, the process of generating pretrained embeddings, which varies depending on the employed NLM, is described.

### Pretrained Word2Vec Embeddings

Since Word2Vec is a context-independent language model, individual words need to be matched against the pretrained Word2Vec embeddings to retrieve the corresponding vector representation. Therefore, to obtain the embeddings for the KG entities and relations, their labels are split into individual words, as depicted in the first two steps in figure 9. The embeddings of the respective separate words are retrieved and averaged, resulting in a vector representation of dimension `(1, embedding dimension)` for each entity and relation. Following the example provided in figure 9, the embedding of the entity "german shepherd dog" is computed by averaging the embeddings of the individual words "german", "shepherd", and "dog". Words lacking a Word2Vec embedding are excluded. Furthermore, for entity or relation labels without any match to the pretrained embeddings, the first five words of the corresponding textual description are included in order to compute the averaged embedding.

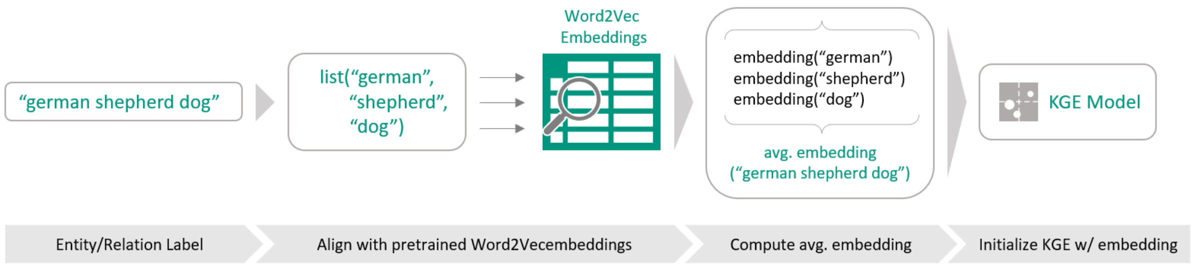


Figure 9: Generation of pretrained Word2Vec embeddings

### Pretrained BERT Embeddings

As a powerful context-aware language model, BERT effectively encodes meaningful information about a word and its surrounding context within the word embeddings. For a given KG, the BERT entity and relations embeddings are computed for both scenarios: `labels only` and `labels & descriptions`. Figure 10 illustrates the BERT embedding retrieval process within the context of KGE model initialization.

For every entity and relation, labels (labels & descriptions) are treated as separate input sequences and are tokenized accordingly. However, since textual descriptions often surpass the limit of 512 tokens imposed by the BERT model, the tokenized descriptions are restricted to the first 512 tokens. Moreover, the tokenized entities and relations are fed to the pretrained BERT model to retrieve the embeddings. BERT outputs embeddings of dimension `(#layers, #batches, #tokens, embedding dimension)` requiring some adjustment to match the input format of the KGE models, since these work on dimensionalities of `(#entities, embedding dimension)` for entities, and `(#relations, embedding dimension)` for relations. Firstly, the batch dimension is resolved by stacking the embeddings across all batches for each layer. Secondly, only the CLS token embedding is further considered, since it is used as the aggregate sequence representation for classification tasks [17]. Lastly, to achieve the desired format of the an entity's or relation's embedding, the CLS token embeddings of the last four hidden layers are averaged.

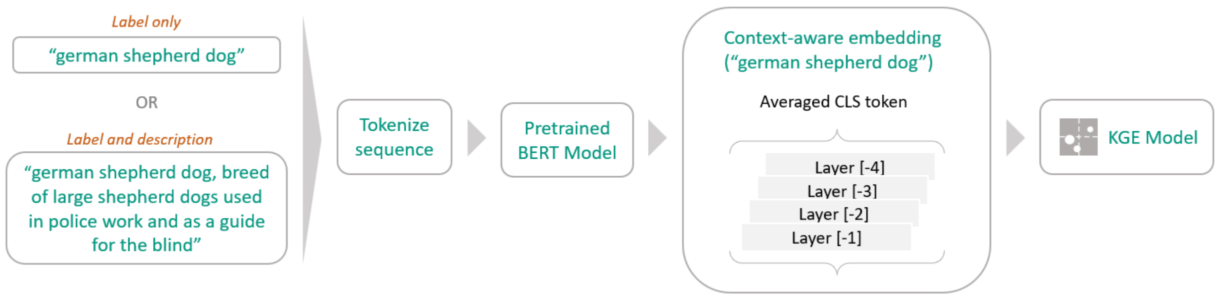


Figure 10: Generation of pretrained BERT embeddings

In commonly used state-of-the-art KGE models, the employed embedding dimensions are smaller, generally ranging from 200 to 500, when compared to BERT's dimensionality, which is 768 [14, 13, 12]. Therefore, an additional initialization strategy involving the application of a dimensionality reduction technique to the already computed BERT embeddings is evaluated.

## 4.2 Pretrained NLMs for Knowledge Graph Completion

The second approach, named `KG-NLM`, explores the capability of pretrained contextual NLMs to perform KGC. To accomplish this, several pretrained NLMs with an added linear layer for sequence classification are fine-tuned on the task of triple classification using a KG's triples. In the context of NLMs, the triple classification task can be understood as a sequence classification task. Hence, each KG triple `(head, relation, tail)` is considered as one input sequence. Therefore, the triples are extracted in textual form `(labels only)` and are adjusted to string sequences. The extracted triples represent the positive samples in the classification dataset, while the negative triples are generated using a negative sampling technique. The latter generates a negative triple for each positive triple by corrupting either the head or the tail entity. Due to the randomized nature of the negative sampling process, positive triples might be generated, however, these are excluded from the final dataset. The positive and the negative triples are joined into one dataset and shuffled to ensure a random distribution of both classification labels.

Figure 11 visualizes the entire process of training the `KG-NLM` models with KG triples. As depicted, a KG triple, for example `(Albert Einstein, place of birth, Germany)`, is arranged into a string sequence. The start of a sequence is identified with a special token, denoted here as `BOS`, inserted before the head entity. Similarly, an additional special token is used to mark the end of a sequence, denoted as `EOS`, placed after the tail entity. Specifically, for classification tasks, a special classification token, denoted as `CLS`, is inserted either at the beginning of the sentence or at the end, depending on the selected tokenizer. In this case, the `CLS token` replaces the `BOS` or `EOS token`. During tokenization, the input sequence is split into full forms, where each word is assigned a token, or into word pieces, where one word can be split into multiple tokens. Consequently, the tokenized input sequence, along with the respective classification label is inputted to the pretrained NLM. To perform classification, a classification head is introduced, which comprises a compact Multi-Layer Perceptron (MLP) featuring two dense layers with a non-linear activation function. This classification head is appended to the CLS-token embeddings to generate a probability distribution through a softmax layer, enabling the determination of class probabilities.

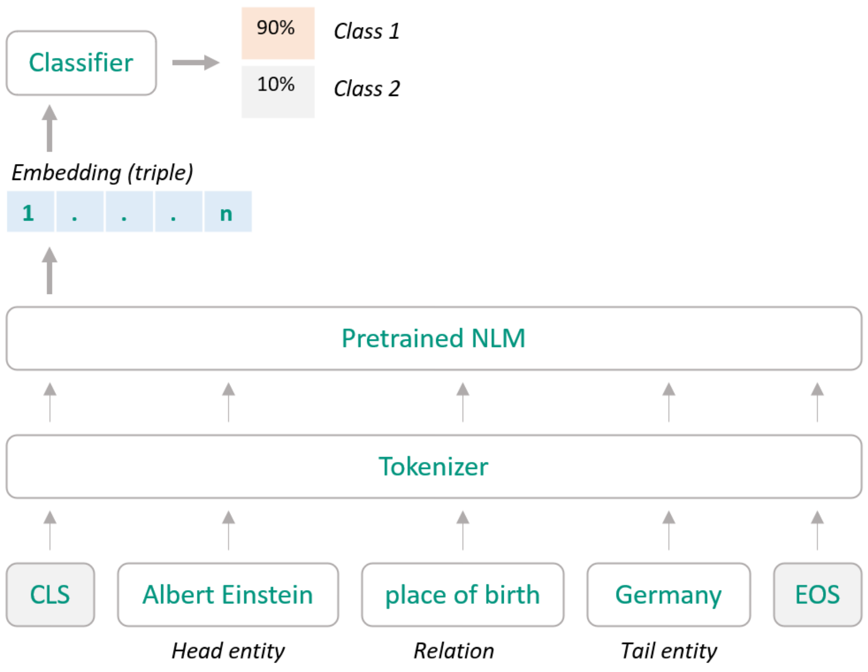


Figure 11: Visualization of the `KG-NLM` approach for KGC

For training the `KG-NLM` models, two strategies are used: the end-to-end training approach and the frozen NLM layers approach. In the former, backpropagation is permitted through all model layers, including those of the pretrained NLM, allowing the update of weights across the entire model. Conversely, in the approach where the layers of the pretrained NLM are frozen, the weights within these specific layers remain static and are not subject to updates.

