On Leakage in Some Popular Benchmarks on Graphs

Anonymous ACL submission

Abstract

A number of benchmarks are based on graphs. Edges are typically split into train, validation and test splits, using a random partition. Leakage has been discovered in a number of popular benchmarks; FB15k has been replaced by FB15k-237 and WN18 has been replaced by WN18RR, though leakage has been reported even after these corrections. This paper will report a new type of leakage, A-leakage, on benchmarks for synonym-antonym classification. A-leakage infers labels for pairs of words in the test split, w_i, w_j, by exploiting labels on paths from w_i to w_j in the training split. We conclude that it is safer to split vertices, V, than edges, E.

1 Introduction

Consider graphs, G, where $G = (V, E)$. V is a set of vertices and E is a set of edges. V is typically a set of words/concepts: $V = \{w_1, w_2, ...\}$, and E is a set of triples, $(head, tail, rel)$, where $head, tail \in V$ and $rel \in R$. R is typically a small set of relations. For one benchmark that will be discussed below, WN18, R is a set of 18 relations borrowed from WordNet.

We will use the terms, *dataset*, to refer to data without standardized splits, and the term, *benchmark*, to refer to a dataset plus standardized splits. Thus, for example, we will refer to WordNet as a dataset, and WN18 as a benchmark.

Many benchmarks split G into train, validation and test splits by partitioning E into three sets. This paper will suggest that partitioning on E can lead to leakage in some cases. It is safer to split on V than to split on E. We will refer to splits on E as the *standard construction* of benchmarks on graphs.

For some examples of the standard construction, consider the literature on knowledge graph completion (KGC)[1] (Nguyen, 2017; Wang et al., 2017; Yu et al., 2019) as well as the literature on node2vec (Grover and Leskovec, 2016), which will be discussed in the next two subsections. Both of these literatures use a number of popular benchmarks with standardized splits for train, validation and test.

The training and validation splits are used to learn a model. The model is then evaluated by how well it predicts edges in the test split. If there is leakage across splits, the integrity of these evaluations is seriously undermined.

1.1 Node2Vec

Node2vec inputs a graph, $G = (V, E)$, and output an embedding, a matrix $M \in \mathbf{R}^{V \times K}$, with K hidden dimensions. Nodes that are "close" in G will be "close" in MM^T, though performance on test sets depends on many factors including the choice of algorithms, various hyper-parameters such as K, and many other details. The software package, nodevectors,[2] supports several node2vec algorithms including: ProNE[3] (Zhang et al., 2019) and GraRep[4] (Cao et al., 2015). More examples of graph benchmarks can be found here.[5] These citations and github repositories mention a number of benchmarks that use the standard construction.

1.2 Knowledge Graph Completion (KGC)

Nguyen (2017) mentions a number of popular datasets for KGC research: WordNet (Fellbaum, 1998), YAGO (Suchanek et al., 2007), Freebase (Bollacker et al., 2008), NELL (Carlson et al., 2010), DBpedia (Lehmann et al., 2015). FB15k and WN18 are two popular KGC benchmarks based on Freebase and WordNet, respectively. Both FB15k and WN18 use the standard construction to create splits.

[1] https://github.com/Sujit-O/pykg2vec

[2] https://pypi.org/project/nodevectors/

[3] https://github.com/THUDM/ProNE

[4] https://github.com/benedekrozemberczki/GraRep

[5] http://snap.stanford.edu/node2vec/

Relation	Edges	Reverse	Edges
hypernyms	37,221	hyponyms	37,221
derivationally related forms	31,867		
member meronym	7928	member holonum	7928
has part	5142	part of	5148
synset domain topic of	3335	member of domain topic	3341
instance hypernym	3150	instance hyponym	3150
also see	1396		
verb group	1220		
member of domain region	983	synset domain region of	982
member of domain usage	675	synset domain usage of	669
similar to	86		

Table 1: 18 Relations in WN18

Both FB15k and WN18 are known to suffer from leakage. As discussed in §4 of (Nguyen, 2017), Toutanova and Chen (2015) and Dettmers et al. (2018) observed leakage involving reversible triples such as:

- feline hyponym cat
- cat hypernym feline

As this example illustrates, links in WordNet usually appear in pairs. The pairs are easy to derive from one another. Therefore, if one member of the pair should appear in one split, and the other member of the pair should appear in another split, as is the case for the feline/cat triples above, then there is leakage between the splits, undermining the integrity of evaluations based on the benchmark.

