Structure Aware Negative Sampling in Knowledge Graphs

Kian Ahrabian*,1,3, Aarash Feizi*,1,3, Yasmin Salehi*,2, William L. Hamilton1,3,4 and Avishek Joey Bose1,3
1 School of Computer Science, McGill University, Canada
2 Department of Electrical and Computer Engineering, McGill University, Canada
3 Montreal Institute of Learning Algorithms (Mila), Canada
4 Canada CIFAR AI Chair
{kian.ahrabian, aarash.feizi, yasmin.salehi} {mail.mcgill.ca}

Abstract
Learning low-dimensional representations for entities and relations in knowledge graphs using contrastive estimation represents a scalable and effective method for inferring connectivity patterns. A crucial aspect of contrastive learning approaches is the choice of corruption distribution that generates hard negative samples, which force the embedding model to learn discriminative representations and find critical characteristics of observed data. While earlier methods either employ too simple corruption distributions, i.e. uniform, yielding easy uninformative negatives or sophisticated adversarial distributions with challenging optimization schemes, they do not explicitly incorporate known graph structure resulting in suboptimal negatives. In this paper, we propose Structure Aware Negative Sampling (SANS), an inexpensive negative sampling strategy that utilizes the rich graph structure by selecting negative samples from a node’s k-hop neighborhood. Empirically, we demonstrate that SANS finds semantically meaningful negatives and is competitive with SOTA approaches while requires no additional parameters nor difficult adversarial optimization.

1 Introduction
Knowledge Graphs (KGs) are repositories of information organized as factual triples \((h,r,t)\), where head and tail entities are connected via a particular relation \((r)\). Indeed, KGs have seen wide application in a variety of domains such as question answering (Yao and Van Durme, 2014; Hao et al., 2017; Moldovan and Rus, 2001) and machine reading (Weissenborn et al., 2018; Yang and Mitchell, 2017) to name a few and have a rich history within the natural language processing (NLP) community (Berant et al., 2013; Yu and Dredze, 2014; Collobert and Weston, 2008; Peters et al., 2019). While often large, real-world KGs such as FreeBase (Bollacker et al., 2008) and WordNet (Miller, 1995) are known to be incomplete. Consequently, KG completion via link prediction constitutes a fundamental research topic ameliorating the practice of important NLP tasks (Sun et al., 2019; Angeli and Manning, 2013).

In recent years, there has been a surge of methods employing graph embedding techniques that encode KGs into a lower-dimensional vector space facilitating easier data manipulation (Zhang et al., 2019) while being an attractive framework for handling data sparsity and incompleteness (Wang et al., 2018). To learn such embeddings, contrastive learning has emerged as the de facto gold standard. Indeed, contrastive learning approaches enjoy significant computational benefits over methods that require computing an exact softmax over a large candidate set, such as over all possible tail entities given a head and relation. Another important consideration is modeling needs, as certain assumptions are best expressed as some score or energy in margin-based or un-normalized probability models (Smith and Eisner, 2005). For example, modeling entity relations as translations or rotations in a vector space naturally leads to a distance-based score to be minimized for observed entity-relation-entity triplets (Bordes et al., 2013).

Leveraging contrastive estimation to train KG embedding models involves optimizing the model by pushing up the energy with respect to observed positive triplets while simultaneously pushing down energy on negative triplets. Consequently, the choice of negative sampling distribution plays a crucial role in shaping the energy landscape as simple random sampling—e.g. Noise Contrastive Estimation (NCE) (Gutmann and Hyyrinen, 2010)—produces negatives that are easily classified and provide little information alongside in the form of a gradient signal. This is easily remedied if the

*Equal contribution, names ordered alphabetically.
corruption process selects a hard negative example through more complex negative sampling distribution, such as adversarial samplers (Cai and Wang, 2018; Bose et al., 2018; Sun et al., 2019). However, adversarial negative sampling methods are computationally expensive, while more tractable approaches—e.g. cache-based methods (Zhang et al., 2019)—are not tailored to the KG setting as they fail to incorporate known graph structure as part of the sampling process. This raises the important question of whether we can obtain a computationally inexpensive negative sampling strategy while benefiting from the rich graph structure of KGs.