# 5 Evaluation

This chapter presents the datasets utilized in this thesis and delves into the experimental setup of the methodology introduced in chapter 4, covering the explored KGE models and pretrained NLMs. The chapter concludes with the presentation and discussion of the results associated with both approaches.

## 5.1 Datasets

The datasets utilized in this thesis include WN18RR [13] and FB15k-237 [69]. WN18RR is a modified version of the WN18 dataset [12], which is a subset of the WordNet KB. The WN18 dataset faces the issue of test leakage due to inverse relations, meaning that in many triples of the test set, the head and tail entities are swapped. To address this problem, Dettmers et al. [13] introduce the WN18RR dataset, which mitigates the issue by eliminating such inverse relations. The WN18RR dataset is formed by extracting WordNet triples in the format `(synset, relation, synset)` to represent the information contained in WordNet, as described in chapter 2.1.

FB15k-237, a widely recognized dataset in KGC, contains triples that capture relationships between entities in a KG derived from the Freebase KB. Originally introduced by Bordes et al. in the FB15k dataset [12], FB15k-237 is a modified version restricted to the most frequently used relations while excluding near-duplicates and inverse ones. The statistics of both datasets are provided in table 2. For every dataset, the pre-computed train, validation and test sets available at PyKEEN Datasets[1] are used. As described in chapter 4.1, the pretrained entity and relation embeddings are generated for labels and descriptions respectively. The labels and descriptions in textual form are extracted from Yao et. al[2] [21].

| **Dataset** | **Nr. Entities** | **Nr. Relations** | **Nr. Triples** | | | |
|---|---|---|---|---|---|---|
| | | | Train | Validation | Test | Total |
| WN18RR | 40559 | 11 | 86835 | 2924 | 2824 | 92583 |
| FB15k-237 | 14505 | 237 | 272115 | 17526 | 20438 | 310079 |

Table 2: Datasets overview

[1]PyKEEN Datasets: https://pykeen.readthedocs.io/en/stable/reference/datasets.html
[2]Git Repository : https://github.com/yao8839836/kg-bert/tree/master/data

## 5.2 Experimental Setup

In the following, the implementation choices related to the explored approaches described in chapter 4 are discussed. This chapter provides an overview on the selected KGE models and NLMs, including the choice of the hyperparameters and the necessary dataset preprocessing steps tailored to each approach. All code is implemented in Python (3.9.7) and run on a NVIDIA Tesla V100.

### NLM-enhanced KGE Models

To carry out the experiments related to the first approach (see chapter 4.1), the PyKEEN[3] package is used. To generate the word embeddings, the following pretrained models are employed:

- Word2Vec pretrained on approximately 100 billion words from the Google News dataset, including 300-dimensional embeddings for about 3 million words and phrases.
- BERT pretrained on a vast amount of data from BooksCorpus and the English Wikipedia as described in section 2.2. The pretrained BERT model outputs word embeddings with a dimensionality of 768 for each of the model's hidden layers.

The impact of incorporating NLMs into KGE models is evaluated on the following models: TransE, TuckER, ConvE and RotatE. Table 3 provides an overview of the pretrained embedding strategies employed for each KGE model.

| **KGE Model** | **Embedding Method** | **Dimension** |
|---|---|---:|
| TransE, ConvE, TuckER | BERT (labels) | 768 |
| | BERT (labels & descriptions) | 768 |
| | BERT PCA (labels) | 200 |
| | Word2Vec | 300 |
| RotatE | BERT (labels) | 768 |
| | Word2Vec | 150 |

Table 3: Overview of pretrained embeddings for KGE model initialization

As outlined in chapter 4.1, BERT embeddings are generated for KG labels and labels with descriptions by averaging the last four hidden layer embeddings of the CLS token. For the initialization employing BERT embeddings with reduced dimensionality, the Principal Component Analysis (PCA) method is utilized. PCA is a statistical technique used to

[3]PyKEEN [70] is an open-source Python library that provides a comprehensive framework for KGE including various KGE models, datasets, and evaluation metrics.

reduce the dimension of datasets while preserving as much information as possible. It achieves this by generating new uncorrelated variables that successively maximize variance [71]. PCA is applied to both entity and relation embeddings, which are derived by averaging the embeddings from the last four hidden layers of BERT. Prior to inputting the NLM-generated embeddings into the KGE model, they are aligned to the sequence order of the entities and relations in the built-in PyKEEN datasets.

In the case of RotatE, where entities and relations are mapped to a complex vector space (see chapter 2.1.2), additional processing of the pretrained embeddings is required. For RotatE, the default initialization of entity and relation representations involves generating random real number vectors twice the length of the predefined dimension. This means that if an embedding of length 100 is needed for a word like "dog" RotatE would generate a randomized vector of size 200. The first 100 entries represent the real part of the complex vector, while the second 100 entries represent the imaginary part. As a result, it is not possible to directly apply pretrained embeddings of length 300 (Word2Vec) or 768 (BERT), which consist of real numbers, to the RotatE model. Therefore, Word2Vec entity embeddings are transformed into complex vectors by splitting them into the first 150 entries as the real part and the second 150 entries as the imaginary part, resulting in a complex vector of length 150. The Word2Vec relation embeddings are dimensionally reduced to 150 using the PCA method. The resulting embedding vector is used to compute the real and the imaginary part of the complex relation representation, as described in the appendix section A.1. In the case of pretrained BERT embeddings, the CLS token is once again employed. The last hidden layer of BERT is utilized as the real part of the complex vector, while the second-to-last hidden layer is utilized as the imaginary part.

The models are trained with the hyperparameters reported in the papers corresponding to the baseline models. This is the case for the models ConvE, TuckER and RotatE. As for TransE, the original paper hyperparameters are not available, thus the hyperparameters used in this thesis are taken from Ali et al.'s [72], who conduct a comprehensive benchmarking study on multiple datasets and KGE models, providing the best achieved hyperparameters for each model-dataset combination. Furthermore, a limited-resource hyperparameter optimization is carried out for the models TransE and ConvE. Computational resources are limited to 24 hours run time or a maximum of 10 trials per model. The employed hyperparameter configurations are denoted as follows:

- `paperP` denotes hyperparameters from the original papers.
- `benchP` refers to hyperparameters derived from Ali et al.'s [72] study.
- `hpoP` refers to a set of hyperparameters acquired through limited resource hyperparameter optimization within the scope of this thesis.

The detailed hyperparameter selection is provided in the appendix section A.2. To conclude, the performance of the KGE models on the task of link prediction is measured using the following metrics: Hits@k for $k = 1, 3, 10$ and MRR. All the models are trained using early stopping on Hits@10, evaluated every 50 epochs and with a patience of 3.

**KG-NLM**

The `KG-NLM` approach consists of evaluating different NLMs for KGC on the task of triple classification, resulting in the following models: `KG-RoBERTa`, `KG-DistilBERT` and `KG-BLOOM`. The pretrained NLMs explored in this thesis include `roberta-base`, `bloom-560m` and `distilbert-base-uncased`. Each dataset is preprocessed as described in chapter 4.2 using a uniform negative sampling technique[4]. The models are trained for 1 and 3 epochs respectively, employing AdamW optimization and a learning rate of $5e^{-5}$. The complete list of the selected hyperparameters is presented in table 4.

| **Hyperparameter** | **Value** |
|---|---:|
| Nr. of epochs | 1, 3 |
| Train batch size | 8 |
| Gradient accumulation steps | 32 |
| Evaluation batch size | 8 |
| Evaluation strategy | steps |
| Warmup steps | 500 |
| Learning rate | 5e-5 |
| Learning rate scheduler | linear |

Table 4: Hyperparameter selection for the `KG-NLM` models

The model's performance on triple classification is evaluated using the metrics accuracy, recall, precision and f1-score.

