APPENDICES

A PROOF OF BINDER'S CONVERGENCE TO LOCAL OPTIMAL SOLUTION

BINDER's algorithm literally solves a combinatorial satisfiability task, a known NP-complete problem. By computing gradient and then utilizing the gradient in deciding bit flipping probability, it works as a gradient descent scheme in the combinatorial space to minimize the objective function defined in Equation 4. However, note that the bit flipping probability decreases gradually as the number of violated constraints decreases with each subsequent epochs. This gives BINDER's learning algorithm a flavor of local search with simulated annealing. The following theorems and lemmas are provided to establish the claim of BINDER's local optimality.

Lemma 1. *When bias $b_\ell = 0$, for any word a, if the j'th bit in the binary representation vector $\mathbf{a}$ is updated by* BINDER*'s probabilistic flipping (keeping the remaining bits the same), the loss function value decreases in the successive iteration.*

Proof. Let S be the set of positive data instances (a, b) where the first entity is the given a. θ is a d-dimensional embedding vector of a and $L(\theta)$ is the component of loss function associated with a. Suppose in an iteration, we probabilistically flip bit j of θ. To compute this probability, BINDER computes the $\Delta_{\mathbf{a}_j} Loss_P$, which is $L(\theta) - L(\theta'_j)$, where θ'_j is the same as θ except that the bit value of j'th position is different. (Recall that we define our gradient to be *positive* when flipping bit j improves our model, thus *decreasing* the loss function.) Based on Eq. 5, this gradient value is $+1$ only for the case when a constraint $\mathbf{a}_j \to \mathbf{b}_j$ is violated (where b is the other element in a training pair) i.e. $\mathbf{a}_j = 0$, but $\mathbf{b}_j = 1$ (see the 3rd column of Table II). Using Line 7 of Algorithm 2 for $b_\ell = 0$, this yields a positive flip probability ($tanh$ is asymmetric function), and with the flip the loss function value decreases by $K\alpha$ (through Eq. 2), where $0 \le K \le |S|$; here K is the number of pairs in S that violate implication constraint with a in the left side. For the other three choices of $\mathbf{a}_j$ and $\mathbf{b}_j$, $(0, 0), (1, 0), (1, 1)$, the contribution to gradient value is 0 or -1, yielding zero flip probability. In all scenarios, the loss value decreases in the successive iteration. □

Lemma 2. *When bias $b_\ell = 0$, for any word b, if the j'th bit in the binary representation vector of b is updated by* BINDER*'s probabilistic flipping (keeping the remaining bits the same), the loss function value decreases in the successive iteration.*

Proof. The proof is identical to the proof of Lemma 1, except that we use gradient value in Eq. 6 instead of Eq. 5. In this case also when only one position of b's embedding vector is flipped probabilistically, the loss function value decreases. □

Lemma 3. *When bias $b_\ell = 0$, given a collection of negative data instances, say, (a', b'), if the j'th bit in the vectors of a' or b' independently (not simultaneously) is updated by* BINDER*'s probabilistic flipping (keeping the remaining bits same), the loss function value decreases or remains the same in the successive iteration.*

Proof. The proof is identical to proof of Lemma 1, except that we use gradient value in Eq. 7 (or Eq. 9) for the case of a', and gradient value of Eq. 8 (or Eq. 10) for b', and the loss function value decreases through Eq. 3. □

These proofs also apply if $r_\ell\alpha \ge b_\ell > 0$ and $r_\ell\beta \ge b_\ell > 0$. In that case, we can flip a bit with zero gradient. Such flips do not immediately increase or decrease the loss function; however, they can allow BINDER to improve from a weak local optimum. In our experiments, $r_\ell\alpha$ and $r_\ell\beta$ are much larger than b_ℓ, so our algorithm prioritizes actual improvements over zero-gradient steps.