**Present Work.** In this work, we introduce *Structure Aware Negative Sampling* (SANS), an algorithm that utilizes the graph structure of a KG to find hard negative examples. Specifically, SANS constructs negative samples using a subset of entities restricted to either the head or tail entity’s $k$-hop neighborhood. We hypothesize that entities that are within each other’s neighborhood but share no direct relation have higher chances of being related to one another and thus are good candidates for negative sampling. We also experiment with a dynamic sampling scheme based on random walks to approximate a node’s local neighborhood. Empirically, we find that negative sampling using SANS consistently leads to improvements upon uniform sampling and sophisticated Generative Adversarial Network (Goodfellow et al., 2014) (GAN) based approaches at a fraction of the computational cost, and is competitive with other SOTA approaches with no added parameters.

### 3 Structure Aware Negative Sampling

Given an observed positive triplet $(h, r, t)$, a negative sample can be constructed by corrupting either the head or tail entity to form a new triplet—i.e. $(h', r, t')$—where either $h', t' \in E$, where $E$ is the set of all entities in the KG. Additionally, we assume that the graph embedding models are trained using a loss function of the following form:

$$
L = -\log \sigma(\gamma - d_r(h, t)) - \sum_{i=1}^{n} \frac{1}{n} \log \sigma(d_r(h'_i, t'_i) - \gamma)
$$

where $d_r(h, t)$ denotes the score assigned to the compatibility of head and tail entities under the relation $r$, $\gamma$ is a fixed margin, $\sigma$ is the sigmoid function, and $n$ is the number of negative samples.

In this paper, we seek to explicitly use the rich graph structure surrounding a particular node when distinguishing observed positive data and fictitious negative examples that are generated by corrupting the positive examples. This general approach is a simplification of NCE, which is based on a Monte-Carlo approximation of the partition function used in Importance Sampling (IS) (Bengio et al., 2003).

### Non-Fixed Negative Sampling

As proposed in (Mikolov et al., 2013), negative triplets can be generated using a uniform sampling scheme. However, such uniform and fixed sampling schemes result in easily-classified negative triplets during training, which do not provide any meaningful information (Sun et al., 2019; Zhang et al., 2019). Hence, as the training progresses, most of the sampled negative triplets receive small scores and almost zero gradients, impeding the training of the graph embedding model after only a small number of iterations.

To address the issue of easy negatives, Sun et al. (2019) propose Self-Adversarial negative sampling, which weighs each sampled negative according to its probability under the embedding model. Alternatively, the authors in (Wang et al., 2018) and (Cai and Wang, 2018) try creating high-quality negative samples by exploiting GANs, which, while effective, are expensive to train and require black-box gradient estimation techniques. Another elegant approach that uses fewer parameters and is easier to train compared to GAN-based methods is NSCaching (Zhang et al., 2019), which involves using a cache of high-quality negative triplets—i.e. those with high scores.
generating negative triplets. We motivate our approach based on the observation that prior work in learning word embeddings (Mikolov et al., 2013), where negative sampling has historically developed, lacked the richness of graph structure that is immediately accessible in the KG setting. Consequently, we hypothesize that enriching the negative sampling process with structural information can yield harder negative examples, crucial to learning effective embeddings. Fig. 1 highlights our approach, which requires the construction of the $k$-hop neighborhood ($K$) for each node at its first step, \[ K = S^+(A^k + A^{k-1}) \] for $k > 0$, where $k$ is an integer, representing the neighborhood radius, $A$ is the KG’s adjacency matrix, and $S^+$ is the element-wise sign function set to 1 if a path exists and 0 otherwise.