## 5.3 Results

### 5.3.1 NLM-enhanced Knowledge Graph Embedding Models

To investigate the influence of pretrained NLM-generated embeddings in the learning process of existing KGE methods, several experiments following the approach described in chapter 4.1, are conducted. The evaluation results for each model and the selected initialization strategies are presented in table 5 for WN18RR and table 6 for FB15k-237. The displayed results represent the best-performing hyperparameter choice for each

[4]Negative sampling from PyKEEN: https://pykeen.readthedocs.io/en/stable/reference/negative_sampling.html

model and dataset. Additional results are presented in table 12 in the appendix. The hyperparameter notation is defined in section 5.2, while the hyperparameter selection is provided in detail in the appendix chapter A.2.

| **Model** | **Initialization Embeddings** | **Hyper -parameters** | **Nr. epochs** | **Metric** Hits@1 | Hits@3 | Hits@10 | MRR |
|---|---|---|---|---|---|---|---|
| TransE baseline [73] | | | | 0.043 | 0.441 | 0.532 | 0.243 |
| TransE | Word2Vec | benchP | 100* | **0.067** | 0.375 | **0.57** | **0.249** |
| | BERT (labels) | benchP | 500 | **0.056** | 0.323 | **0.536** | 0.222 |
| | BERT PCA (labels) | benchP | 500 | **0.074** | 0.389 | **0.571** | **0.256** |
| | BERT (labels & descriptions) | benchP | 500 | 0.044 | 0.319 | 0.529 | 0.214 |
| ConvE baseline [13] | | | | 0.4 | 0.44 | 0.52 | 0.43 |
| ConvE | Word2Vec | hpoP | 700* | 0.190 | 0.294 | 0.360 | 0.253 |
| | BERT (labels) | hpoP | 500 | 0.025 | 0.068 | 0.121 | 0.058 |
| | BERT PCA (labels) | hpoP | 1000 | 0.15 | 0.235 | 0.314 | 0.206 |
| | BERT (labels & descriptions) | hpoP | 500 | 0.035 | 0.088 | 0.149 | 0.074 |
| TuckER baseline [14] | | | | 0.443 | 0.482 | 0.526 | 0.47 |
| TuckER | Word2Vec | paperP | 1000 | **0.453** | **0.491** | 0.522 | **0.478** |
| | BERT PCA (labels) | paperP | 1000 | **0.46** | **0.491** | 0.511 | **0.485** |
| RotatE baseline [46] | | | | 0.428 | 0.492 | 0.571 | 0.476 |
| RotatE | Word2Vec | paperP | 1500 | 0.427 | 0.487 | 0.564 | 0.472 |
| | BERT (labels) | paperP | 500 | 0.187 | 0.315 | 0.358 | 0.259 |

Table 5: WN18RR - Evaluation results of NLM-enhanced KGE models. Number of epochs marked with * signifies early stopping.

For the WN18RR dataset, TransE shows a better performance compared to the baseline in terms of Hits@1, Hits@10 and MRR for three of the initialization strategies: `Word2Vec`, `BERT (labels)` and `BERT PCA (labels)`. Using BERT embeddings generated for both labels and descriptions to initialize TransE produces comparable results to the baseline. Similar results can be observed for TuckER, where initialization with `Word2Vec` and `BERT PCA (labels)` embeddings leads to slightly better performance than the baseline concerning Hits@1, Hits@3 and MRR. The utilization of pretrained NLM-generated embeddings for ConvE seems to lead to a decline in the model's performance in the case of the WN18RR dataset. Experimental results highlight that ConvE's performance is significantly affected by the embedding dimensionality. Initializing ConvE with `BERT (labels)` of 768 dimensions results in poor performance across all metrics when compared to initializing it with `BERT PCA (labels)` of dimension 200. In terms of RotatE, the initialization using either `Word2Vec` or `BERT (labels)` embeddings yields comparable outcomes in the long run. However, it is important to highlight that RotatE utilizing `Word2Vec` embeddings is trained over a greater number of epochs relative to the model initialized with `BERT (labels)`.

| **Model** | **Initialization** | **Hyper** | **Nr.** | **Metric** | | | |
|---|---|---|---|---|---|---|---|
| | **Embeddings** | **-parameters** | **epochs** | Hits@1 | Hits@3 | Hits@10 | MRR |
| TransE baseline [73] | | | | 0.198 | 0.376 | 0.441 | 0.279 |
| TransE | Word2Vec | hpoP | 250* | 0.148 | 0.245 | 0.373 | 0.223 |
| | BERT (labels) | hpoP | 250* | 0.115 | 0.184 | 0.274 | 0.17 |
| | BERT PCA (labels) | hpoP | 500 | 0.165 | 0.266 | 0.402 | 0.245 |
| | BERT (labels & descriptions) | hpoP | 100* | 0.126 | 0.206 | 0.306 | 0.188 |
| ConvE baseline [13] | | | | 0.237 | 0.356 | 0.501 | 0.325 |
| ConvE | Word2Vec | paperP | 350* | 0.144 | 0.212 | 0.297 | 0.196 |
| | BERT (labels) | paperP | 500 | 0.128 | 0.188 | 0.265 | 0.174 |
| | BERT PCA (labels) | paperP | 500 | 0.147 | 0.212 | 0.295 | 0.172 |
| | BERT (labels & descriptions) | paperP | 500 | 0.125 | 0.186 | 0.264 | 0.197 |
| TuckER baseline [14] | | | | 0.266 | 0.394 | 0.544 | 0.358 |
| TuckER | Word2Vec | paperP | 500 | 0.185 | 0.273 | 0.377 | 0.250 |
| | BERT (labels) | paperP | 500 | 0.197 | 0.293 | 0.396 | 0.265 |
| | BERT PCA (labels) | paperP | 500 | 0.176 | 0.257 | 0.351 | 0.235 |
| | BERT (labels & descriptions) | paperP | 500 | 0.198 | 0.291 | 0.395 | 0.265 |
| RotatE baseline [46] | | | | 0.241 | 0.357 | 0.533 | 0.338 |
| RotatE | Word2Vec | paperP | 2000 | 0.229 | 0.35 | 0.496 | 0.318 |
| | BERT (labels) | paperP | 500 | 0.213 | 0.328 | 0.471 | 0.299 |

Table 6: FB15k-237 - Evaluation results of NLM-enhanced KGE models. Number of epochs marked with * signifies early stopping.

For the FB15k-237 dataset, all models and initialization strategies demonstrate lower performance compared to the baselines. With the exception of TransE, it is evident that the embedding dimensionality has less impact on the model's performance for this dataset. For every model, the employed initialization strategies result in highly similar performances across all metrics.

## Discussion

While the performance of TransE on the WN18RR dataset surpasses the baseline, the majority of the results indicate no improvement in model performance when initializing with pretrained NLM-generated embeddings. It appears that the knowledge encapsulated by pretrained NLMs within the entity and relation embeddings do not assist the KGE models explored in this thesis in learning better KG embeddings compared to random initialization vectors. This could be attributed to the distinct learning methods employed by KGE models and NLMs. KGE models learn KG embeddings through patterns inherent in the graph structure, rather than through language understanding as in the case of NLMs. This divergence in the generation of KG embeddings could potentially explain the difficulty in transferring the knowledge between the two. Another reason could be the choice of the hyperparameters. In this thesis, restricted hyperparameter optimization, limited to 24 hours run time or a maximum of 10 trials per model, has been carried out. However, achieving a high-quality set of hyperparameters requires more extensive computational resources. The right hyperparameter configuration, tailored to the initialization with pretrained embeddings could lead to higher KGE model performance.