Theorem 4. *When bias $b_l = 0$, if Line 8 of Algorithm 2 is executed sequentially for each index value j for each of the entities,* BINDER *reaches a local optimal solution considering a 1-hamming distance neighborhood.*

Proof. Using earlier Lemmas, each bit flipping in any embedding vector of any of the entities, either decreases the loss function or keeps it the same. When Line 8 of Algorithm 2 is executed sequentially for each index j (only one change in one iteration) for each of the entities, the loss function value monotonically decreases in each successive iteration, At a local optimal point, none of the single bit flip improves the value of loss function. Now, if the bias $b_l = 0$, for each entity, the probability of bit-flipping for each index is computed to be 0 (by Line 7 in Algorithm 2), embedding of none of the entities changes any further and BINDER reaches a local optimal solution considering a 1-hamming distance neighborhood. In other words, for an entity a, considering that all the entity embedding is fixed, if we change any single bit of a's embedding, the original embedding of a is guaranteed to be at least as good as the changed embedding. □

When we change only one bit at a time keeping everything else the same (as in the proof), our optimization algorithm becomes a greedy hill climbing algorithm. However, this would make BINDER extremely slow to converge, and it may get stuck in a bad local optimal solution. Thus, we allow all bits to change simultaneously, so it behaves like gradient descent: Suppose θ is an embedding vector of an entity and $L(\theta)$ is the component of loss function associated with this entity. For minimizing $L(\theta)$, at each iteration, a hill climbing method would adjust a single element in θ to decrease $L(\theta)$; on the other hand, gradient descent will adjust all values in θ in each iteration by using $\theta^{new} = \theta^{old} - \alpha\Delta_\theta L(\theta^{old})$. During early iterations, BINDER works like gradient descent, but as iteration progresses, it behaves more like a hill climbing method as gradient values for most bit positions decrease, causing fewer bits to flip.

B PSEUDO-CODE

Pseudo-code of BINDER is shown in Algorithm 2. We initialize the embedding matrix with all 0's. The algorithm goes for at most T epochs (for loop in Line 4-17), updating bit vectors of each vocabulary word in each iteration by flipping bits with probability based on gradient computed through Algorithm 1. The balanced-accuracy metric, defined as $\frac{1}{2}\left(\frac{TP}{TP+FN} + \frac{TN}{TN+FP}\right)$, is computed at the end of each epoch and best accuracy is recorded. We exit early if no improvement on validation data is seen over two consecutive windows of ω epochs, for user-specified ω. The overall computational complexity is $O(ndT(|P| + |N|))$, for n words and d dimensions, which is linear in each variable.

Algorithm 1 Gradient Computation

```
Require: Zero-one Embedding Matrix $B$ of size $n \times d$ initialized with all 0; positive Is-A relation set $P = \{(a^i, b^i)\}_{i=1}^{m}$; negative set $N = \{(a'^i, b'^i)\}_{i=1}^{m'}$; positive and negative sample weights $\alpha, \beta$
1:  $\Delta^+ \leftarrow$ zero matrix, same size as $B$
2:  $\Delta^- \leftarrow$ zero matrix, same size as $B$
3:  for $(a, b) \in P$ do                                   ▷ $*$ is element-wise product
4:      $\Delta^+[a,:] \leftarrow \Delta^+[a,:] + B[b,:] * (1 - 2B[a,:])$
5:      $\Delta^+[b,:] \leftarrow \Delta^+[b,:] + (1 - B[a,:]) * (2B[b,:] - 1)$
6:  end for
7:  for $(a', b') \in N$ do
8:      $\mathbf{G} \leftarrow B[b',:] * (1 - B[a',:])$             ▷ "good" bit pairs (a vector)
9:      if $\sum_j \mathbf{G}_j = 0$ then                      ▷ false positive, flip something
10:         $\Delta^-[a',:] \leftarrow \Delta^-[a',:] + B[a',:] * B[b',:]$
11:         $\Delta^-[b',:] \leftarrow \Delta^-[b',:] + (1 - B[a',:]) * (1 - B[b',:])$
12:     else if $\sum_j \mathbf{G}_j = 1$ then                 ▷ close to being wrong, so protect
13:         $\Delta^-[a',:] \leftarrow \Delta^-[a',:] - \mathbf{G}$         ▷ note only one element of $\mathbf{G}$ is 1
14:         $\Delta^-[b',:] \leftarrow \Delta^-[b',:] - \mathbf{G}$
15:     end if
16: end for
17: return $\Delta := \alpha\Delta^+ + \beta\Delta^-$
```