Note that there are many more triples like the feline/cat, as shown in Table 1. All of the edges on the right hand side of Table 1 are redundant, and many of these redundant pairs are split across splits, introducing leaks that undermine evaluations based on WN18.

To address this kind of leakage, FB15k has been replaced with FB15k-237[6] and WN18 has been replaced with WN18RR.[7]

The WN18RR construction addresses much of the leakage, but not all of it, by removing the duplicated links on the right hand side of Table 1.

Unfortunately, this construction does not remove leakage involving derivationally related links, as discussed in Table 4 of (Church and Bian, 2021).

[6] https://paperswithcode.com/dataset/fb15k

[7] https://paperswithcode.com/dataset/wn18

Dataset	Splits		
	train	val	test
adj	5562	398	1986
noun	2836	206	1020
verb	2534	182	908
fallows	58,494	7190	7366
fallows-s	5886	753	777

Table 2: Sizes (edges) of synonym-antonym datasets

These derivationally related links also come in pairs, but in this case, both the forward link and the reverse link are expressed with the same relation, and therefore, the WN18RR construction does not address this leakage. Thus, while WN18RR does not leak as badly as WN18, there are serious leaks in both benchmarks.

1.3 Synonym/Antonym Classification

Leakage can also be found in other benchmarks that use the standard construction. Consider the synonym-antonym task discussed in (Nguyen et al., 2017). The task is to input a pair of words and output a binary label: 0 (synonym) or 1 (antonym). Sizes of the splits are shown in Table 2.

The first three benchmarks can be downloaded from the supplemental materials of (Xie and Zeng, 2021).[8] Fallows is based on an online thesaurus (Fallows, 1898).[9] Fallows-s is a random sample of the edges in Fallows. Splits for Fallows and Fallows-s will be posted on github. The standard construction was used to create these splits.

The next section will discuss leakage in the benchmarks in Table 2.

2 Paths and Leaks Across Splits

The splits can be viewed as sparse graphs, as shown in Table 3. For comparison sake, SimLex[10] (Hill et al., 2015) and NRC-VAD[11] (Mohammad, 2018) are also shown in Table 3.

NRC-VAD is much larger than the other graphs in Table 3, both in terms of words ($V = \{w_1, w_2, ...\}$), but especially in terms of relations on words ($E = \{(head, tail, rel) | head, tail \in$

[8] https://aclanthology.org/2021.acl-short.71/

[9] https://www.gutenberg.org/files/51155/51155-0.txt

[10] https://aclweb.org/aclwiki/SimLex-999_(State_of_the_art)

[11] https://saifmohammad.com/WebPages/nrc-vad.html

training set	V	E	CC
adj	3315	5562	285
noun	3654	2836	1204
verb	1859	2534	199
fallows	15,466	58,494	32
fallows-s	6326	5886	907
SimLex	1028	999	151
NRC-VAD	20,007	$20,007^2$	1

Table 3: Vertices (V), edges (E) and connected components (CC) in training sets. Graphs are sparse: $E \ll V^2$. SimLex-999 and NRC-VAD are shown for comparison.

Path Length	**adj**	**noun**	**verb**	**fallows**
0				2
1	99	59	60	946
2	80	7	15	3835
3	59	3	7	1156
4+	70	2	35	639
NA	90	135	65	612
total	398	206	182	7190

Table 4: For most pairs of words in the validation set, w_1 and w_2, there is a short path from w_1 to w_2 based on edges in the training set. Path lengths were computed with SciPy (Virtanen et al., 2020), selecting options for undirected graphs.

$V, rel \in R\}$). NRC-VAD is also a dense (fully-connected) graph, unlike the other rows which are sparse graphs with $E \ll V^2$ with more connected components. Note that NRC-VAD has a single connected component, whereas the others have many more than just a single connected component.