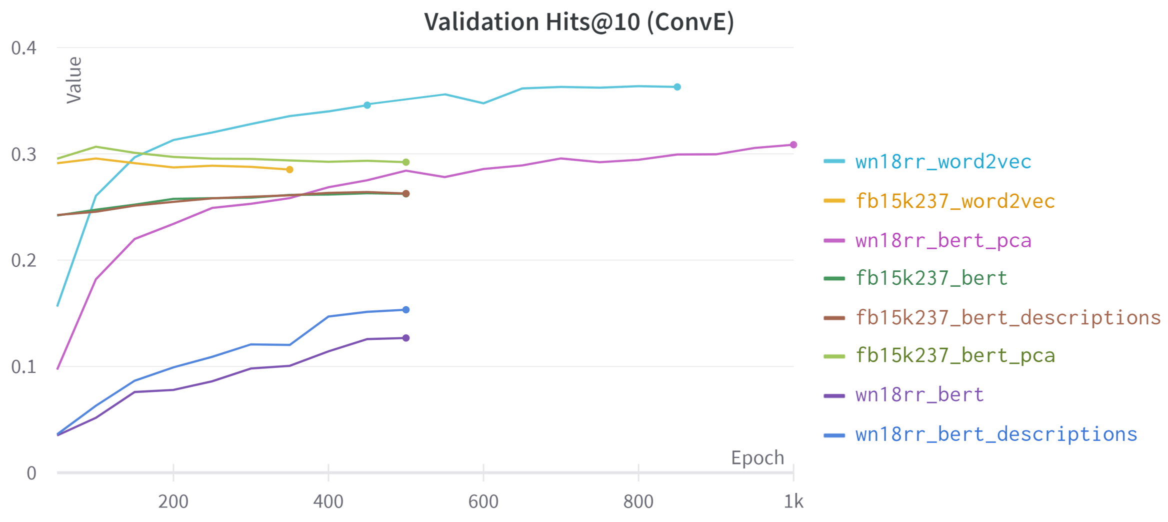


Figure 12: ConvE performance over the number of epochs

Among the explored BERT embedding strategies: `BERT (labels)`, `BERT PCA (labels)` and `BERT (labels & descriptions)`, the results show that most models perform better with `BERT PCA (labels)` embeddings. Figure 12 makes this observation more explicit by displaying the Hits@10 curves for the ConvE model's training phase under different initialization strategies. While ConvE is trained for 500 epochs for most of the initialization strategies, the combinations `wn18rr_word2vec` and `wn18rr_bert_pca` are trained for 1000 epochs due to more promising results. For each dataset, the `BERT PCA (labels)` curve lies well above the other BERT initialization curves, indicating a better performance since the early training epochs. This can be attributed to the high dimensionality of BERT embeddings, which adds to the complexity of the KGE models. Typically, these models are designed to function optimally in lower dimensional spaces.

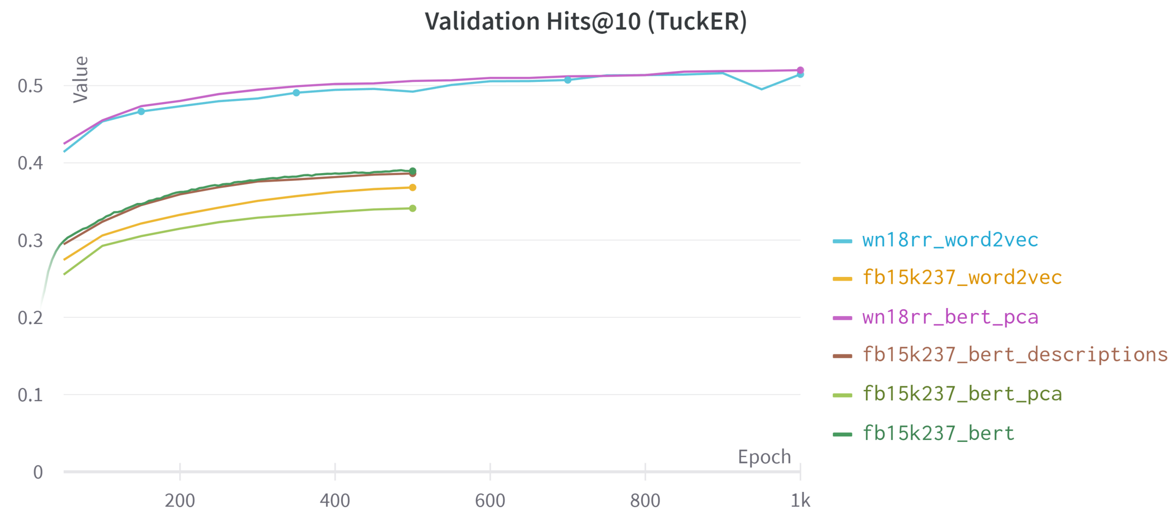


Figure 13: TuckER performance over the number of epochs

Moreover, for the models TransE, ConvE and TuckER, no substantial difference is observed in the model's performance when using context-aware (BERT) or context-independent (Word2Vec) embeddings. The performances of the `BERT (labels)` and `BERT (labels & descriptions)` initialization strategies are comparable, visualized in figures 12 and 13. This suggests that embedding additional context does not necessarily enhance the performance of these models. In the case of RotatE, initialization with BERT embeddings leads to a faster model convergence, as demonstrated in figure 14.

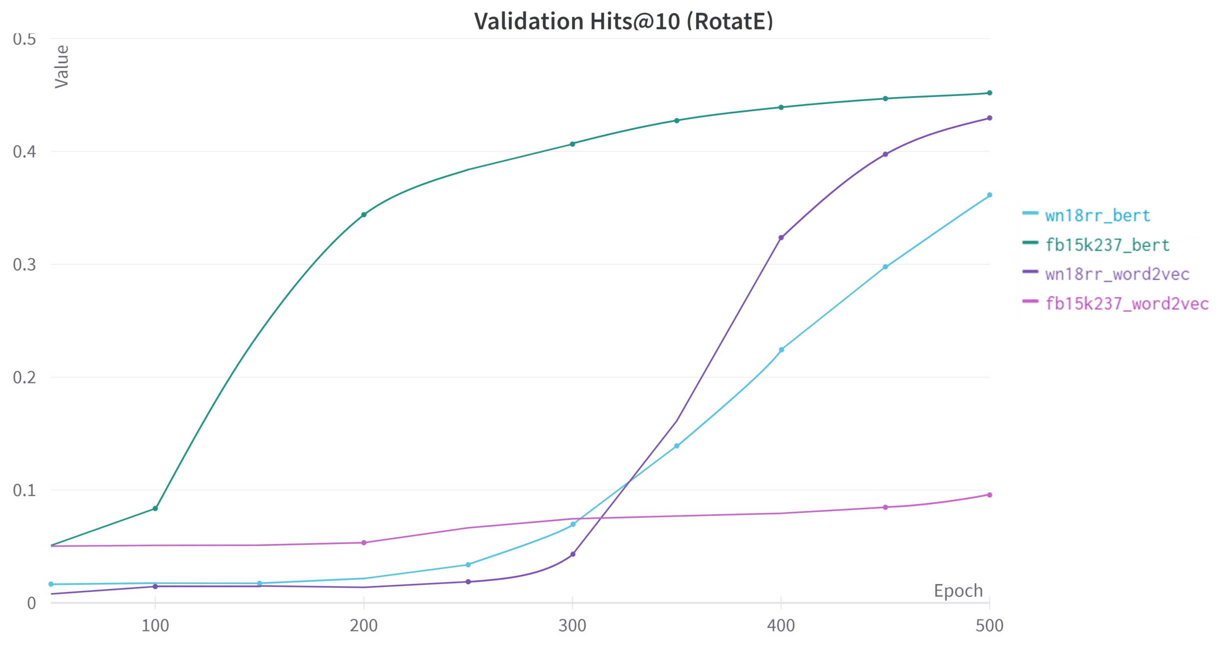


Figure 14: RotatE performance over the number of epochs

### 5.3.2 Pretrained NLMs for Knowledge Graph Completion

To assess the performance of pretrained NLMs for KGC, several experiments employing multiple NLMs for triple classification are conducted. The general approach is outlined in section 4.2, while the experimental setup is described in section 5.2. The results are presented in table 7 for the WN18RR dataset and table 8 for the FB15k-237 dataset. The outcomes of the models trained with frozen layers of the pretrained NLM are omitted due to their inferior performance. Based on existing literature, the pretrained NLMs involved in this thesis have not been yet evaluated on the task of triple classification on the WN18RR and FB15k-237 datasets. The selected baseline model, named AR-KGAT [74], uses a neural attention mechanism, enhanced through association rules, to learn KG embeddings. The model is evaluated with the WN18RR and FB15k-237 datasets on the task of triple classification. The model's accuracy is reported in the respective dataset tables, the other metrics are not provided in the original paper.