C TRAINING SETUP AND HYPERPARAMETER TUNING

For BINDER, we learn a d-bit array for each concept in the hierarchy. For all tasks, we train BINDER for 10000 epochs, where each epoch considers the full batch for training. We tune our hyperparameters: dimension d, positive and negative sample weights α, β, negative sample multiplier n^-, and the learning rate and bias r_ℓ, b_ℓ manually by running separate experiments for each dataset. We

Algorithm 2 Training Algorithm

```
Require: Word list $W = (w_1, \dots, w_n)$; Dimension $d$; Positive training set $P = \{(a^i, b^i)\}_{i=1}^m$; validation sets $VP, VN$; gradient weights $\alpha, \beta$, learning params $r_\ell, b_\ell$, negative sample multiplier $n^-$ (must be even); maximum epochs $T$, early stop width $\omega$
 1: $B \leftarrow$ zero matrix of size $|W| \times d$
 2: $Acc \leftarrow$ empty list
 3: $(BestEmbedding, BestAcc) \leftarrow (B, 0)$
 4: for $t = 1$ to $T$ do
 5:     $N \leftarrow$ negative samples (Section 2.3)
 6:     $\Delta \leftarrow$ gradient from Algorithm 1
 7:     $X \leftarrow \max\left\{0, \frac{1}{2}\tanh(2(r_\ell \Delta + b_\ell))\right\}$        ▷ flip probabilities
 8:     Flip each bit $B[w, j]$ with (independent) probability $X[w, j]$
 9:     $acc \leftarrow$ BalancedAccuracy(Evaluate$(B, VP, VN)$)
10:     if $acc > BestAcc$ then
11:         $(BestEmbedding, BestAcc) \leftarrow (B, acc)$
12:     end if
13:     Append $acc$ to list $Acc$
14:     if mean(last $2\omega$ elements of $Acc$) $\geq$ mean(last $\omega$ elements of $Acc$) then
15:         Exit Loop        ▷ Early Exit Criterion if no improvement
16:     end if
17: end for
18: return $BestEmbedding$
```

find that the optimal learning rate r_ℓ and learning bias b_ℓ are 0.008 and 0.01 respectively for all data sets and tasks. The learning bias b_ℓ = 0.01 means that bits whose gradient was exactly neutral had a 1% chance of flipping. We fix β at 10 and tuned α; we always find that $\alpha \leq \beta$ gives far too many false negatives. By compressing the expressiveness of the model, we force the binary embeddings to make "decisions" about which attributes to merge, thus increasing its predictive power. For reconstruction task we need more bits to increase the capacity of our model to better reconstruct training edges. Optimal $(\text{bits}, \alpha, n^-)$ for reconstruction task on Medical, Music, Shwartz Lex and Shwartz Random datasets is (50, 30, 32), and for WordNet Nouns dataset it is (80, 15, 8). For link prediction transitive closure experiment on Medical, Music, Shwartz Lex and Shwartz Random datasets with transitive closure 0% and 10%: (100, 25000, 12) and for 25% and 50%: (100, 50000, 12). Optimal $(\text{bits}, \alpha, n^-)$ for link prediction transitive closure task on WordNet Nouns dataset with all transitive closure configurations (0%, 10%, 25%, 50%) is (120, 25000, 12). We use these hyper parameters to obtain results of Table 3, 4 and 5. For the competing models, except T-Box, we exhaustively tuned dimensions $d = 5, 10, 20, 50, 100, 200$ keeping other hyperparmeters similar to original papers. Since T-Box requires 2*d dimension for its representation, to be fair with other models, we tuned T-Box for dimensions $d = 2, 5, 10, 20, 50, 100$ keeping other hyperparmeters similar to original paper implementation. All the models were run on a Tesla V100 GPU.

