Abstract

Knowledge Graph Embedding models have become an important area of machine learning. Those models provide a latent representation of entities and relations in a knowledge graph which can then be used in downstream machine learning tasks such as link prediction. The learning process of such models can be performed by contrasting positive and negative triples. While all triples of a KG are considered positive, negative triples are usually not readily available. Therefore, the choice of the sampling method to obtain the negative triples plays a crucial role in the performance and effectiveness of Knowledge Graph Embedding models. Most of the current methods fetch negative samples from a random distribution of entities in the underlying Knowledge Graph which also often includes meaningless triples. Other known methods use adversarial techniques or generative neural networks which consequently reduce the efficiency of the process. In this paper, we propose an approach for generating informative negative samples considering available complementary knowledge about entities. Particularly, Pre-trained Language Models are used to form neighborhood clusters by utilizing the distances between entities to obtain representations of symbolic entities via their textual information. Our comprehensive evaluations demonstrate the effectiveness of the proposed approach on benchmark Knowledge Graphs with textual information for the link prediction task.

1 Introduction

Knowledge graphs (KG) are a data model to represent the facts from the real world in the form of triple \((h, r, t)\) where \(h\) and \(t\) refer to head and tail entities, and \(r\) denotes the relations between them. Despite the size of some KGs reaching billions of triples, they usually remain incomplete. This is due to the difficulties in capturing all relevant facts about a domain of interest at the time of KG construction. Knowledge Graph Embedding (KGE) models are the most prominent approach to complete KGs. The learning process of KGEs involves contrasting positive and negative triples which plays an important role for the effectiveness of the models as well as the ultimate results on downstream tasks. Despite the importance of the methods for sampling negative triples from the existing positive triples in a KG, only recently this has gained attention and some techniques have been proposed to increase their efficiency (Mikolov et al., 2013; Sun et al., 2019; Cai and Wang, 2018; Wang et al., 2018; Dash and Gliozzo, 2019; Alam et al., 2020). However, still most of these methods only fetch the negative samples (NS) through a random distribution including uniform distribution (He et al., 2017) or population-based distribution (Chen et al., 2017) from the entities in the underlying KG. The drawback of these approaches is that the negative samples also include a high percentage of meaningless triples, i.e., the classification problem of distinguishing positive and negative triples becomes (too) simple.

In other works, Generative Adversarial networks (GANs) (Goodfellow et al., 2014) are used for generating negative samples for KGEs. Most of these works are computationally expensive as they either suffer from the vanishing gradient problem or require a high number of training parameters. Furthermore, these approaches focus on learning the graph patterns rather than considering any comple-
mentary knowledge that is carried by entities and relations. Generating such negative samples can reduce the performance of KGE models and affect their link prediction capability. Especially in large KGs, it is very likely that most negative samples are sampled from a very large distribution.

To alleviate the aforementioned issues, we propose LEMON, a novel method for negative sample generation in KGE models using Pre-trained Language Models (PLMs). The core idea is that the vector representations of contextual knowledge for each entity are obtained from PLMs and influence the similarity of entities. The choice of PLMs for this work is Sentence-BERT (Reimers and Gurevych, 2019) as pre-trained language model to obtain the contextual representation of KG entities. Sentence-BERT is chosen due to its ability to derive semantically meaningful sentences with high computational efficiency. Furthermore, a dimensionality reduction algorithm is used together with the K-means++ clustering technique in order to measure the similarity of entities in a run-time efficient manner. The embeddings obtained from the language models convey contextual meaning as such models are already trained on large corpora. Specifically, for an entity (either as the head or tail of a triple) to be corrupted, the model considers the corresponding textual information and identifies an ideal set of clusters to which the target entity belongs. When considering the nearest clusters, number of hops are the set of clusters (containing entities) in the desired ranges. From these clusters the negative candidate entities are sampled.

Figure 1 demonstrates the difference between LEMON and a random negative sampling approach. In order to corrupt a particular tail entity (in this case Vitamin), the distance between the candidate entity and the clusters (containing entities) is crucial. The upper part of the figure demonstrates the neighborhood between the clusters based on the various distances \( d \in \{9.36, 12.15, 13.24\} \) from the candidate entity Vitamin. Each value of \( d \) represents the distance between candidate entity (which is to be corrupted) and K-th cluster. The model chooses contextually meaningful entities from the clusters using the desired distances. The lower part of the figure illustrates the sampling method of random corruption and the possibility of sampling meaningful entities.

The key contributions of this paper can be summarized as follows:

- A novel approach named LEMON is proposed that employs pre-trained language models and clustering methods to obtain meaningful entities to sample negative triples. To the best of our knowledge, we are the first to employ and investigate pre-trained language models for the purpose of negative sampling approach in KGEs.

- A comprehensive evaluation across standard publicly available benchmarks namely WN18, and WN18RR that have sufficient textual information have been performed. The evaluation contains several widely used KGE models including TransE (Bordes et al., 2013), RotatE (Sun et al., 2019), DistMult (Yang et al., 2015).