Across both datasets, the `KG-NLM` approach demonstrates relatively high performance, with over 80% on all metrics. While `KG-RoBERTa` and `KG-DistilBERT`, trained for 3 epochs on FB15k-237, outperform the baseline model, they fail to surpass the baseline on the WN18RR dataset. Notably, `KG-DistilBERT` outperforms the other two models, achieving over 90% accuracy and f1-score on both datasets. Furthermore, all models achieve higher performance when trained for 3 epochs compared to 1 epoch, on both datasets. `KG-BLOOM` shows the lowest performance on both datasets, however, it is noteworthy that the employed pretrained version of BLOOM has the lowest number of parameters among the available pretrained BLOOM models.

| **Model** | | **Epochs** | | | | **Metric** |
|---|---|---|---|---|---|---|
| | | | Accuracy | F1-Score | Precision | Recall |
| Baseline | AR-KGAT [74] | - | 0.973 | - | - | - |
| Ours | KG-RoBERTa | 1 | 0.877 | 0.880 | 0.859 | 0.903 |
| | | 3 | 0.904 | 0.905 | 0.894 | 0.917 |
| | KG-DistilBERT | 1 | 0.864 | 0.866 | 0.855 | 0.876 |
| | | 3 | 0.917 | 0.919 | 0.900 | 0.939 |
| | KG-BLOOM | 1 | 0.781 | 0.792 | 0.754 | 0.835 |
| | | 3 | 0.868 | 0.867 | 0.870 | 0.865 |

Table 7: WN18RR - Evaluation results of the `KG-NLM` approach on triple classification

Figure 15 illustrates the performance of the `KG-NLM` models during the training phase on FB15k-237, in terms of training loss and accuracy (see figure 16 in the appendix for further metrics). Instead of showing the number of training epochs, the x-axis represents the total number of samples divided by the gradient accumulation step for both graphs. In the first

| **Model** | | **Epochs** | | | | **Metric** |
|---|---|---|---|---|---|---|
| | | | Accuracy | F1-Score | Precision | Recall |
| Baseline | AR-KGAT [74] | - | 0.925 | - | - | - |
| Ours | KG-RoBERTa | 1 | 0.934 | 0.938 | 0.890 | 0.991 |
| | | 3 | **0.945** | 0.947 | 0.910 | 0.986 |
| | KG-DistilBERT | 1 | 0.934 | 0.938 | 0.889 | 0.993 |
| | | 3 | **0.946** | 0.948 | 0.911 | 0.988 |
| | KG-BLOOM | 1 | 0.868 | 0.87 | 0.865 | 0.867 |
| | | 3 | 0.924 | 0.928 | 0.883 | 0.977 |

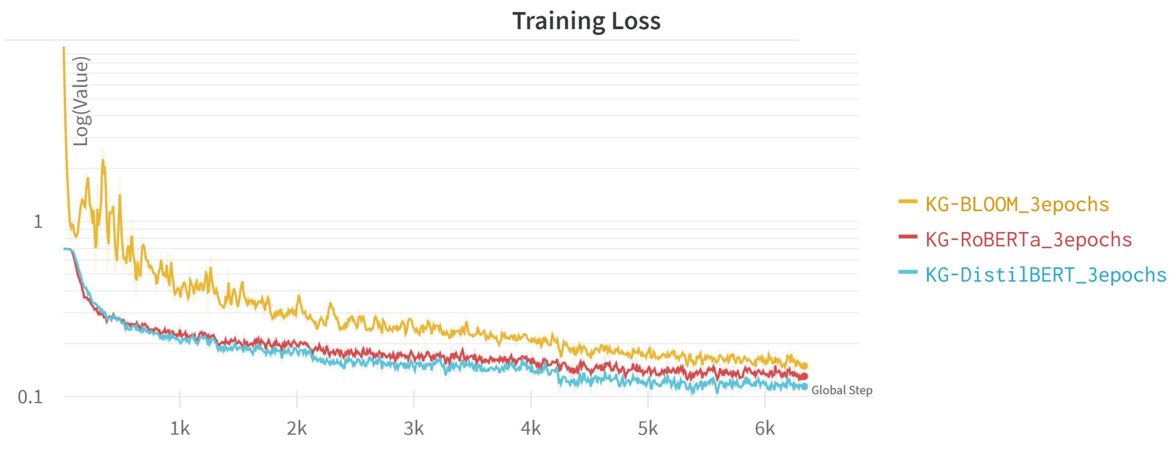
Table 8: FB15k-237 - Evaluation results of the `KG-NLM` approach on triple classification

graph, which displays the training loss curves (in logarithmic scale) for the three models over 3 training epochs, it is evident that `KG-RoBERTa` and `KG-DistilBERT` converge at similar rates. Initially, `KG-RoBERTa` exhibits a lower loss during the early training steps, but as training progresses, `KG-DistilBERT` surpasses `KG-RoBERTa`, achieving a lower loss. The second graph showcases the accuracy curves of the three models on the FB15k-237 dataset, also considering training for 1 epoch. Similar to the loss function, the accuracy curves mirror the behaviour observed in the first graph.

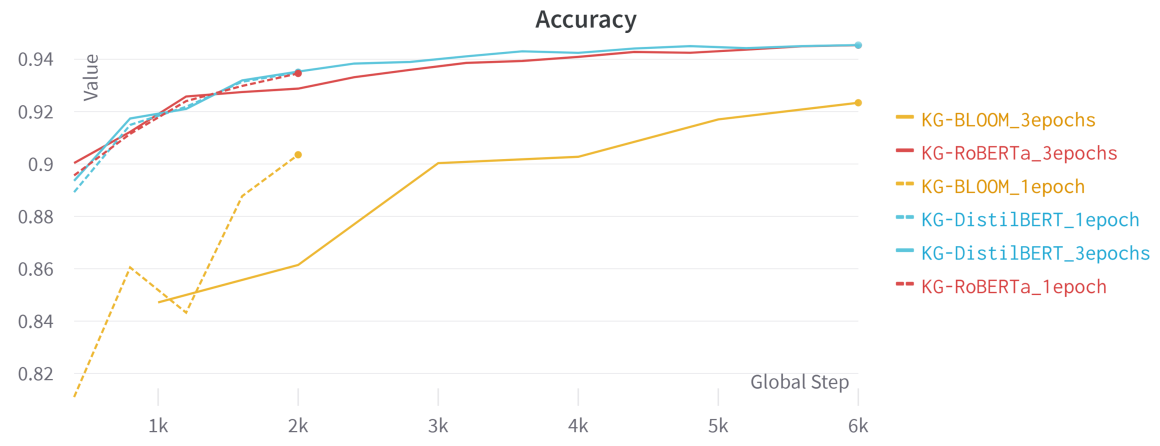
Both DistilBERT and RoBERTa have already proven to achieve very high performance on general text classification tasks. In particular, Qasim et al. [75] report accuracy and precision values of over 96% achieved by fine-tuning DistilBERT and RoBERTa on binary text classification task with two datasets: fake-news detection and hate speech. The task of triple classification, explored in this thesis, could be understood as a binary text classification task. Specifically, an NLM input unit is formed by joining the KG's head entity, relation, and tail entity into one sequence to resemble a sentence. Each sequence is accompanied by a classification label, indicating whether the sequence is part of the KG (true) or not (false).

One possible reason for the strong performance of `KG-DistilBERT` and `KG-RoBERTa` on the FB15k-237 dataset is their pretraining on the English Wikipedia data, from which the FB15k-237 dataset is also derived (as discussed in chapter 2.1). Consequently, the BERT-based models have already acquired some knowledge about the triples during their pretraining phase and can leverage this knowledge for the current task. In a study conducted by Lv et al. [76], they investigate the influence of prior knowledge in BERT on its performance in a subset of FB15k-237. For each triple of the validation and test sets, they count the number of sentences in Wikipedia that involve both head and tail entities and assume that BERT has been trained on these sentences as often as they occur in Wikipedia. Lv et al. report that BERT's performance improves as the number of sentences per triple increases, while the performance of TuckER and ConvE remains relatively constant. This suggests that BERT benefits from its prior knowledge when the entities in the triples are mentioned more frequently in the training data.

The relatively lower performance on the WN18RR dataset can be attributed to its specific characteristics. In comparison to FB15k-237, WN18RR contains a significantly smaller number of relations and less than half the training samples (refer to table 2 in chapter 5.1). Additionally, the relations in WN18RR are shorter, comprising a maximum of three words, whereas FB15k-237 includes relations with a more extensive set of words, conveying more contextual meaning that the models can leverage for better performance.



(a)



(b)

Figure 15: Performance of the `KG-NLM` models on the FB15k-237 dataset

# 6 Conclusion

## Summary

This thesis aims to explore the potential of using pretrained NLMs for KGC tasks. Two distinct approaches are investigated: (1) using pretrained NLM embeddings of entities and relations as initialization vectors for KGE models, and (2) fine-tuning a pretrained NLM with KG triples for triple classification.

In line with the first approach and targeting the first two research questions, a series of experiments utilizing four KGE models is conducted. The experimental results indicate that, on the WN18RR dataset, TransE surpasses the baseline models in terms of Hits@1, Hits@10, and MRR. Similarly, TuckER exceeds the baseline on the same dataset for Hits@1, Hits@3, and MRR. However, the remaining models do not exhibit any performance improvement when initialized with pretrained NLM-generated embeddings. Among the investigated BERT embedding strategies, which include `BERT (labels)`, `BERT PCA (labels)`, and `BERT (labels & descriptions)`, most models demonstrate superior performance when initialized with `BERT PCA (labels)` embeddings. The performance of the models initialized with `BERT (labels)` and `BERT (labels & descriptions)` is largely similar. Furthermore, except for RotatE, no significant variation in model performance is observed across both datasets when utilizing either context-aware (BERT) or context-independent (Word2Vec) embeddings.