D CASE STUDY

We present a case-study experiment, which will provide the reader a sketch of BINDER's embedding results. For this we run our model with 8 bits on the toy lattice from Vendrov et al. (2015). Because the lattice is very small, it is possible for BINDER to achieve perfect accuracy. Figure 1 shows the final embedding. Given the embeddings for `boy`, `person`, and `city`, a human can determine that, according to the model, `boy` is-a `person` but not is-a `city`. In theory, each bit can correspond to some "attribute" of each object, where all attributes are passed down to hyponyms. This can help to build an explainable embedding. For instance, the second circle clearly denotes an attribute which could be named *has-life*. Sometimes, however, bits are used with different meanings in different words: the right-most bit is 1 on `man`, `girl`, and `SanJuan`. This is partly because the toy lattice is very sparse, with only two sub-components of `city` compared to ten of `livingThing`, and `adult` and `child` were not included in the lattice, as they are in WordNet.

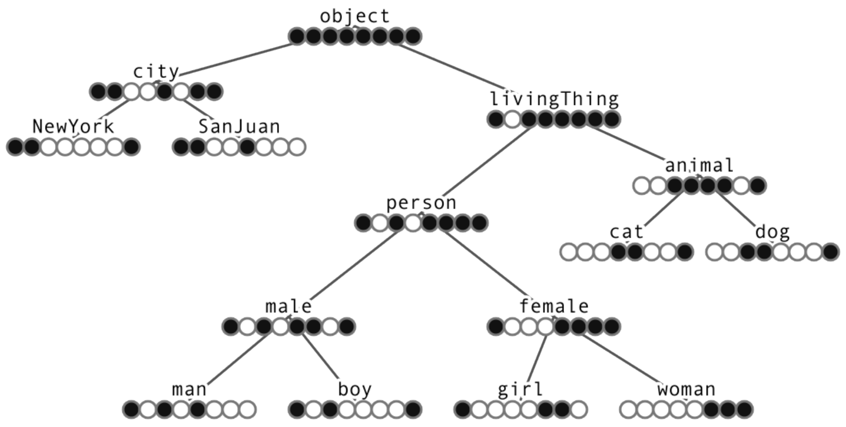

Figure 1: Visual representation of toy dataset results. White circles represent 1 and black circles 0.

Table 6: Edge distribution for all datasets

Dataset	Edge Counts		
	Direct Edge	Transitive (Indirect) Edge	Full Transitive Closure (Direct + Transitive)
Medical	2616	1692	4308
Music	3920	2608	6528
Shwartz Lex	5566	7940	13506
Shwartz Random	13740	42437	56177
WordNet Nouns	84363	576764	661127

E Dataset Statistics

We have explained before that since BINDER does not rely on transitive edges for link prediction, unlike its competitors, BINDER performance is superior compared to its competitors at 0% transitive closure. The difference margin with the competitors are larger for large datasets where transitive edge percentages are significantly higher compared to direct edge percentages, as shown in Table 6 and figure 2. Hence it proves the superiority of BINDER embedding.

F More Experimental Results

F.1 Ablation Study

For an ablation study, we observe the effect on model accuracy on the validation data by removing β and b_ℓ separately while keeping other parameters at best value. For Nouns data set on reconstruction task, setting $\beta = 0$ (i.e. ignoring the negative samples) gives accuracy 76% after 500 iterations. If we set b_ℓ to 0 then we don't see any significant effect on accuracy. For the 0% transitive closure prediction task on Nouns data set, when we set $\beta = 0$ accuracy saturates at 86% after 800 iterations.If we set b_ℓ to 0 then we see accuracy drops by several percentages from the best result.