To construct negatives triplets, we may now simply sample from the nonzero cell of $K$, which represents a subset of all entities for each node in the KG—i.e. $K \subset 1^{E \times E}$. Intuitively, SANS exploits the locality of an entity’s neighborhood, where negative samples are defined as entities that are not directly linked under a relation $r$ but can be accessed through a path of at most length $k$. We argue that such local negatives are harder to distinguish and lead to higher scores as evaluated by the embedding model. One important technical detail in constructing $K$ is the existence of multiple relation types, which requires an additional dimension to represent the graph connectivity as adjacency and $k$-hop tensors.

### 3.1 Variants of SANS

Although SANS requires a one-time preprocessing step to construct $K$ as defined in Eqn. 2, this may still be costly for large and dense KGs. To combat this inefficiency, we introduce RW-SANS in Alg. 1, which uses $\omega$ random walks (Perozzi et al., 2014) of length $k$ in the adjacency tensor to approximate the $k$-hop neighborhood.

As SANS constructs a local neighborhood from which negative samples are drawn, it can also be combined with other negative sampling approaches. In this work, we extend the Self-Adversarial approach in (Sun et al., 2019) and combine it with SANS by restricting the negative triplet candidate set to the $k$-hop neighborhood. In the subsequent sections, we refer to this technique as Self-Adversarial (Self-Adv.) SANS, whereas the former approach is referred to as Uniform SANS.

### 4 Experiments

We investigate the application of SANS-based negatives to train KG embedding models based on the TransE, DistMult, and RotatE models for the task of KG completion. We evaluate our proposed approach on standard benchmarks, consisting of FB15K-237 (Bollacker et al., 2008), WN18 and WN18RR (Miller, 1995). From our experiments we seek to answer the following questions:

**(Q1)** Hard Negatives: Can we sample hard negatives purely using graph structure?

**(Q2)** Can we combine graph structure with other SOTA negative samplers?

**(Q3)** Can we effectively approximate the adjacency tensor with random walks?

In our experiments, we rely on three representative baselines, namely uniform negative sampling (Bordes et al., 2013), KBGAN (Cai and Wang, 2018), and NSCaching (Zhang et al., 2019). We also compare with the current SOTA approach in Self-Adversarial negative sampling (Sun et al., 2019), and we test whether local graph structure can also be leveraged in this setting.

#### 4.1 Results

We now address the core experimental questions. **Q1:** Table 1 summarizes our main quantitative results where we highlight SANS and RW-SANS. We also compute the difference between the best variant of SANS against the best performing baseline in row $\Delta$. Overall, we find that SANS negatives almost always lead to harder negative samples over Uniform and KBGAN negatives on all three datasets. Furthermore, SANS achieves competitive performance with NSCaching when combined with...
with TransE, and is the second best-performing algorithm when combined with DistMult without requiring additional parameters. We observe average $\Delta$ values of 0.0231, $-0.0341$, and 0.0056 in MRR for TransE, DistMult, and RotateE respectively, which confirm our approach’s effectiveness compared to SOTA while remaining computationally efficient.

We also qualitatively investigate the semantic hardness of SANS negatives against negatives generated via uniform sampling. For instance, using the center node “arachnoid” in the WN18RR dataset as an example, the negatives sampled via SANS within a 2-hop neighborhood are “arachnida,” “biology,” “arthropod,” “wolf spider,” and “garden spider,” while the ones picked by uniform sampling are “diner,” “refusal,” “landscape,” “rise,” and “nurser.” Clearly, the negatives found via SANS are semantically harder to distinguish, and as a result, they also confirm the importance of incorporating graph structure into negative samplers to aid in ‘hard’ negative mining. A more detailed qualitative analysis of negative samples—including the effect of varying neighborhood sizes—generated by SANS can be found in C.1.