2 Related Work

Here, we provide a summary of relevant approaches which we classified as following:

Distribution-based Approaches. Many of the initial negative sampling approaches follow random distribution for selection of entities to be corrupted (Mikolov et al., 2013; Sun et al., 2019). Besides uniform (He et al., 2017) and population-based distributions (Chen et al., 2017), Bernoulli distribution is considered to limit the appearance of false negative triples in the existing relations (Wang et al., 2014). Despite their simplicity, such approaches suffer from the vanishing gradient problem as described in (Cai and Wang, 2018; Wang et al., 2018), and the provided negative samples are not informative.

Generative Adversarial Network-based Approaches. Recently, Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) have been explored for negative sampling to overcome the limitations of fixed distribution based sampling (Cai and Wang, 2018; Wang et al., 2018). The discriminator is trained to minimize the margin-based ranking loss, while the generator learns to sample high-quality negative samples (Cai and Wang, 2018). Although such techniques are capable of generating high-quality negative samples, they are expensive to train as addressed in (Zhang et al., 2019). It uses a cashing mechanism which solved the time-efficiency issues, but does not improve the effectiveness of the generated NS.

Structure-aware Approaches. More recently, a structure-aware negative sampling technique (SANS) was proposed (Ahrabian et al., 2020) to
consider the neighbourhood information for generating negative triples. To corrupt an entity of a triple for negative sampling, all the k-hop neighborhood nodes connected to the entity are considered negative. Thus, the negative triples are generated from the structural information (i.e., the k-hop neighborhood). However, it is crucial for the KGE models to know about possible connections beyond k-hop to perform link prediction.

**Other Approaches.** Data augmentation methods are recently used in negative sampling (Huang et al., 2021). Graph neural network-based positive mix and hop-mix strategies have been proposed to generate negative samples for a set of neighborhood hops. Following a similar research line, MixKG (Che et al., 2022) introduced an approach for generating hard negative samples with a similarity based negative triples. These methods are not structure-based nor PLM-based. However, the performance of such methods decreases with the increased size of hard negative samples which limits the scaling capability of the model.

### 3 Approach: LEMON

In this section, we describe the details about LEMON on how it leverages a pre-trained language model and neighborhood clustering algorithm to generate negative samples for training KGE models. As the initial step, the embedding of the entity label (text) are obtained from Sentence-BERT (Reimers and Gurevych, 2019). Furthermore, to cluster similar entities together for building a neighborhood, the cluster size needs to be defined which is considered as one of the key hyperparameters for obtaining the clusters. Moreover, Principal Component Analysis (PCA) (Abdi and Williams, 2010) dimensionality reduction techniques is employed to adjust the dimension of the PLM embeddings. This is done to eradicate the curse of dimensionality (Verleysen and François, 2005), which is often caused by large embedding dimensions (in our case Sentence-BERT embeddings). Finally, the K-means++ (Arthur and Vassilvitskii, 2006) algorithm is leveraged to build the neighborhood clusters. Our proposed method consists of two phases: Building the neighborhood clusters, and negative sample generation. The phases are described in details below.

#### 3.1 Building the Neighborhood Clusters

Building the neighborhood is considered a crucial step, since it puts the KG entities in different clusters based on the pre-trained embeddings of the text. Figure 2 illustrates our proposed neighborhood clustering step.

The input for Sentence-BERT is the textual representation of the set \( E_{text} \) which is the set of corresponding textual representation of the entity set \( E \). Let us consider Sentence-BERT as a function \( \Omega \). The output of \( \Omega \) is a \( |E| \times 786 \) dimensional matrix representation. Formally, this can be defined as:

\[
\Omega : E_{text} \rightarrow E_{ST} \in \mathbb{R}^{|E| \times 786}
\]

The ”curse of dimensionality” refers to the fact that large dimensional embeddings can be harmful to learning systems (Verleysen and François, 2005). Because PLM embeddings are large, we suspect that, it may not be a suitable fit for clustering them for the link prediction challenge. To mitigate the issue, we have used dimensionality reduction algorithm to reduce the PLM embedding dimension to...
\( \mathcal{Z} \) with a value of less than 768 in order to avoid the curse of dimensionality. Let us define the dimensionality reduction function as \( \phi \). The input to this function is the output of \( \Omega \) which is defined in equation 1. The output of \( \phi \) is a \(|\mathcal{E}| \times \mathcal{Z} \) dimensional vector namely \( \mathcal{E}^Z_{ST} \in \mathbb{R}^{|\mathcal{E}| \times \mathcal{Z}} \). Formally, this can be defined as:

\[
\phi : \mathcal{E}_{ST} \rightarrow \mathcal{E}^Z_{ST} \in \mathbb{R}^{|\mathcal{E}| \times \mathcal{Z}}
\]

(2)

K-means++ is applied on the resultant entity vectors from PLMs \( \mathcal{E}^Z_{ST} \) to obtain \( K \) clusters consisting of entities \( \{e\} \subseteq \mathcal{E} \). Other K-means attributes such as cluster centroids and distances between the cluster centroids are retrieved as well. The objective function for building the neighborhood is aligned with K-means clustering algorithms’ objective, formally defined as:

\[
\arg\min_C \sum_{i=1}^{K} \sum_{e_i \in \mathcal{E}^Z_{ST}} \| \mathcal{E}^Z_{ST} - \mu_i \|^2
\]

(3)

Here, \( \mu \) is a variable, optimized for each cluster centers \( c_i \). Finally, each entities that are mapped to their respective cluster centroids acts as the representative of that particular cluster. After assigning each entity \( e \) to its respective clusters \( c_i \in \mathcal{C} \), we construct the mapping dictionary \( \text{dict} \), where the cluster assignment of each entity is preserved. In \( \text{dict} \), the keys are the set of all the entity symbols \( \mathcal{E} \) and the values are associated to their representative cluster centroids \( \mathcal{C} \) (Equation 4). The distances of the cluster centroids are preserved under the

**Algorithm 1: Building neighborhood.**

**INPUT:** Entity embedding \( \mathcal{E} \) from PLMs, Desired number of clusters \( K \), dimensionality reduction function \( \phi \), reduced dimensionality \( \mathcal{Z} \).