F.2 Justification of Balanced Accuracy metric over F1-score

For classification over imbalance data, F1 measure is a good metric; however some argue (and we agree with them) that balanced accuracy (BA) is actually a better metric, which is defined as the average of positive class $recall_+ = \frac{TP}{TP+FN}$, and negative class $recall_- = \frac{TN}{TN+FP}$. Balanced accuracy is a better metric than F1-score for imbalanced datasets, because F1-score does not care about how many true negatives are being classified. F1-score uses precision and recall, which together use only three entries of the confusion matrix (TP, FP, FN); on the other hand, balanced accuracy uses all four entries of the confusion matrix. For an example, say a dataset has 1000 negative and 10 positive examples. If the model predicts there are 15 positive ($TP = 5, FP = 10$), and predicts the rest as negative ($TN = 990, FN = 5$), we get $Precision = \frac{5}{15} \approx 0.33$, and $Recall = \frac{5}{10} = 0.5$ it yields $F_1 = 0.4$. The model does not get much credit for correctly predicting 990 out of 1000

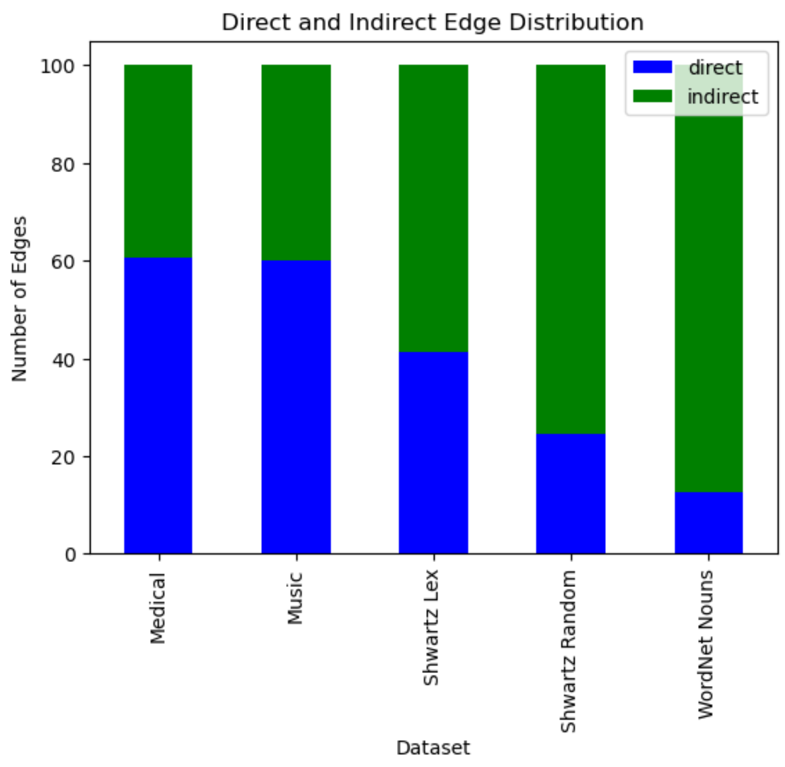

Figure 2: Distribution of Direct and Transitive (Indirect) edges for all datasets.

examples as negative. However, the balanced accuracy is $\frac{1}{2} * (\frac{5}{10} + \frac{990}{1000}) = 0.745$, which provides somewhat a more realistic picture.

Additionally, balanced accuracy does not depend on the ratio of positive to negative edges, which is important because our datasets have different negative edge ratios: Medical has about 1400 entities and 4300 edges, for a negative ratio of about 450, while the ratio for Nouns is about 9000, and evaluating the entire negative edge set is impractical. In practice, we chose a 10:1 ratio of negative to positive test cases, but this choice was arbitrary and made because Nickel & Kiela (2017) used the same ratio. Balanced accuracy is unaffected by these arbitrary decisions, provided the samples are large enough to avoid statistical error.

We actually used an F1 measure with the 10:1 ratio, but found it to be very harsh for the competitors. In the Table below, we show F1-score results for the four transitive link prediction experiments. As can be seen BINDER's F1-scores (bold numbers) are significantly better than all other methods on all datasets. The competitor methods suffer severely due their poor precision. We have reported F1-score for all our experiments in Tables 7, 8 and 9.