Addressing the third research question, experiments based on the `KG-NLM` approach are carried out for the following NLM-based models: `KG-RoBERTa`, `KG-DistilBERT`, and `KG-BLOOM`. These models exhibit significant performance on both datasets, achieving over 80% across all metrics. `KG-RoBERTa` and `KG-DistilBERT`, trained for three epochs on the FB15k-237 dataset, exceed the baseline model, however falling short in outperforming the baseline on the WN18RR dataset. Overall, `KG-DistilBERT` outperforms the other two models, achieving over 90% accuracy and f1-score on both datasets. Conversely, `KG-BLOOM` records the lowest performance on both datasets.

This thesis presents several contributions to current research. Firstly, it utilizes both context-independent and context-aware NLM-generated embeddings to initialize established KGE models. Secondly, it extends the exploration to include two KGE models, TuckER and ConvE, that have not been previously assessed in this context. Lastly, it introduces three pretrained NLM-based models for KGC: `KG-RoBERTa`, `KG-DistilBERT`, and `KG-BLOOM`. These are evaluated on the task of triple classification, on two benchmark datasets: WN18RR and FB15k-237.

## Limitations and Future Work

This thesis, while enhancing the understanding of utilizing pretrained NLMs for KGC, acknowledges certain limitations. The number of epochs used to train the KGE models with pretrained NLM-generated embeddings is confined by computational resources. Training for a larger amount of epochs could elevate the models' performance. Additionally, a comprehensive hyperparameter optimization, tailored to the specific pretrained embeddings and datasets, could yield superior performance. Moreover, it would be beneficial to conduct more in-depth research to understand why the knowledge transfer from pretrained NLMs to KGE methods explored in this thesis is less effective than anticipated.

The proposed `KG-NLM` approach considers solely entity and relationship labels, neglecting potentially valuable information from textual descriptions. Integrating this information could provide a richer context and enhance model performance, presenting a noteworthy direction for future exploration. While the `KG-NLM` models employed in this thesis are evaluated on triple classification, extending their evaluation to the task of link prediction could provide a comprehensive understanding of their utility in KGC tasks. Ultimately, the applicability of the proposed method for real-world use cases, such as corporate KGs featuring knowledge about more specific domains, is yet to be investigated. Further research should examine whether the `KG-NLM` approach retains its effectiveness when applied to these more specialized KGs.

# A Appendix

## A.1 RotatE Relation Initialization

The main novelty about the RotatE model lies in the fact that relations are modelled as a rotation from the head entity the to tail entity in the complex vector space [46]. For each relation, its complex vector representation is calculated using the phase of a complex number, which refers to the angle it makes with the positive real axis in the complex plane. It represents the direction of the complex number from the origin (0) to its location in the complex plane. In RotatE, the phase of the complex relation vector is calculated as shown in equation A.1, relying on the real part $re\_v$ of a given complex vector $v$:

$$phase = 2 * \pi * re\_v \tag{A.1}$$

As a result, the real part of the complex relation vector is determined by taking the *cosine* of the *phase*, while the imaginary part is obtained by taking the *sine* of its *phase*. In the standard initialization process of RotatE, the complex vector $v$, and thus $re\_v$, is randomly generated. In the case of initialization with pretrained embeddings, the same logic as above is followed treating the pretrained embeddings as the vector $v$.

## A.2 Evaluation

The hyperparameter configurations described in chapter 5.2 are provided in detail in table 9 (`benchP`), table 10 (`hpoP`) and table 11 (`paperP`).

| **Hyperparameter** | **TransE** | |
|---|---|---|
| | WN18RR | FB15k-237 |
| scoring norm | 1 | 1 |
| optimizer | adam | adam |
| learning rate | 0.0011049153751436596 | 0.002256570208126583 |
| weight decay | 0.0 | 0.0 |
| loss function | softplus | crossentropy |
| training approach | lcwa | lcwa |
| training batch size | 512 | 128 |
| label smoothing | 0.00200051768009458 | 0.7644642800393661 |
| evaluator | rankbased | rankbased |
| filtered | true | true |

Table 9: Hyperparameter selection `benchP` sourced from [72].

| **Hyperparameter** | **ConvE** | **TransE** |
|---|---|---|
| | WN18RR | FB15k-237 |
| scoring function | - | 1 |
| optimizer | adam | adam |
| learning rate | 0.009330514459787512 | 0.00091246982681624 |
| loss function | bcewithlogits | bcewithlogits |
| training approach | lcwa | lcwa |
| training batch size | 128 | 256 |
| label smoothing | 0.004869831870660787 | 0.006091616913055568 |
| evaluator | rankbased | rankbased |
| filtered | true | true |
| feature map dropout | 0.5 | - |
| input dropout | 0.4 | - |
| kernel height | 3 | - |
| kernel width | 3 | - |

Table 10: Hyperparameter selection `hpoP`, acquired through hyperparameter optimization in the scope of this thesis.

| **Hyperparameter** | **ConvE** | **TuckerE** | | **RotatE** |
|---|---|---|---|---|
| | both datasets | WN18RR | FB15k-237 | both datasets |
| optimizer | adam | adam | adam | adam |
| learning rate | 0.003 | 0.01 | 0.0005 | 0.00005 |
| learning rate decay | 0.995 | 1.0 | 1.0 | - |
| weight decay | 0.0 | - | - | 0.0 |
| loss function | bcewithlogits | bcewithlogits | bcewithlogits | NSSALoss |
| training approach | lcwa | lcwa | lcwa | lcwa |
| training batch size | 128 | 128 | 128 | - |
| label smoothing | 0.1 | 0.1 | 0.1 | - |
| evaluator | rankbased | rankbased | rankbased | rankbased |
| filtered | true | true | true | true |
| feature map dropout | 0.2 | - | - | - |
| input dropout | 0.2 | - | - | - |
| hidden dropout | 0.3 | - | - | - |
| dropout 0 | - | 0.2 | 0.3 | - |
| dropout 1 | - | 0.2 | 0.4 | - |
| dropout 2 | - | 0.3 | 0.5 | - |
| margin | - | - | - | 9 |
| adversarial temperature | - | - | - | 1.0 |

Table 11: Hyperparameter selection `paperP`, sourced from original papers ConvE [13], TuckER [14] and RotatE [46].

The remaining experimental results from the first approach are provided in table 12.

| **Model** | **Initialization** | **Hyper** | **Nr.** | **Metric** | | | |
|---|---|---|---|---|---|---|---|
| | **Embeddings** | **-parameters** | **epochs** | Hits@1 | Hits@3 | Hits@10 | MRR |
| **Dataset: FB15k-237** | | | | | | | |
| TransE | random | benchP | 500 | 0.141 | 0.216 | 0.309 | 0.198 |
| | BERT (labels) | benchP | 500 | 0.099 | 0.153 | 0.209 | 0.138 |
| | BERT (labels & descriptions) | benchP | 325* | 0.114 | 0.162 | 0.217 | 0.149 |
| **Dataset: WN18RR** | | | | | | | |
| ConvE | random | paperP | 1000 | 0.333 | 0.448 | 0.507 | 0.399 |
| | Word2Vec | paperP | 500 | 0.034 | 0.097 | 0.149 | 0.075 |
| | BERT PCA (labels) | paperP | 500 | 0.1 | 0.172 | 0.247 | 0.149 |

Table 12: Additional evaluation results of TransE and ConvE. The number of epochs marked with * signifies early stopping.

Figure 16 illustrates the precision, recall and f1-score curves during training phase for the three `KG-NLM` models on the FB15k-237 dataset.