**OUTPUT:** Cluster Dictionary \( \text{dict} \) where keys← mapped entities \( \{e\} \subseteq \mathcal{E} \) values← cluster centroids \( \{c\} \), \( K \) as K-means++ Attributes such as distance to centroids, number of centroids etc.

**Function** Build entity clusters(\( \mathcal{E}, K, \mathcal{Z} \)):

\[
\mathcal{E}_Z \leftarrow \phi(\mathcal{E}, \mathcal{Z})
\]

\[
\text{KM} \leftarrow \text{Initialize K-means (numberofclusters} = K)\]

\[
D_{\mathcal{E}_Z, \mathcal{C}}, \text{KMeans\_attributes} \leftarrow \text{KM}(\mathcal{E}_Z)
\]

// For each entities in \( \mathcal{E}_Z \) obtain assigned cluster number \( \mathcal{C} \)

// \( D_{\mathcal{E}_Z} \) is the distance between cluster centroids

\[
\text{dict} \leftarrow \{\}
\]

for each entity \( e \in \mathcal{E} \) do

\[
\text{dict}[\{\text{key}: e, \text{value}: c\}] \leftarrow \text{map}(\{e\} \subseteq \mathcal{E}_Z, c \in \mathcal{C})
\]

// map the matching entities \( \{e\} \subseteq \mathcal{E} \) belonging to the centroid \( c \in \mathcal{C} \)

Return \( \text{dict}, K \)
attribute of K-means++, namely $K M_\theta$.

$$\text{dict} : \mathcal{E} \rightarrow \mathcal{C}$$

(4)

The neighborhood building process is summarized in Algorithm 1.

3.2 Negative Sample Generation

Typically, KGE models create negative samples from the existing positive triples during the training process. Either the head or the tail is corrupted by replacing it with candidate entities which eventually form negative triples. We follow the standard way to generate negative samples during the training phase. However, our proposed method utilizes the created cluster dictionary $\text{dict}$ and K-means attributes $K M_\theta$ to generate negative samples. Our proposed approach aims to generate negative samples while preserving the contextual meaning of the entity, so that meaningful negative samples are fetched. Our intuition is that, KGE models may exhibit better performance, if meaningful entities are selected while the corruption process.

Let us define the training set as $T_{h,r,t}$, where $h,r,t$ represents head, relation, and tail, respectively. The K-means++ attributes, $K M_\theta$ are used to obtain the distance between candidate entities to be corrupted and the cluster centroids. Our goal is to generate $\mathcal{N}$ negative samples per positive triple $t_{h,r,t}$ in each given batch $B$. We only consider the entities $\{e\} \subseteq \mathcal{E}$, that are within $\mathcal{H}$ hops and distance $d$. Assuming that within $\mathcal{H}$ nearest hops we have $k \in \mathcal{K}$ clusters. For each triple $t_{h,r,t}$, firstly the head or tail corruption is done in a probabilistic manner. For the target entity $e$, we compute the distances from the centroids of each cluster. The information of the distance between clusters is obtained from the $\text{dict}$ which has been obtained in the phase of building the neighborhood (algorithm 1). The clusters are then sorted based on their distances $d$ which is between the entity to be corrupted and the other cluster centroids. Let us define the distance function as $CD$, the cluster centroids that are stored in K-means attribute $K M_\theta$ as $C$, the desired number of clusters as $\mathcal{K}$, and obtained distance vector is $M$, formally defined as:

$$CD(e, C_{KM_\theta}) = M \in \mathbb{R}^{1 \times |\mathcal{K}|}$$

(5)

Upon obtaining the sorted clusters and their member entities, it is possible to obtain $\mathcal{N}$ negative entities randomly as the counterpart for the target entity $e$. With the randomly picked entities $\{e'\}$ from the hop $\mathcal{H}$ clusters within the distance $d$, the negative triple $T_{h',r',t'}$ is formed. Let us consider obtaining $\mathcal{N}$ negative samples $\{e'_0, e'_1, ..., e'_\mathcal{N}\}$ for a particular entity and the number of clusters based on tolerated distances to clusters as $d_{\max}$. The set of possible negative entities resides in the union of all the clusters till that distance $d_{\max}$. We can formally define that as:

$$\{e'_0, e'_1, ..., e'_\mathcal{N}\} \in \mathcal{E}' = \mathcal{E}_{H_0} \cup \mathcal{E}_{H_1} \cup ... \mathcal{E}_{H_{d_{max}}}$$

(6)