F.3 BINDER: RESULTS FURTHER DISCUSSION

For reconstruction task, OE achieves better performance than BINDER on two smaller datasets (Edges: Medical: 4.3k, Music: 6.5k) and achieves equal performance on mid-sized datasets (Edges: Shwartz Lex: 13.5k, Shwartz Random: 56.2k) for fewer dimensions. For smaller datasets, OE achieves 100% accuracy by using $d = 10$ and 20, but BINDER achieves 99.9% accuracy by using 50 bits. Based on this observation, one can conclude that BINDER does not show superiority over OE. However, we argue that BINDER still wins because it uses bits, whereas other methods operates in real number domain. We need at least 4 bytes (32 bits) to represent a real number. So, the claim that OE is using fewer dimension is untrue, because OE is using $d \geq 10$ ($10 \times 32 = 320$ bits), whereas BINDER is using only $d = 50$ bits. If BINDER is allowed only 50 bits, for a fair comparison other methods should be allowed $\lceil 50/32 \rceil = 2$ dimensions. From our experiments,

Table 7: Reconstruction Results F1-score(%) (dim)

Model	**Medical** entities = 1.4k edges = 4.3k	**Music** entities = 1k edges = 6.5k	**Shwartz Lex** entities = 5.8k edges = 13.5k	**Shwartz Random** entities = 13.2k edges = 56.2k	**WordNet Nouns** entities = 82k edges = 743k
OE	**100** (20)	**100** (20)	100 (20)	100 (50)	97.5 (200)
Poincaré	61.0 (100)	45.0 (50)	40.1 (10)	28.6 (100)	97.2 (50)
HEC	87.5 (100)	73.2 (100)	97.7 (20)	88.6 (10)	91.3 (100)
T-Box	100 (25)	100 (50)	100 (25)	100 (50)	99.9 (50)
BINDER	99.9 (50)	99.9 (50)	**100** (50)*	**100** (50)*	**99.9** (80)*

*For Shwartz Lex and Random dataset, BINDER dimension is higher compared to OE but considering space complexity (1 bit vs 4 bytes) for each dimension we conclude BINDER as the wining model.

Table 8: Link Prediction (Transitive Closure) Results F1-score(%) (dim)

Model	**Medical**	**Music**	**Shwartz Lex**	**Shwartz Random**	**Nouns**
Transitive Closure 0%					
OE	83.1 (10)	74.8 (10)	46.7 (10)	42.1(50)	48.9 (20)
Poincaré	44.0 (100)	27.5 (50)	28.8 (5)	31.9 (5)	33.1(200)
HEC	58.3 (100)	38.2 (20)	41.2 (50)	32.3 (5)	39.6 (200)
T-Box	29.8 (50)	29.5 (50)	25.5 (100)	22.8 (50)	25.7 (100)
BINDER	**96.6** (100)	**87.8** (100)	**98.4** (100)	**97.5** (100)	**91.7** (120)
Transitive Closure 10%					
OE	87.3 (100)	81.8 (20)	56.7 (5)	51.6 (5)	62.1 (5)
Poincaré	55.2 (50)	30.0 (10)	22.6 (5)	27.8 (200)	35.6 (10)
HEC	71.5 (50)	51.6 (50)	81.1 (200)	66.8 (50)	87.2 (200)
T-Box	38.7 (100)	35.2 (100)	28.3 (100)	28.5 (100)	35.6 (100)
BINDER	**99.4** (100)	**83.9** (100)	**100** (100)	**99.9** (100)	**99.2** (120)
Transitive Closure 25%					
OE	87.3 (10)	83.2 (20)	55.9 (10)	51.1 (10)	71.2 (10)
Poincaré	54.4 (10)	33.2 (20)	22.7 (20)	23.4 (10)	39.9 (50)
HEC	78.9 (100)	58.4 (5)	85.9 (20)	74.1 (50)	91.2 (200)
T-Box	39.6 (100)	35.4 (100)	28.4 (100)	28.5 (100)	45.6 (100)
BINDER	**99.1** (100)	**87.0** (100)	**100** (100)	**99.9** (100)	**98.5** (120)
Transitive Closure 50%					
OE	92.0 (50)	87.9 (10)	53.3 (5)	61.6 (10)	80.7 (10)
Poincaré	62.1 (50)	32.4 (200)	24.4 (100)	23.6 (20)	43.0 (200)
HEC	87.2 (200)	68.9 (5)	83.1 (50)	77.3 (100)	95.3 (50)
T-Box	48.8 (100)	42.5 (100)	28.1 (100)	28.5 (50)	57.9 (100)
BINDER	**99.7** (100)	**90.0** (100)	**100** (100)	**99.9** (100)	**99.6** (120)