Q2.: We now combine our approach SANS with Self-Adversarial negative sampling (Sun et al., 2019). Our results are presented in Table 2 under Self-Adv. SANS and Self-Adv. RW-SANS, both of which reweigh the negative triplets as done in (Sun et al., 2019). We observe comparable performance between the two approaches, but crucially this is achieved by mostly considering each is achieved by mostly considering the partially-filled adjacency tensors improve the semantic hardness of entities in the datasets like in WN18 and WN18RR, as indicated in Table 3. By considering that the partially-filled adjacency tensors improve computational feasibility for requiring less memory and allowing sparse tensor operations to take place.

### Table 1: Comparison of different negative sampling algorithms. Bold and marked bold* numbers represent the best SOTA and SANS algorithms respectively.

| Score Function | Algorithm | FB15K-237 | WN18 | WN18RR |
|----------------|-----------|----------|------|--------|
|                |           | Hit@10 (%) | MRR  | Hit@10 (%) | MRR  | Hit@10 (%) | MRR  |
| TransE         | NSCaching (Zhang et al., 2019) | 46.59 | 0.2926 | 94.80 | 0.6606 | 43.24 | 0.1808 |
|                | Uniform (Sun et al., 2019) | 48.03 | 0.2927 | 95.53 | 0.6085 | 49.63 | 0.2022 |
|                | Uniform SANS (ours) | 48.35 | 0.2962 | 95.09 | 0.8028$^\ast$ | 51.15 | 0.2254 |
|                | Uniform RW-SANS (ours) | 48.50$^\ast$ | 0.2981$^\ast$ | 95.22$^\ast$ | 0.8195 | 53.41$^\ast$ | 0.2317$^\ast$ |

### Table 2: Comparison of the Self-Adversarial negative sampling technique with our Self-Adversarial SANS. Marked bold* numbers are the results of the best SANS implementation.

| Score Function | Algorithm | FB15K-237 | WN18 | WN18RR |
|----------------|-----------|----------|------|--------|
|                |           | Hit@10 (%) | MRR  | Hit@10 (%) | MRR  | Hit@10 (%) | MRR  |
| TransE         | Self-Adv. (Sun et al., 2019) | 52.73 | 0.3296 | 92.02 | 0.7722 | 52.78 | 0.2232 |
|                | Self-Adv. SANS (ours) | 52.03$^\ast$ | 0.3265$^\ast$ | 84.06 | 0.7136 | 53.21 | 0.2249 |
|                | Self-Adv. RW-SANS (ours) | 50.04 | 0.3060 | 88.51$^\ast$ | 0.7429$^\ast$ | 53.81 | 0.2273$^\ast$ |
| DistMult       | Self-Adv. SANS (ours) | 48.17 | 0.3071 | 91.08 | 0.6634 | 48.24 | 0.3091 |
|                | Self-Adv. RW-SANS (ours) | 48.48$^\ast$ | 0.3100$^\ast$ | 93.04$^\ast$ | 0.7561$^\ast$ | 53.80 | 0.4499 |
|                |          | $\Delta$  | -0.70 | -0.0031 | -3.51 | -0.0293 | +1.03 | +0.0041 |
| RotateE        | Self-Adv. SANS (ours) | 53.03 | 0.3362 | 96.05 | 0.9498 | 57.29 | 0.4760 |
|                | Self-Adv. RW-SANS (ours) | 51.07 | 0.3161 | 96.09$^\ast$ | 0.9496$^\ast$ | 56.94 | 0.4805$^\ast$ |
|                |          | $\Delta$  | +0.09 | -0.0004 | +0.04 | -0.0002 | -0.17 | +0.0045 |
the appeal of incorporating graph structure while choosing negative samples is further highlighted.

| k  | 2  | 3  | 4  | 5  |
|----|----|----|----|----|
| FB15K-237 | 34 | 83 | 97 | 99 |
| WN18    | 0.19 | 0.75 | 3.22 | 10.2 |
| WN18RR  | 0.16 | 0.65 | 2.76 | 8.67 |

Table 3: Percentage (%) of filled entries in the $k$-hop adjacency tensor.