The whole process of obtaining the negative sampling during the training process can be described briefly in algorithm 2. In algorithm 2, for each triple $t_{h,r,t}$ in $T_{batch}$, we first decide about the head or tail corruption and apply the corruption on either the head or tail position. The entity to be corrupted is then compared with other cluster centroids. The information about the clusters centroids and their member entities is available in the built neighborhood which is represented as a dictionary $\text{dict}$. It is later sorted based on the distance between the entity to be corrupted namely $\text{entity2corrupt}$ and its distance to other cluster centroids. We consider fetching the entities for the corruption from only the first $\mathcal{H}$ clusters from $\text{dict}_{\text{sorted}}$. The $\mathcal{N}$ corrupted entities are then picked from these nearest clusters in a random fashion. The negative triple is formed by replacing the candidate or target entity in the corrupt position with the chosen $\text{corruption\_candidates}$ which gives us $\mathcal{N}$ corrupted triples $t'$ aligned in the current batch.

4 Experimental Set-up

4.1 Dataset

We evaluate our approach on publicly available datasets of namely: WN18 (Bordes et al., 2013), and WN18RR (Dettmers et al., 2018). WN18 dataset contains lexical relations between words (Sun et al., 2019) where several patterns can be found from that dataset including symmetric, antisymmetric and inverse. WN18RR is a subset of WN18 in which many of the inverse relations are removed due to their leakage. The main patterns in this dataset are: asymmetric, symmetric, and composition (Sun et al., 2019). In Table 4, we provide the statistical information for number of entities, relations, triples, vocabulary and average characters of these datasets. FB15k-237 is not used due to the missing text or missing entity labels. As a result, many of the triples are to be removed from
the dataset and KGE models are unable to perform link prediction in a desired way.

4.2 Hyper-parameter Settings

For WN18 and WN18RR many of the hyperparameters are taken from the best settings of RotatE (Sun et al., 2019), also collected from their Github.

For these two datasets, both adversarial and non-adversarial methods (Sun et al., 2019) have been considered. For the underlying KGEs, the embedding dimension $D=\{400, 500, 1000\}$, the margin $\gamma=\{6.0, 12.0\}$, learning rate $\alpha=\{0.0001, 0.00005, 0.001, 0.002\}$, the temperature $\tau=\{0.5, 1.0\}$, the number of NS $N=\{50, 100\}$, batch size $B=\{512, 1024\}$ were used. The specific hyperparameters of our approach are $K$, $\mathcal{H}$ in the range of $\{10, 20\}$ and $\{2, 3, 5\}$, accordingly which correspond to the number of clusters and number of $d$ distant hops. The dimensionality of the embedding from PLM has been reduced to 200 after performing a number of experiments and initial analysis.

4.3 Baseline Models and Evaluation Metrics

We evaluated our proposed approach on the following baseline models: 1) RotatE - a popular KGE model trained with the score function $|h \circ r - t|$, TransE (Bordes et al., 2013) with the score function $|h + r - t|$, and DistMult is a KGE model that employs a bi-linear formulation with the score function of $|h \ast r \ast t|$.

Evaluation Metrics We follow the baseline models and assess the performance of the KGE models based on different negative sampling methods, with standard metrics such as Hit@10 and Mean Reciprocal Rank (MRR) as reported in (Sun et al., 2019), (Bordes et al., 2013) and (Ali et al., 2021). For the evaluation of the KGE models, we rely on link prediction task where the queries are formed as head prediction (?,$r,t$) and tail prediction ($h,$ $r,$ ??) for an existing true triple in the test set. The ? is replaced by all the entities in the Knowledge Graph in order to generate the false triples. The rank of the true triple is considered as the position of it when sorted based on the plausibility against all the possible combinations of corrupted ones. The mean rank (MR) is the average rank of all the test triples in the test set. The lower score in MR indicates better performance. MRR represents the inverse of the average rank obtained from the score of each test triple. In MRR, a better performance is indicated by a higher value. The Hit@10 represents the accuracy that the target entity for a particular query appeared in the top 10 prediction during the inference. The average Hit@10 score is used to measure the performance on the test set.

5 Evaluation and Analysis

In this section, we present the results of our evaluation on the the two benchmark datasets. We compare against the baseline approaches reported in (Ahrabian et al., 2020) which includes KBGAN (Cai and Wang, 2018), NSCaching (Zhang et al., 2019) and SOTA NS approaches for Self Adversarial Negative Sampling (Sun et al., 2019).
Table 1: Dataset statistics. This table presents the number of entities, relation, triples as well as division of train, validation and test sets, where \textit{Chars} refers to characters on WN18 and WN18RR.

| Dataset  | \(|E|\) | \(|R|\) | \#train | \#valid | \#test | Vocabulary | \text{Avg. #Chars (}\(E\)) | \text{Avg. #Chars (}\(R\)) |
|----------|--------|--------|--------|--------|--------|------------|------------------|------------------|
| WN18     | 40,943 | 18     | 141,442| 5,000  | 5,000  | 126,087    | 19.691           | 19.000           |
| WN18RR   | 40,943 | 11     | 86,835 | 3,034  | 3,134  | 81,954     | 19.691           | 18.455           |

Table 2: Performance of algorithms without adversarial NS. Here, \textit{Uni.} is the short form of the algorithm \textit{Uniform} and \(\Delta\) indicates the difference between LEMON and the other NS approaches on Hit@10. The green color shows the increase and the red color is dedicated for decreases in performance comparisons. “-” indicates the unavailability.