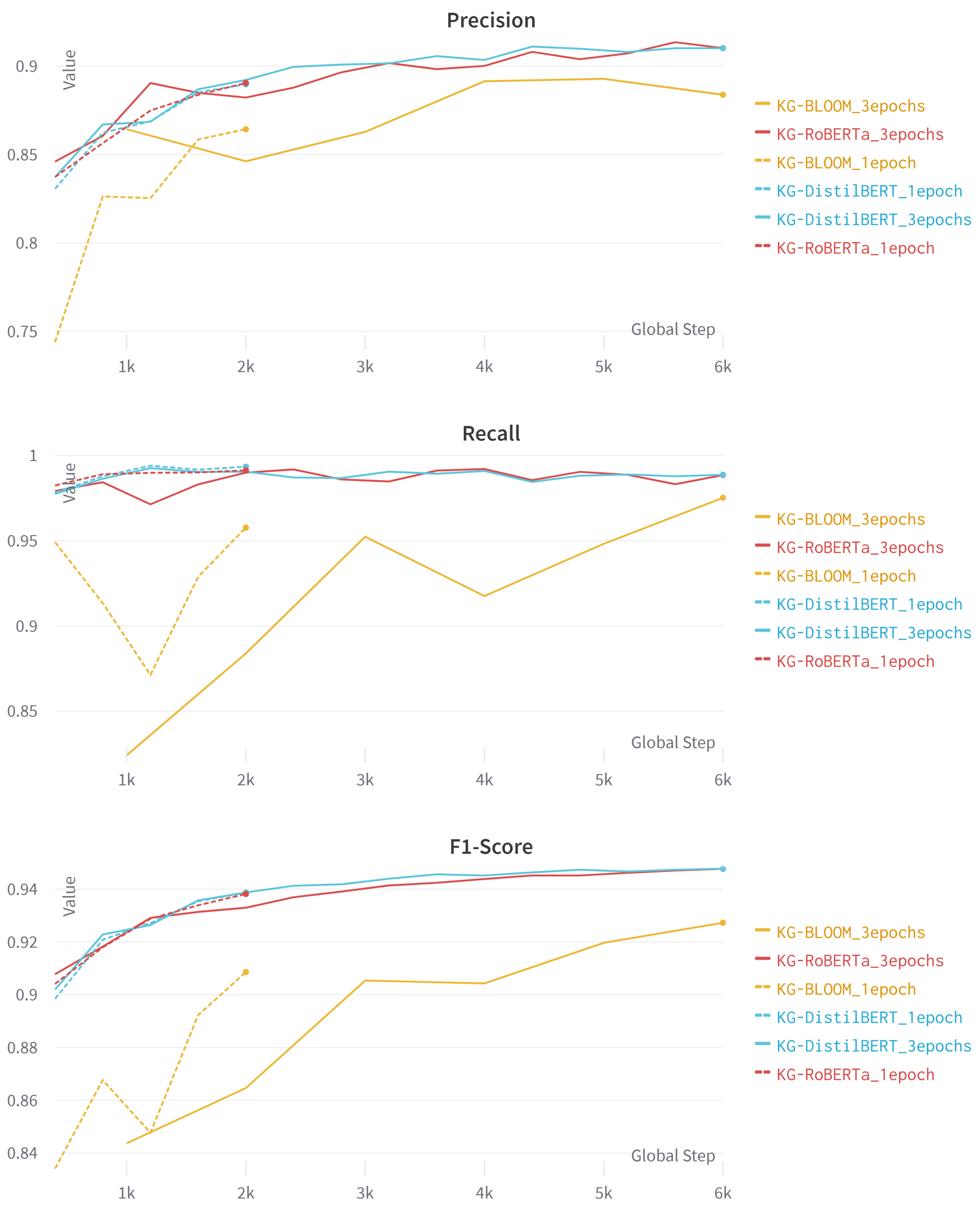


Figure 16: Visualization of precision, recall and f1-score of the `KG-NLM` models on FB15k-237

## References

[1] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743, 2017.

[2] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S. Yu. A survey on knowledge graphs: Representation, acquisition and applications. *CoRR*, abs/2002.00388, 2020.

[3] Kurt D. Bollacker, Patrick Tufts, Tom Pierce, and Robert Cook. A platform for scalable, collaborative, structured information integration. 2007.

[4] 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.

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

[6] Xin Luna Dong. Challenges and innovations in building a product knowledge graph. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, page 2869, New York, NY, USA, 2018. Association for Computing Machinery.

[7] Thomas Hubauer, Steffen Lamparter, Peter Haase, and Daniel M. Herzig. Use cases of the industrial knowledge graph at siemens. In *International Workshop on the Semantic Web*, 2018.

[8] Dongzhuoran Zhou, Baifan Zhou, Zhuoxun Zheng, Egor V. Kostylev, Gong Cheng, Ernesto Jiménez-Ruiz, Ahmet Soylu, and Evgeny Kharlamov. Enhancing knowledge graph generation with ontology reshaping – bosch case. In Paul Groth, Anisa Rula, Jodi Schneider, Ilaria Tiddi, Elena Simperl, Panos Alexopoulos, Rinke Hoekstra, Mehwish Alam, Anastasia Dimou, and Minna Tamper, editors, *The Semantic Web: ESWC 2022 Satellite Events*, pages 299–302, Cham, 2022. Springer International Publishing.

[9] Irlán Grangel-González, Felix Lösch, and Anees ul Mehdi. Knowledge graphs for efficient integration and access of manufacturing data. In *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 1, pages 93–100, 2020.

[10] 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(2):48–75, apr 2019.

[11] Zhe Chen, Yuehan Wang, Bin Zhao, Jing Cheng, Xin Zhao, and Zongtao Duan. Knowledge graph completion: A review. *IEEE Access*, 8:192435–192456, 2020.

[12] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[13] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. *CoRR*, abs/1707.01476, 2017.

[14] Ivana Balazevic, Carl Allen, and Timothy Hospedales. TuckER: Tensor factorization for knowledge graph completion. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5185–5194, Hong Kong, China, November 2019. Association for Computational Linguistics.

[15] Yankai Lin, Zhiyuan Liu, and Maosong Sun. Modeling relation paths for representation learning of knowledge bases. 06 2015.

[16] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. Recent trends in deep learning based natural language processing [review article]. *IEEE Computational Intelligence Magazine*, 13(3):55–75, 2018.

[17] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[18] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[19] Fabio Petroni, Tim Rocktäschel, Patrick S. H. Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. Language models as knowledge bases? *CoRR*, abs/1909.01066, 2019.

[20] Zhiyuan Zhang, Xiaoqian Liu, Yi Zhang, Qi Su, Xu Sun, and Bin He. Pretrain-kge: Learning knowledge representation from pretrained language models. In *Findings*, 2020.

[21] Liang Yao, Chengsheng Mao, and Yuan Luo. KG-BERT: BERT for knowledge graph completion. *CoRR*, abs/1909.03193, 2019.

[22] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019.

[23] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108, 2019.

[24] Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilic, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, Jonathan Tow, Alexander M. Rush, Stella Biderman, Albert Webson, Pawan Sasanka Ammanamanchi, Thomas Wang, Benoît Sagot, Niklas Muennighoff, Albert Villanova del Moral, Olatunji Ruwase, Rachel Bawden, Stas Bekman, Angelina McMillan-Major, Iz Beltagy, Huu Nguyen, Lucile Saulnier, Samson Tan, Pedro Ortiz Suarez, Victor Sanh, Hugo Laurençon, Yacine Jernite, Julien Launay, Margaret Mitchell, Colin Raffel, Aaron Gokaslan, Adi Simhi, Aitor Soroa, Alham Fikri Aji, Amit Alfassy, Anna Rogers, Ariel Kreisberg Nitzav, Canwen Xu, Chenghao Mou, Chris Emezue, Christopher Klamm, Colin Leong, Daniel van Strien, David Ifeoluwa Adelani, and et al. BLOOM: A 176b-parameter open-access multilingual language model. *CoRR*, abs/2211.05100, 2022.

[25] Gerhard Lakemeyer and Bernhard Nebel. Foundations of knowledge representation and reasoning. volume 810, pages 1–12, 01 1992.

[26] John Mylopoulos. An overview of knowledge representation. In *Proceedings of the 1980 Workshop on Data Abstraction, Databases and Conceptual Modeling*, page 5–12, New York, NY, USA, 1980. Association for Computing Machinery.

[27] R. H. Richens. Preprogramming for mechanical translation. *Mech. Transl. Comput. Linguistics*, 3:20–25, 1956.

[28] Fritz Lehmann. Semantic networks. *Computers Mathematics with Applications*, 23(2):1–50, 1992.

[29] TIM BERNERS-LEE, JAMES HENDLER, and ORA LASSILA. The semantic web. *Scientific American*, 284(5):34–43, 2001.

[30] Dan Brickley, Ramanathan V Guha, and Andrew Layman. Resource description framework (rdf) schema specification. Technical report, Technical report, W3C, 1999. W3C Proposed Recommendation. http://www. w3 . . . , 1998.

[31] Graham Klyne and Jeremy J. Carroll. Resource description framework (rdf): Concepts and abstract syntax. W3C Recommendation, 2004.