In general, the standard construction tends to split connected components. Note that fallows-s has many more connected components than fallows. Since fallows-s is a random sample of edges in fallows, the fact that fallows-s has more connected components than fallows illustrates the tendency for the standard construction to cut connected components into multiple components.

Cutting connected components in this way introduces a risk of leakage. When parts of a component end up in one split, and the rest ends up in other splits, there is a risk that information could leak from one split to another if there are clues left behind providing hints about how to reconstruct the connected component.

Table 4 suggests that path lengths provide hints for reconstructing components. Consider the 398 edges, $E = (w_i, w_j)$, in the validation set for adj. Table 4 reports that 99 of these 398 edges have a path of length 1 using edges from the training set. There are another 80 of 398 with a path of length 2. All but 90 of 398 are part of a connected component in the training set.

When an edge in one split is part of a connected component in another split, it is likely that the label on the edge can be inferred from the labels associated with the component in the other split. In this way, it is likely that information is leaking across splits, when edges are randomly assigned to splits under the standard construction.

Consider the 99 edges of length 1. These are particularly worrisome. There are 99 pairs like *good* and *awful*, where the same edge is in both train and validation, but in reverse directions. This pair is clearly leaking information between the training split and validation split.

Edges of length 2 are not leaking as badly as edges of length 1, but we are concerned about them. Some examples from adj of length 2 paths are: *innocent* $\rightarrow$ *harmless* (via *harmful*), *fresh* $\rightarrow$ *old* (via *aged*), *dead* $\rightarrow$ *deceased* (via *alive*).

How can we exploit these paths to leak labels across splits? Let A be the number of antonym labels on a path from w_i to w_j. A is computed based on edges in the training set. For the purposes of computing A, edges in the training set will be treated as undirected edges.

The decision trees in Figure 1 show how a machine learning system can exploit the leakage. These trees were created with rpart.[12] These trees suggest that an edge in a held out split (test/validation) is likely to be antonymous iff A is odd. We will refer to this heuristic as A-*leakage*. Table 5 reports considerable A-leakage.

There are 4 trees in Figure 1. The two trees on the left fit: $gold \sim A$ for two datasets: fallows and adj. Based on these two trees, we obtained the simpler trees on the right by fitting: $gold \sim A + A.odd$. Decision trees learn the simple rule, leakage depends on the parity of A. It is not necessary to know the exact value of A. The parity is more than sufficient to predict many of the labels in the validation and test splits based on the labels in the training split.

[12] https://www.rdocumentation.org/packages/rpart/versions/4.1-15/topics/rpart

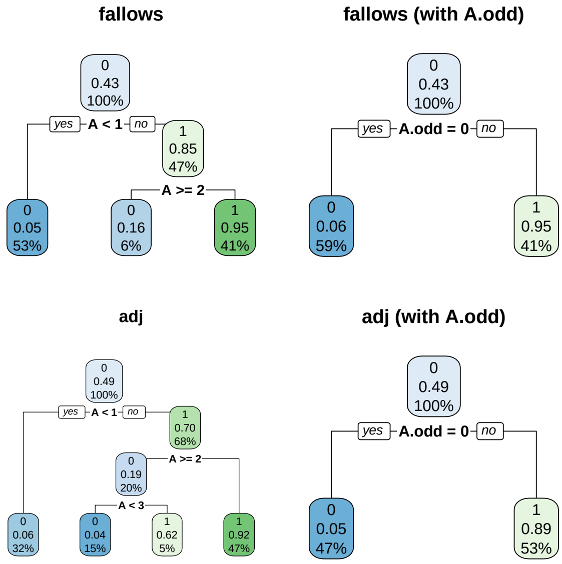

Figure 1: A-*leakage*: Decision trees learn to classify pairs as antonyms iff A is odd. There are three numbers associated with each subtree: (A) a label (1/0), (B) $Pr(1)$ and (C) coverage. By construction, at each level in the tree, coverage sums to 1. (Decision trees were computed using the rpart package in R.)