we observed that even with $d = 5$ ($5 \times 32 = 160$ bits) and $d = 10$ (320 bits), OE achieves 99.4% and 98.6% accuracy respectively for Medical and Music, which is poorer than BINDER. Most importantly, on the largest dataset (Nouns), which has 100 times more edges than the smaller datasets, BINDER achieves 99.7% accuracy with 100 bits, whereas OE achieves 96.7% accuracy by using 200 dimensions, or $200 \times 32 = 6400$ bits.

To summarize, OE uses the smallest dimension of all the competitors because it uses real-number space which is less constrained, whereas hyperbolic space is more constrained, and box embedding requires two vectors per dimension. Comparing to BINDER, OE actually uses more memory, as OE uses real space and BINDER uses binary vectors.

Table 9: Distribution of BINDER Results F1-score ($\mu \pm \sigma$)%

Task	**Medical**	**Music**	**Shwartz Lex**	**Shwartz Random**	**Nouns**
Recon (100% TC)	99.9 ± 0.03	99.8 ± 0.04	99.9 ± 0.11	99.9 ± 0.07	99.8 ± 0.04
Pred (0% TC)	93.9 ± 2.4	78.2 ± 5.5	97.7 ± 0.5	96.9 ± 0.3	98.1 ± 0.2
Pred (10% TC)	93.8 ± 2.5	77.4 ± 3.7	99.4 ± 0.3	99.1 ± 0.1	97.8 ± 1.1
Pred (25% TC)	94.1 ± 0.2	79.6 ± 5.8	99.4 ± 0.3	99.5 ± 0.1	97.7 ± 0.9
Pred (50% TC)	97.1 ± 0.9	78.4 ± 6.5	99.8 ± 0.2	99.6 ± 0.1	99.4 ± 0.3

TC = Transitive Closure

F.4 BINDER: MODEL CONVERGENCE RESULTS

We run our models with a large number of iterations to maximize the accuracy. Although the model attains high accuracy very quickly, it continues to improve steadily for reconstruction task, as shown in the first graph in Figure 3. The 0% transitive closure prediction task accuracy saturates at around 1000 iterations and then start decreasing, as shown in the second graph in Figure 3.

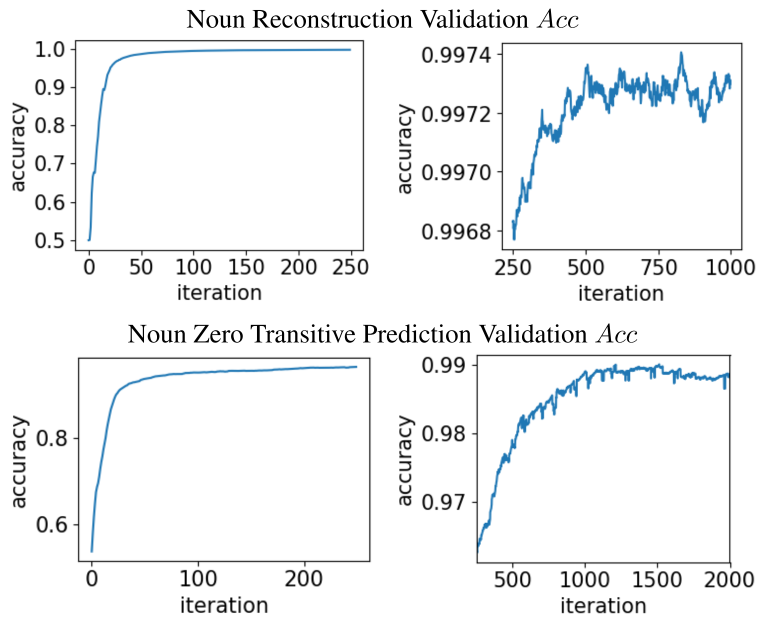

Figure 3: Graph of validation Acc score of the two types of Noun experiments, for the first 250 iterations (left) and the last 750 and 1750 (right).