5.1 Evaluation Results

Table 2 and table 3 reports the results of LEMON and the baseline models, with and without adversarial settings (Sun et al., 2019) for NS. For RotatE KGE, the achieved Hit@10 by LEMON on on WN18 and WN18RR are 96.18\% and is 57.33\%, respectively. On WN18, with our proposed negative sampling method, the TransE model achieved a Hit@10 score of 95.57\%. Furthermore, a significant improvement in the performance of TransE is visible in both MRR (95.90\%) and Hit@10 (94.92\%) metrics, using LEMON. For the RotatE model on WN18RR, we observed an improved result in Hit@10. As can be seen, we calculated the \(\Delta\) as the difference of performance in Hit@10 of LEMON with other NS methods. In all the cases, LEMON outperforms other NS approaches used with RotatE, TransE both on WN18, and WN18RR in non adversarial settings (table 2). For the DistMult model, we outperform others on WN18RR in non adversarial settings. Some of the large gains in Hit@10 can be observed in table 2 in WN18RR for DistMult and TransE model. On WN18RR, the improvement against KBGAN, Uniform Negative Sampling, uni-SANS and RW-SANS are +23.36, +7.43, +8.14 and +3.79, respectively. On WN18RR, some of the significant achievements are seen on TransE model against KBGAN and NSCaching approaches which are +7.98 and 3.39 respectively. For achieving comparable results on WN18, a more extensive hyper-parameter search can be deployed which is among our future work. On the MRR metric, LEMON is getting comparable results. We shall note that, the results of the baseline models are reported from the original work on structure aware negative sampling (Ahrabian et al., 2020). In Table 3, similar improvement in Hit@10 can be seen on WN18 and WN18RR datasets. There is a significant gain in Hit@10 for DistMult model compare to LEMON with SANS(+14.10) and RW-
Figure 3: Example set of candidate entities that form negative samples. The Uniform and SANS negative entities are taken from (Ahrabian et al., 2020).

| Anchor Node | Uniform | Uniform SANS | Candidate Nodes | Uniform LEMON |
|-------------|---------|--------------|-----------------|--------------|
| arachnoid   | diner   | refusenurse  | arachnida       | vertebrata   |
|             |         |              | biology         | amphiuranida |
|             |         |              | arthropod       | peturotidae  |
|             |         |              | wolf           | anharchitidae |
|             |         |              | spider          | dinoflaglata |
| empathy     | beach   | pea          | sympathetic     | regular     |
|             |         |              | feeling         | succeed     |
|             |         |              | commiserate     | promiss      |
|             |         |              | passion         | - refusal    |
|             |         |              | pride           | existent    |
|             |         |              | state           | humnaness   |
| wheat       | lend    | align        | wildrice        | Edene       |
|             |         |              | tabletop        | United States |
|             |         |              | barley          | camp.follower |
|             |         |              | mesen      | american.colon |
|             |         |              | bread            | field.frial |
|             |         |              | fruit            | common buterkup |
|             |         |              | beer             | agricultural.labore |
|             |         |              | cumbella        | - susquehanna |

Table 3: Performance of algorithms with adversarial NS. Uni. denotes Uniform and ∆ indicates the difference between LEMON and the other NS approaches on Hit@10. The green and red colors are for increase/decreases in performance comparisons, "-" for not available.

SANS(+10.06). TransE model also obtains significant improvement when comparing it with Uniform, SANS and RW-SANS with LEMON on WN18 dataset. The improvement in Hits@10 are +2.90, +10.86 and +6.41 respectively.

5.2 Relevant Negative Sampling

Figure 3 reports the examples which are drawn as effective candidates for corruption by LEMON. In this regard, a comparison between LEMON, Uniform and SANS on the WN18RR is shown. The output of the baseline NS methods are taken from the SANS paper (Ahrabian et al., 2020). From the investigation, we observed that the contextual relevance decays as we go into the further hops.

The analysis of the drawn examples show that meaning-wise, both LEMON and Uniform SANS provide more relevant candidates than Uniform. This is while Uniform SANS needs a bigger pre-processing time for building neighborhoods. Unlike LEMON, it does not consider any complementary knowledge. Several ablation studies and comprehensive analysis of the performance and plausibility of the proposed method have been conducted which can be found in the Appendix.

6 Conclusion

This paper addresses the problem of creating meaningful negative sampling for knowledge graph embedding models. We presented LEMON, that replaces the old paradigm of creating negative sampling using a random distribution technique. LEMON is model-independent and easy to integrate into widely used KGE models (i.e., TransE, RotatE and DistMult) and evaluated on standard benchmark knowledge graphs, namely WN18 and WN18RR. The empirical results exhibit a significant improvement in the performance of the underlying KGE models. Some of the biggest gains of our approach can be seen in DistMult model against KBGAN, Uniform Negative Sampling, Uni-SANS which are +23.36, +7.43 and +8.14 respectively on WN18RR dataset considering non adversarial settings. In the adversarial settings significant improvement over SANS and RW-SANS can be seen on WN18RR dataset for DistMult model which are +14.10 and +10.06 respectively. In future work, we plan to consider other language models and combine LEMON with other KGE models.
Limitations

The embedding vectors obtained from PLMs are highly important for LEMON. The generation of negative samples considers the quality of the contextual representation of the entity text in order to fetch the relevant and useful negative candidates for corruption. On the other hand, the structure of the underlying Knowledge Graphs often does not align with the the contextual meaning of the text. In order to align them properly, a quality check might be needed for the input text of PLMs.