	Validation		**Test**	
	Acc.	**Applic.**	**Acc**	**Applic.**
adj	0.916	308/398	0.906	1482/1986
noun	0.930	72/206	0.983	302/1020
verb	0.872	118/182	0.882	587/908
fallows	0.945	6576/7190	0.949	6722/7366
fallows-s	0.683	223/753	0.694	241/777

Table 5: Evaluation of A-leakage heuristic which predicts an edge should be labeled as antonym iff A is odd. Accuracies (acc) are computed over applicable edges in Validation and Test sets. Edges in the validation/test set, (w_i, w_j), are applicable if there is a path from w_i to w_j based on (undirected) edges in the training set. A counts the number of antonym labels on this path. Denominators are borrowed from Table 2.

3 Conclusions

This paper discussed leakage in a number of popular benchmarks on graphs. Leakage has been previously reported for a number of benchmarks such as WN18 and FP15k. There have been attempts to remedy these leaks by replacing WN18 with WN18RR, and replacing FP15k with FP15k-237. However, flaws with these remedies have also been previously reported.

This paper reported some novel leaks in benchmarks for synonym-antonym classification. We introduced a novel exploit, A-leakage, that counts, A, the number of antonym labels on paths in the training set. The proposed heuristic infers labels on edges in the test set, (w_i, w_j), from A on paths connecting w_i to w_j in the training set. We found that this heuristic is much better than chance for a number of benchmarks that have been used in the literature for synonym-antonym classification, and therefore, information is leaking in ways that seriously undermine the integrity of those evaluations. In addition, we introduced a novel benchmark based on (Fallows, 1898), and found that that benchmark is also leaking.

What should we do about all this leakage? First, it may be necessary to retract papers based on flawed evaluations. After that, we might attempt to plug each new leak with corrections along the lines of WN18RR and FP15k-237. We are concerned, though, that future researchers will keep finding new exploits. We may not be able to keep up, if exploits are discovered faster than we can deploy remedies. Perhaps, it would be safer and more expedient to split graphs on V than to split on E using the standard construction.

In related work, we decided to stop working on synonym-antonym classification and focus on VAD regression instead. The two tasks are similar, but as shown in Table 2, NRC-VAD is larger and more connected, making it easier to compare and contrast different sampling methods, including sampling on V as opposed to sampling on E.

References

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.

Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM international on conference on information and knowledge management*, pages 891–900.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka, and Tom M Mitchell. 2010. Toward an architecture for never-ending language learning. In *Twenty-Fourth AAAI conference on artificial intelligence*.

Kenneth Church and Yuchen Bian. 2021. Data collection vs. knowledge graph completion: What is needed to improve coverage? In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6210–6215, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Samuel Fallows. 1898. *A Complete Dictionary of Synonyms and Antonyms or, Synonyms and Words of Opposite Meaning*. Good Press.

Christiane Fellbaum, editor. 1998. *WordNet: An electronic lexical database*. MIT press.

Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. SimLex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. 2015. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic web*, 6(2):167–195.

Saif Mohammad. 2018. Obtaining reliable human ratings of valence, arousal, and dominance for 20,000 English words. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 174–184, Melbourne, Australia. Association for Computational Linguistics.

Dat Quoc Nguyen. 2017. An overview of embedding models of entities and relationships for knowledge base completion. *arXiv preprint arXiv:1703.08098*.

Kim Anh Nguyen, Sabine Schulte im Walde, and Ngoc Thang Vu. 2017. Distinguishing antonyms and synonyms in a pattern-based neural network. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 76–85, Valencia, Spain. Association for Computational Linguistics.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706.

Kristina Toutanova and Danqi Chen. 2015. 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.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272.

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

Zhipeng Xie and Nan Zeng. 2021. A mixture-of-experts model for antonym-synonym discrimination. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 558–564, Online. Association for Computational Linguistics.

Shih Yuan Yu, Sujit Rokka Chhetri, Arquimedes Canedo, Palash Goyal, and Mohammad Abdullah Al Faruque. 2019. Pykg2vec: A python library for knowledge graph embedding. *arXiv preprint arXiv:1906.04239*.

Jie Zhang, Yuxiao Dong, Yan Wang, Jie Tang, and Ming Ding. 2019. ProNE: Fast and scalable network representation learning. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 4278–4284. International Joint Conferences on Artificial Intelligence Organization.