[32] Martin Dürst and M. Suignard. Internationalized resource identifiers (iris). 01 2005.

[33] Michael Färber, Frederic Bartscherer, Carsten Menne, and Achim Rettinger. Linked data quality of dbpedia, freebase, opencyc, wikidata, and yago. *Semantic Web*, 9:77–129, 2017.

[34] Lisa Ehrlinger and Wolfram Wöß. Towards a definition of knowledge graphs. In *International Conference on Semantic Systems*, 2016.

[35] Liang Wang, Wei Zhao, Zhuoyu Wei, and Jingming Liu. SimKGC: Simple contrastive knowledge graph completion with pre-trained language models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4281–4294, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[36] George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J. Miller. Introduction to WordNet: An On-line Lexical Database*. *International Journal of Lexicography*, 3(4):235–244, 12 1990.

[37] Christiane Fellbaum. *A Semantic Network of English: The Mother of All WordNets*, page 137–148. Kluwer Academic Publishers, USA, 1998.

[38] Ruslan Mitkov. *The Oxford Handbook of Computational Linguistics (Oxford Handbooks)*. Oxford University Press, Inc., USA, 2005.

[39] 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*, SIGMOD '08, page 1247–1250, New York, NY, USA, 2008. Association for Computing Machinery.

[40] Thomas Pellissier Tanon, Denny Vrandečić, Sebastian Schaffert, Thomas Steiner, and Lydia Pintscher. From freebase to wikidata: The great migration. WWW '16, page 1419–1428, Republic and Canton of Geneva, CHE, 2016. International World Wide Web Conferences Steering Committee.

[41] Visualization of subgraph of freebase. `https://colab.research.google.com/drive/1Fcf8vkuaO6VCOB3MAZlpDebCAgyUnMBj?usp=sharing#scrollTo=cHzhvBhbegPX`. Accessed: 2023-07-13.

[42] Zhiqing Sun, Shikhar Vashishth, Soumya Sanyal, Partha Talukdar, and Yiming Yang. A re-evaluation of knowledge graph completion methods, 2020.

[43] Chenchen Li, Aiping Li, Ye Wang, Hongkui Tu, and Yichen Song. A survey on approaches and applications of knowledge representation learning. In *2020 IEEE Fifth International Conference on Data Science in Cyberspace (DSC)*, pages 312–319, 2020.

[44] 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, jan 2016.

[45] Ilaria Ferrari, Giacomo Frisoni, Paolo Italiani, Gianluca Moro, and Claudio Sartori. Comprehensive analysis of knowledge graph embedding techniques benchmarked on link prediction. *Electronics*, 11:3866, 11 2022.

[46] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. *CoRR*, abs/1902.10197, 2019.

[47] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 809–816, Madison, WI, USA, 2011. Omnipress.

[48] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and li Deng. Embedding entities and relations for learning and inference in knowledge bases. 12 2014.

[49] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. *CoRR*, abs/1606.06357, 2016.

[50] L. R. Tucker. The extension of factor analysis to three-dimensional matrices. In H. Gulliksen and N. Frederiksen, editors, *Contributions to mathematical psychology.*, pages 110–127. Holt, Rinehart and Winston, New York, 1964.

[51] Vineet Bhatt, Sunil Kumar, and Seema Saini. Tucker decomposition and applications. *Materials Today: Proceedings*, 46:10787–10792, 2021. International Conference on Technological Advancements in Materials Science and Manufacturing.

[52] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In Aldo Gangemi, Roberto Navigli, Maria-Esther Vidal, Pascal Hitzler, Raphaël Troncy, Laura Hollink, Anna Tordai, and Mehwish Alam, editors, *The Semantic Web*, pages 593–607, Cham, 2018. Springer International Publishing.

[53] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha P. Talukdar. Composition-based multi-relational graph convolutional networks. *CoRR*, abs/1911.03082, 2019.

[54] Zhanqiu Zhang, Jie Wang, Jieping Ye, and Feng Wu. Rethinking graph convolutional networks in knowledge graph completion. In *Proceedings of the ACM Web Conference 2022*, WWW '22, page 798–807, New York, NY, USA, 2022. Association for Computing Machinery.

[55] K. R. Chowdhary. *Natural Language Processing*, pages 603–649. Springer India, New Delhi, 2020.

[56] Felipe Almeida and Geraldo Xexéo. Word embeddings: A survey. *CoRR*, abs/1901.09069, 2019.

[57] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3(null):1137–1155, mar 2003.

[58] Fu-Lian Yin, Xing-Yi Pan, Xiao-Wei Liu, and Hui-Xin Liu. Deep neural network language model research and application overview. In *2015 12th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, pages 55–60, 2015.

[59] Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. Pre-trained models for natural language processing: A survey. *CoRR*, abs/2003.08271, 2020.

[60] Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Liang Zhang, Wentao Han, Minlie Huang, Qin Jin, Yanyan Lan, Yang Liu, Zhiyuan Liu, Zhiwu Lu, Xipeng Qiu, Ruihua Song, Jie Tang, Ji-Rong Wen, Jinhui Yuan, Wayne Xin Zhao, and Jun Zhu. Pre-trained models: Past, present and future. *CoRR*, abs/2106.07139, 2021.

[61] Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In Yoshua Bengio and Yann LeCun, editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013.

[62] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[63] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff

Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016.

[64] The bigscience roots corpus: A 1.6tb composite multilingual dataset, 2023.

[65] David Powers and Ailab. Evaluation: From precision, recall and f-measure to roc, informedness, markedness correlation. *J. Mach. Learn. Technol*, 2:2229–3981, 01 2011.

[66] Lianbo Ma, Peng Sun, Zhiwei Lin, and Hui Wang. Composing knowledge graph embeddings via word embeddings. *CoRR*, abs/1909.03794, 2019.

[67] Mirza Mohtashim Alam, Md Rashad Al Hasan Rony, Mojtaba Nayyeri, Karishma Mohiuddin, M. S. T. Mahfuja Akter, Sahar Vahdati, and Jens Lehmann. Language model guided knowledge graph embeddings. *IEEE Access*, 10:76008–76020, 2022.

[68] Russa Biswas, Radina Sofronova, Mehwish Alam, and Harald Sack. Contextual language models for knowledge graph completion. In *MLSMKG@PKDD/ECML*, 2021.

[69] Kristina Toutanova and Danqi Chen. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pages 57–66, Beijing, China, July 2015. Association for Computational Linguistics.

[70] Mehdi Ali, Max Berrendorf, Charles Tapley Hoyt, Laurent Vermue, Sahand Sharifzadeh, Volker Tresp, and Jens Lehmann. Pykeen 1.0: A python library for training and evaluating knowledge graph embeddings. *CoRR*, abs/2007.14175, 2020.

[71] Ian T. Jolliffe and Jorge Cadima. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150202, 2016.

[72] Mehdi Ali, Max Berrendorf, Charles Tapley Hoyt, Laurent Vermue, Mikhail Galkin, Sahand Sharifzadeh, Asja Fischer, Volker Tresp, and Jens Lehmann. Bringing light into the dark: A large-scale evaluation of knowledge graph embedding models under a unified framework. *CoRR*, abs/2006.13365, 2020.

[73] Deepak Nathani, Jatin Chauhan, Charu Sharma, and Manohar Kaul. Learning attention-based embeddings for relation prediction in knowledge graphs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4710–4723, Florence, Italy, July 2019. Association for Computational Linguistics.

[74] Zhenghao Zhang, Jianbin Huang, and Qinglin Tan. Association rules enhanced knowledge graph attention network. *Knowledge-Based Systems*, 239:108038, 2022.

[75] Rukhma Qasim, Waqas Bangyal, Mohammed Alqarni, and Abdulwahab Almazroi. A fine-tuned bert-based transfer learning approach for text classification. *Journal of Healthcare Engineering*, 2022:1–17, 01 2022.

[76] Xin Lv, Yankai Lin, Yixin Cao, Lei Hou, Juanzi Li, Zhiyuan Liu, Peng Li, and Jie Zhou. Do pre-trained models benefit knowledge graph completion? a reliable evaluation and a reasonable approach. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3570–3581, Dublin, Ireland, May 2022. Association for Computational Linguistics.

# Erklärung

*Ich versichere wahrheitsgemäß, die Arbeit selbstständig verfasst, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde sowie die Satzung des KIT zur Sicherung guter wissenschaftlicher Praxis in der jeweils gültigen Fassung beachtet zu haben.*

Karlsruhe, 04.08.2023

Vjola Cili

[illegible]