References

Hervé Abdi and Lynne J Williams. 2010. Principal component analysis. Wiley interdisciplinary reviews: computational statistics.

Kian Ahrabian, Aarash Feizi, Yasmin Salehi, William L. Hamilton, and Avishek Joey Bose. 2020. Structure aware negative sampling in knowledge graphs. In EMNLP, pages 6093–6101.

Mirza Mohtashim Alam, Hajira Jabeen, Mehdi Ali, Karishma Mohiuddin, and Jens Lehmann. 2020. Affinity dependent negative sampling for knowledge graph embeddings. In DL4KG@ESWC.

Mehdi Ali, Max Berrendorf, Charles Tapley Hoyt, Laurent Vermue, Mikhail Galkin, Sahand Sharifzadeh, Asja Fischer, Volker Tresp, and Jens Lehmann. 2021. Bringing light into the dark: A large-scale evaluation of knowledge graph embedding models under a unified framework. IEEE Transactions on Pattern Analysis and Machine Intelligence.

David Arthur and Sergei Vassilvitskii. 2006. k-means++: The advantages of careful seeding. Technical report, Stanford.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics, 5:135–146.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. NeurIPS.

Liwei Cai and William Yang Wang. 2018. KBGAN: Adversarial learning for knowledge graph embeddings. In NAACL-HLT.

Feihu Che, Guohua Yang, Pengpeng Shao, Dawei Zhang, and Jianhua Tao. 2022. Mixkg: Mixing for harder negative samples in knowledge graph. arXiv preprint arXiv:2202.09606.

Ting Chen, Yizhou Sun, Yue Shi, and Liangjie Hong. 2017. On sampling strategies for neural network-based collaborative filtering. In SIGKDD.

Sarthak Dash and Alfio Gliozzo. 2019. Distributional negative sampling for knowledge base completion. arXiv preprint arXiv:1908.06178.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2D knowledge graph embeddings. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 32.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. NeurIPS.

Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In Proceedings of the 26th international conference on world wide web, pages 173–182.

Tinglin Huang, Yuxiao Dong, Ming Ding, Zhen Yang, Wenzheng Feng, Xinyu Wang, and Jie Tang. 2021. Mixgcf: An improved training method for graph neural network-based recommender systems. In SIGKDD (2021).

Stanley Kok and Pedro Domingos. 2007. Statistical predicate invention. In Proceedings of the 24th international conference on Machine learning, pages 433–440.

Alexa T McCray. 2003. An upper-level ontology for the biomedical domain. Comparative and functional genomics, 4(1):80–84.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems, pages 3111–3119.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In (EMNLP-IJCNLP), pages 3982–3992.

Tara Safavi and Danai Koutra. 2020. CoDEx: A Comprehensive Knowledge Graph Completion Benchmark. In (EMNLP), pages 8328–8350.

Zhiqing Sun, Zhi-Hong Deng, Jian-YunNie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. In ICLR.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. JMLR, 9(11).

Michel Verleysen and Damien Francois. 2005. The curse of dimensionality in data mining and time series prediction. In International work-conference on artificial neural networks.

Denny Vrandečić. 2012. Wikidata: A new platform for collaborative data collection. In Proceedings of the 21st international conference on world wide web, pages 1063–1064.
Peifeng Wang, Shuangyin Li, and Rong Pan. 2018. Incorporating gan for negative sampling in knowledge representation learning. In AAAI (2018).

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 28.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In ICLR (2015).

Yongqi Zhang, Quanming Yao, Yingxia Shao, and Lei Chen. 2019. Nscaching: simple and efficient negative sampling for knowledge graph embedding. In ICDE (2019).

A Appendix

A.1 Datasets

Apart from WN18 (Bordes et al., 2013), and WN18RR (Dettmers et al., 2018), we also trained TransE and RotatE on the following datasets: Nations (Kok and Domingos, 2007), UMLS (McCray, 2003), and CoDEx (Safavi and Koutra, 2020)

Nations and UMLS are small scale yet efficient benchmarks. Nations includes a set of relationships between nations and their features (Kok and Domingos, 2007). The dataset consists of binary and unary relations while UMLS dataset (standing for Unified Medical Language System ) is a high-level ontology for organizing a large number of terminologies used in the biomedical domain. It translates into a unified vocabulary that allows for uniform access to medical resources. CoDEx provides three comprehensive knowledge graph datasets that include positive and hard negative triples, entity types, entity, and relation descriptions. The knowledge graph in CoDEx is constructed from Wikidata (Vrandečić, 2012).

A.2 Hyperparameters

For UMLS, Nations and CoDEx-M datasets we have used TransE, and RotatE model for evaluation of our negative sampling approach. For fairness of evaluations, a fixed set of hyperparameters were used. For nations and UMLS the $D$, $\gamma$, $\alpha$, $B$ and $Z$ are set to 100, 12.0, 0.01, 64 and 2 respectively. The $N$, $K$ and $H$ are kept in range of {3,10}, {4,5,6,7,8} and {3,4,5,6,7} accordingly. In CoDEx-M, we used $D$, $\gamma$, $\alpha$, $B$, $N$ as 500, 9, 0.00005, 512, 50. $K$ and $H$ are in the ranges of {8,12} and {7,9} accordingly.