Q3: We now analyze the impact of approximating the local neighborhood using random walks. Fig. 2 depicts the effect of varying the number of random walks ($\omega$) with neighborhoods of different radii and MRR. We report two baselines, one being the performance of uniform sampling, and the other being our best performance achieved by Uniform SANS when combined with TransE, for which the $k$-hop tensor was explicitly computed. Interestingly, we find that the $k$-hop tensor can not only be well approximated with 3000 random walks, but RW-SANS beats both baselines. We reconcile this result by noting that certain nodes have a higher probability of being sampled due to sharing a larger number of paths with the center node, resulting in an implicit weighted negative sampling scheme.

Figure 2: The performance of Uniform RW-SANS with TransE on FB15K-237 using different $\omega$ values.

5 Conclusion and Future Directions

In this work, we introduced SANS, a novel negative sampling strategy, which directly leverages information about $k$-hop neighborhoods to select negative examples. Our work sheds light on the need and importance of incorporating graph structure when designing negative samplers for KGs, and for which SANS can be seen as a cheap yet powerful baseline that requires no additional parameters or difficult optimization. Empirically, we find that SANS-based negatives have comparable performance with SOTA approaches and even outperform previous sophisticated GAN-based approaches.

Acknowledgments

The authors would like to thank the anonymous EMNLP reviewers for their helpful and constructive feedback. This research was supported by a Canada CIFAR AI Chair and NSERC Discovery Grant RGPIN-2019-0512. Avishek Joey Bose is also generously supported through the IVADO PhD fellowship.

References

Gabor Angeli and Christopher D Manning. 2013. Philosophers are mortal: Inferring the truth of unseen facts. In Proceedings of the seventeenth conference on computational natural language learning, pages 133–142.

Yoshua Bengio, Jean-Sébastien Senécal, et al. 2003. Quick training of probabilistic neural nets by importance sampling. In AISTATS, pages 1–9.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In Proceedings of the 2013 conference on empirical methods in natural language processing, pages 1333–1344.

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

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In Advances in neural information processing systems, pages 2787–2795.

Avishek Joey Bose, Huan Ling, and Yanshuai Cao. 2018. Adversarial contrastive estimation. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1021–1032.

Liwei Cai and William Yang Wang. 2018. Kbgan: Adversarial learning for knowledge graph embeddings. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pages 1470–1480.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In Proceedings of the 25th international conference on Machine learning, pages 160–167.
A Experimental Settings

This section provides an overview of the datasets and evaluation protocols used for obtaining our results.

A.1 Datasets

To conduct experiments for our proposed methods, datasets FB15K-237, WN18, and WN18RR were used. FB15K-237 is a subset of FB15K, which has been derived from the FreeBase Knowledge Base (KB) (Bollacker et al., 2008), a large database that contains general facts about the world with many different relation types. On the other hand, WN18 and WN18RR contain less relation
types than FB15K-237. A summary of the number of entities and relation types corresponding to each of these datasets is provided in Table 4.

| Dataset     | #entity | #relation |
|-------------|---------|-----------|
| FB15K-237   | 14,541  | 237       |
| WN18        | 40,943  | 18        |
| WN18RR      | 40,943  | 11        |

Table 4: Dataset Information (Sun et al., 2019).

A.2 Evaluation Protocols

To evaluate our negative sampling approach, we used standard evaluation metrics, consisting of Mean Reciprocal Rank (MRR) and Hits at N (H@N). The train/validation/test split information is provided in Table 5.

| Dataset     | #training | #validation | #test   |
|-------------|-----------|-------------|---------|
| FB15K-237   | 272,115   | 17,535      | 20,466  |
| WN18        | 141,442   | 5,000       | 5,000   |
| WN18RR      | 86,835    | 3,034       | 3,134   |

Table 5: Train/Validation/Test Split Information (Sun et al., 2019).