F.5 BINDER: RESOURCE CONSUMPTION RESULTS

One of the main advantages of using binary vectors is their efficiency compared to floating-point vectors. The algorithms for BINDER are fast since BINDER uses only basic integer arithmetic except for the `tanh` function in the flip probability. On Nouns dataset the Reconstruction task takes 22m 50s for 1000 iterations and the 0% transitive closure task takes 2m 09s for 2000 iterations. Furthermore, the final representation of the concepts using BINDER are binary vectors; the storage required for n words and d dimensions is $\frac{dn}{8}$ bytes instead of Order Embedding's $4dn$ bytes if 32-bit floating point values are used. We want to add the following table where we consider WordNet Nouns dataset and calculate the size of final embedding of different models for d=100 in table 10

Table 10: Space complexity for all models

Model	Storage
OE	34.2 MB
Hyperbolic methods	34.2 MB
T-Box	67.0 MB
BINDER	2.36 MB

G FURTHER JUSTIFICATION OF BINDER'S ALGORITHM

G.1 COMPARISON OF BINDER WITH SPARSE ADJACENCY LIST (SAL)

Sparse adjacency list (SAL) does not provide fixed-size vector embedding of entities, but BINDER's bit-vector representation provides that. SAL does not capture order between entities, but BINDER's bit-vector provides that. SAL only captures the edges of the relation, but BINDER's bit-vector is an order-embedding, which represents nodes as vectors which are transferable to other subsequent knowledge discovery task. For the Noun dataset, which has 743K edges and 82K vertices, BINDER's bit vector will take $82\,\mathrm{kB} \cdot 100/8 = 1025\,\mathrm{kB}$, whereas a sparse adjacency list will take at least $(743\mathrm{k} + 82\mathrm{k}) * 4 = 3300$ Kbytes (considering 4 bytes for integer).

G.2 COMPARISON OF BINDER'S RANDOMIZED ALGORITHM WITH DETERMINISTIC ALGORITHM FOR GENERATING BIT-VECTOR EMBEDDINGS

A deterministic algorithm offers no learning, it is simply memorizing the edges. It can only be used when an entire minimal set of edges of the DAG is given. But, one cannot expect that all the edges between entity pairs are already known/given in the training data. If that is the case, no learning or embedding is needed and a fixed deterministic method can be used.

The advantage of BINDER is that it has learning capability. It assign bit-vectors so that it can infer missing edges. In other words, if we remove some direct edges in the training data, BINDER will still be able to embed entities reasonably. We performed link prediction experiments to prove this claim on Mammals and Nouns dataset; this setup is identical to that of Vendrov et al. (2015). For Mammals dataset (6.5k edges) and Nouns dataset (743k edges) we randomly take out 300 and 3500 edges[2] respectively, which may or may not be direct edges, to construct positive test data. We create negative test data by corrupting positive pairs. Results from this experiment are reported in table 11. For Nouns, Vendrov et al. (2015) reported an accuracy of 90.6% in their work, which is worse than BINDER's 93.9%.

Table 11: Results of Vendrov et al. (2015) style Link Prediction experiment

Model	**Dataset**			
	Mammals		Nouns	
	Acc(%)	F1-score(%)	Acc(%)	F1-score(%)
BINDER	94.2 ± 1.1	94.1 ± 1.1	93.9 ± 0.4	93.6 ± 0.5

[2]Vendrov et al. (2015) removed 4000 edges from their Nouns dataset. However, they appear to have used a different version of WordNet Nouns with about 838,000 edges, and so we remove fewer edges to maintain approximate proportion.