For executing the experiments with Uniform RW-SANS on Nations, UMLS and CoDEx-M datasets, we set the $-\nu$ hyperparameter to 1000 which indicates number of random walks. For Nations, UMLS and CoDEx-M the results are obtained without using negative adversarial sampling.

A.3 Pre-processing time

In Table 5, we compare the pre-processing time (in minutes) for building the neighborhood in Uniform SANS, RW-SANS and LEMON. In terms of the pre-processing time, LEMON outperforms the other two by taking only 2.9052 minutes in total. Meanwhile, Uniform SANS and RW-SANS takes 88.46, and 215.705 minutes accordingly. Although, RW-SANS achieved a comparable performance, depending on the length of the entities and relation it can be really resource hungry. The pre-processing times are computed based on the performance on the CoDEx-M dataset.

B Experimental Result on Nations, UMLS and CoDEx-M

Table 6 reports the results of baseline algorithms with KGE models with RotatE and TransE on Nations, UMLS and CoDEx-M datasets. We observed a performance gain of the KGE models in most of the cases on benchmark datasets with our proposed approach LEMON proposed method. The difference between the performance of LEMON and Uniform is stated as $\Delta_1$ which is calculated for Hits@1, 3 and 10. $\Delta_2$ is corresponding to the difference between the LEMON and Uniform SANS while $\Delta_3$ is demonstrating the difference between the LEMON and RW SANS negative sampling. A considerable performance increase in different models specially in RotatE (by a margin of +3.86 on Hit@1) and TransE (by a margin of +6.28 on Hit@1) can be observed. In few cases, the performance was just comparable with other NS approaches which is (to our observations) due to the structural dominance attribute of the underling KGs. This fact affects the adaptability of the NS approaches on the employed KGE models. We discuss these aspects further in details.

B.1 Ablation Study on Time Complexity

Table 7 demonstrates the complexity of LEMON and the baseline algorithms in terms of pre-processing, run-time, and space complexity. Here, $t$ is the number of GAN parameters, $N$ denotes the
Table 4: Dataset statistics. This table presents the number of entities, relation, triples as well as division of train, validation and test sets, where $\text{Chars}$ refers to Characters of Nations, UMLS and CoDEX-M.

| Dataset  | $|E|$ | $|R|$ | $|\text{train}|$ | $|\text{valid}|$ | $|\text{test}|$ | Vocabulary | Avg. $\text{Chars} (E)$ | Avg. $\text{Chars} (R)$ |
|----------|------|------|----------------|----------------|----------------|-------------|----------------|------------------|------------------|
| Nations  | 14   | 55   | 1,592          | 199            | 201            | 1,992       | 7.786         | 14.455           |
| UMLS     | 135  | 46   | 5,216          | 652            | 661            | 2,614       | 20.830        | 13.336           |
| CoDEX-M  | 17,050 | 51 | 185,584        | 10,310         | 10,311         | 23,492      | 45.435        | 102.725          |

Table 5: Pre-processing performance of LEMON and baseline NS methods.

| Methods    | Total pre-processing time |
|------------|---------------------------|
| Uniform SANS | 88.46 minutes            |
| RW-SANS    | 215.705 minutes           |
| LEMON      | Embedding Generation: 2.018 minutes |
|            | Cluster Formation: 0.08 minutes |
|            | Total Time: 2.9052 minutes |

The number of negative samples, $B$ is the batch size, $E$ is an edge set, $\mathcal{E}$ is the set of entities, $R$ for relation set, and $D$ represents the embedding dimension. Furthermore, $H$ is the hops count, $r$ for random walk count, and $Z$ is a PCA dimensionality reduction parameter. In the pre-processing part, LEMON has the same dimensionality reduction complexity of PCA, and K-means. The time complexity of the K-means++ algorithm is $O(\frac{K}{I} I Z)$, where $D_{\text{PLM}}$ indicates the output feature dimension of PLM. The time complexity of the K-means++ algorithm is $O(\frac{|\mathcal{E}|}{I} I Z)$. Since we are using a reduced dimension for K-means clustering, the feature dimension is $Z$. Here, $I$ is the number of iteration and $|\mathcal{E}|$ is the length of the entity set. LEMON’s runtime complexity is $O(|\mathcal{H}| I Z + BN)$. During the training, LEMON sorts the desired number of Hops $\mathcal{H}$, based on the computed distances. The sorting complexity is only added to our approach, comparing to the Uniform negative sampling. The space complexity of our proposed method is $O(|\mathcal{H}|)$, which only depends on the number of desired hops. Other complexity of other NS approaches in Table 7 are taken from SANS (Ahrabian et al., 2020).