B Implementation Details

This section of the supplemental goes over the implementation details of our RW-SANS algorithms—i.e. Uniform RW-SANS and Self-Adv. RW-SANS, which use random walks to approximate the k-hop adjacency tensor. Other experimental setups are further detailed herein.

B.1 Hyperparameters

\(k\) and \(\omega\) (for when the k-hop neighborhood is being approximated by Alg. 1) are the hyperparameters in our negative sampling algorithms. To find the optimal hyperparameter values that resulted in the highest performance on the validation set of different datasets, \(k\) and \(\omega\) values in range 2 to 8 and 1000 to 5000 were used respectively during the negative sampling step. In other words, the best performances on the validation sets in our empirical study were found by manual tuning of the hyperparameters. More information about the experimental trials can be found in Table 6, where the total number of trials for training each of the graph embedding models on each dataset is also indicated.

| Algorithm     | k-range | \(\omega\)-set | #trials |
|---------------|---------|----------------|---------|
| Uniform/      | 2-8     | N/A            | 7       |
| Self-Adv      |         |                |         |
| Uniform RW/   | 2-5     | \{1000, 1500,  
| Self-Adv RW   |         |   . . . , 4500\} | 32     |

Table 6: Hyperparameter combination sets and number of trials per model.

Additionally, Table 7 lists the hyperparameters with which different graph embedding models were trained to reach their optimal performance on the validation sets.

| Hyperparameter | TransE | DistMult | RotatE |
|----------------|--------|----------|--------|
| Embedding      | 1024   | 1024     | 1024   |
| Dimension      |        |          |        |
| Batch Size     | 1000   | 2000     | 1000   |
| \(\gamma\)     | 9      | 200      | 9      |
| Optimizer      | Adam   | Adam     | Adam   |
| \(\alpha\)     | 5E-05  | 1E-03    | 5E-05  |

Table 7: Graph Embedding Models’ Hyperparameters.

B.2 Preprocessing

Building the k-hop neighborhood of the nodes within the KG can be regarded as the preprocessing step, essential to implementing SANS. In this paper, we propose two techniques for doing so, which are:

1. explicit computation of the k-hop neighborhood by manipulating Eqn. 2 while accounting for different relation types and,
2. approximation of the k-hop neighborhood using random walks, as detailed in Alg. 1.

B.3 Infrastructure Settings

The experiments in our study were carried on a server with one NVIDIA V100 GPU, 10 CPU cores, and 46GB RAM.

C Experimental Results

C.1 Qualitative Assessment of Negative Samples

In this section, we assess the semantic meaningfulness of negative samples produced by Uniform SANS and those produced by uniform sampling using the WN18RR dataset. As presented by the
Table 8: Example set of candidate nodes to form a negative triplet given an anchor node produced by uniform sampling (Uniform) and Uniform SANS. In this table, $k$ refers to the radius of the $k$-hop neighbourhood from which the candidate nodes are drawn by SANS.

| Anchor Node | Uniform | Uniform SANS |
|-------------|---------|--------------|
|             | $k = 2$ | $k = 3$      | $k = 4$ | $k = 5$ |
| diner       | arachnida | biological  | plectronathi | actinidia |
| refusal     | biology   | ostracoda    | neritidea   | bangiaceae |
| landscape   | arthropod | subkingdom  | amphiban_family | barn_spider |
| rise        | wolf      | placodermi  | pelecaniformes | holarhenna |
| nurse       | spider    | scyphozoa   | categorize   | lucilia    |

| Candidate Nodes | $k = 2$ | $k = 3$ | $k = 4$ | $k = 5$ |
|-----------------|---------|---------|---------|---------|
| arachnoid       | beach   | sympathetic | cheerlessness | cheerfulness |
| refusai         | pea     | sympathizer | ambition | sympathize |
| landscape       | albinism | feeling | expectation | have a bun in the oven |
| rise            | micromeria | commiserate | pride | pleasure |
| nurse           | banking industry | commiseration | state | attribute |
|                  |         |         |         |         |
| wheat           | lend    | wild     | fast     | Edirne   |
| align           | rice    | salad    | United States | Jena    |
| doodad          | tabbouleh | mess | fixings | seasoning |
| mismanage       | barley  | stodge   | form | Washington |
| semiconductor   | buckwheat | meal | Iraqi Kurdistan | Bursa  |

Table 9: The hyperparameter values associated with the best performance on the validation sets, used for obtaining the test results. The different variations of SANS in this table explicitly compute the $k$-hop adjacency tensor.

| Dataset  | Score Function | SANS Algorithm | $k$ | H@10 Validation | Test | MRR Validation | Test |
|----------|----------------|----------------|-----|-----------------|------|----------------|------|
| FB15K-237| TransE         | Uniform        | 3   | 48.35           | 0.3010 | 0.2962        |
|          | Self-Adversarial | 3              | 52.51 | 0.3340 | 0.3265 |
|          | DistMult        | Uniform        | 3   | 40.86           | 0.2599 | 0.2595        |
|          | Self-Adversarial | 3              | 49.07 | 0.3131 | 0.3100 |
|          | RotatE          | Uniform        | 3   | 48.64           | 0.3031 | 0.2985        |
|          | Self-Adversarial | 5              | 53.72 | 0.3432 | 0.3358 |
| WN18     | TransE          | Uniform        | 5   | 95.89           | 0.8237 | 0.8288        |
|          | Self-Adversarial | 5              | 95.69 | 0.9492 | 0.9494 |
|          | DistMult        | Self-Adversarial | 3  | 92.90           | 0.7534 | 0.7561        |
|          |                    | Uniform        | 4   | 95.61           | 0.9492 | 0.9494        |
|          | RotatE          | Self-Adversarial | 5  | 95.61           | 0.9492 | 0.9494        |
| WN18RR   | TransE          | Uniform        | 4   | 50.89           | 0.2228 | 0.2254        |
|          | Self-Adversarial | 8              | 52.46 | 0.2207 | 0.2249 |
|          | DistMult        | Uniform        | 6   | 44.73           | 0.4047 | 0.4025        |
|          | Self-Adversarial | 8              | 39.01 | 0.3749 | 0.3684 |
|          | RotatE          | Self-Adversarial | 4  | 55.78           | 0.4816 | 0.4769        |

Examples given in Table 8, Uniform SANS results in negative examples that are harder to distinguish semantically compared to uniform sampling. We also notice the semantic meaningfulness of SANS-based negatives decline as we increase the size of the $k$-hop neighbourhood. This observation is indeed expected since as the neighbourhood increases in size (i.e. $k \to \infty$) Uniform SANS will become analogous to uniform sampling.

C.2 SOTA Algorithms

Results for the Uniform and Self-Adversarial algorithms in Table 1 and Table 2 respectively were achieved by re-running the code provided by (Sun et al., 2019) using the hyperparameters they reported for the best performance on the validation set of different datasets. Additionally, the results for KBGAN and NSCaching in Table 1 are the scratch results directly taken from (Zhang et al., 2019).

C.3 SANS Algorithms

Table 9 and Table 10 report the performance of the graph embedding models fused with our negative sampling techniques on the validation and test sets with respect to the evaluation metrics. Additionally, they list the hyperparameter values corresponding to Uniform/Self-Adv. SANS and Uniform/Self-Adv. RW-SANS that resulted in the best performance on the validation sets. Based on our out-
Table 10: The hyperparameter values associated with the best performance on the validation sets, used for obtaining the test results. The different variations of SANS in this table approximate the $k$-hop adjacency tensor by random walks (RW-SANS) using Alg. 1.

Table 11: Comparison of different negative sampling algorithms in terms of preprocessing, runtime, and space complexities given batch size $b$, negative sample size $n$, cache size $c$, cache extension size $e$, node set $V$, edge set $E$, relation set $R$, embedding dimension $d$, hops count $k$, random walks count $r$, and GAN parameters count $t$.