B.2 Ablation Study on Trained Entity Embedding

To demonstrate the effectiveness of the language model based clustering in the negative sampling, we illustrate the clusters obtained from the pre-trained embeddings from the Sentence Transformer model in Figure 4, Sentence BERT in the left and fastText (Bojanowski et al., 2017) in the right. The clusters depicted in Figure 4 are constructed based on the UMLS dataset. Through a systematic analysis, the entities which are supposed to be in the same cluster are marked in the same color either green or red. For example, clinical\_drug, medical\_device, research\_device are green that belong to the same cluster. That means the embeddings generated by PLMs are already possessing pre-trained knowledge which can be leveraged by KGEs during the training process. Using PLMs has a trade-off as the embeddings generated by them are rich in textual meaning, they lack structural information which can be alleviated by KGE models. Figure 5, demonstrates the effect of PLMs on NS sampling by LEMON where the entities are well-separated in different clusters (see the left side of Figure 5). However, there are overlapping entities between the two groups in the case that KGE model are trained by Uniform (see the right side of Figure 5). In Figure 5, we observed that, there are certain entities that are not clustered in the expected group for Uniform negative sampling technique. Therefore, for the analysis purpose, we have used UMLs dataset due to its exclusivity. 'manufactured\_device', 'drug\_delivery\_device', 'clinical\_drug', 'medical\_device' and 'research\_device' are the entities (in green) which are classified with 'language', 'molecular\_sequence', 'spacial\_concept' etc for the trained embeddings from uniform negative sampling (in red). In our approach, both the red and green entities belong to two separate group of clusters.

For the visualization purpose, we used embeddings from a trained checkpoint of both Uniform and LEMON NS methods. TSNE (Van der Maaten and Hinton, 2008) is employed to reduce the dimension of the trained embedding to 2. The axes are denoted as TSNE-1, and TSNE-2 which indicate the two reduced dimensions in Figure 5. Due to the space limitation, we restricted the number of entity labels that appear in the visualization to a subset of the entire entity set. For clustering the entity embeddings, K-means++ has been trained till 1000 iterations. We obtained the KGE embeddings from...
the RotatE model. The hyperparameters used in RotatE this purpose are as follows: $D=200$, $B=64$, $K=6$, $N=3$, $\gamma=12$, $H=2$ and, $\alpha=0.1$.

### B.3 Influence of PLMs: Structure vs. Textual Knowledge

During our experiment we observed that, the impact of using PLMs in KGEs depend on the richness of the textual information. Consider an example from the CoDEx-M dataset, the pre-trained embedding representation of the entity $e$ with text "1922 1991 country in Europe and Asia", carries less meaningful information about the context. In such cases, the KGE model benefits from its own capability in learning graph structure that preforms better. We call it structurally dominant KG (i.e., CoDEx-M – see Figure 7), since the meaning of the text is not very relevant to the corresponding triple. On the other hand, for some KGs (i.e., UMLS) the PLM embeddings not only aligned with the structure-based embeddings, but also provide more informative candidates (Figures 8). For instance, the pre-trained embeddings from Sentence-BERT (left) shows similarity with the one from a KGE model (right) (i.e, “mammal” and “human”). The heatmap of the following entities also confirm the claim: ‘clinical_drug’, ‘drug_delivery_device’, and ‘medical_device’ having similarity in heatmap makes the pre-trained entities from the PLMs agree with the one from a trained KGE model. Hence, this dataset can be considered as balanced.

Figure 7 demonstrates dissimilarities between the entity embeddings from the PLMs and KGE model. Consider the following entities: ‘German philosopher’ and ‘German poet’, ‘philosopher’, ‘historian’, and ‘playwright’. Though they show high similarity measures in the heatmap of PLMs, their similarity values are relatively low in the KGE model-based heatmap. For instance, the information provided in the PLM could capture the location information very well. How-

### Table 6: Result of different KGE models with different negative sampling techniques on Nations, UMLS, CoDEx-M datasets. Here, $Uni.$ is the short form of the algorithm Uniform and $\Delta$ indicates the difference between LEMON and the baseline algorithms, in terms of Hit@10. $\Delta_1$, $\Delta_2$, and $\Delta_3$ refer to the difference between LEMON and $Uni.$, $Uni.$ $SANS$ and $Uni.$ $RW-SANS$, respectively. The green color shows the increase and the red color is dedicated for decreases in performance comparisons. The outperforming cases are shown as bold for LEMON.

### Table 7: Complexity comparison of negative sampling techniques in terms of pre-processing, run-time and space complexity.
however, it does not correspond to the information from the structured KG. Even though often the embeddings from the PLMs corresponds to the structural information in KGE (i.e., 'city in Hesse, Germany', city in Hessen, Germany), in many cases entities which are similar, the embedding from the PLM end up in a different cluster in the KGE model. This happens due to their differences in symbolic structural information. In the cases of structural dominance the quality of the clusters reduces and fetches less meaningful negative samples in our case. Figure 8 demonstrates a balanced scenario where the structural and contextual information from the text exists, which we observed from the patterns in the heatmap. We noticed several examples such as ‘animal’, ‘mammal’, and ‘human’ that have higher cosine similarity among themselves in both of the heatmaps. On the other hand less similarity exists in 'mammal' and 'clinical_drug' in both the images.
Figure 7: Difference between words from different clusters in CODEx-m dataset.

Figure 8: Correlation between words from different clusters in UMLS dataset.

Figure 9: Comparison of KGE embedding (right) vs. Sentence-BERT (left) in both pair of figures. (a) embeddings differs between PLM and KGEs. (b) embeddings of PLMs and KGEs are aligned.