Invertible Tree Representations Using a Cryptographic Role Embedding Scheme

Coleman Haley[1] and **Paul Smolensky**[1, 2]
[1]Johns Hopkins University [2]Microsoft Research AI

Introduction

Deep Learning struggles with *compositional generalization* relative to traditional methods in symbolic AI. Such generalizations can be facilitated by the use of structured representations. One method proposed for representing structure while maintaining the parallel and distributed nature of neural/vector space computation is the Tensor Product Representation (TPR). TPRs express structure in terms of variable binding of fillers to roles. In the case of a tree, such a role might be `left-child-of-right-child-of-root`, with the filler as the label of that node.

Tensor Product Representations

Let a symbolic structure be decomposed into N fillers $\{f_i\}$ and roles $\{r_i\}$. To form a TPR, we first embed all roles and fillers into vector spaces of dimensions n and d respectively using the injective functions emb_r and emb_f resulting TPR T an then be expressed as follows:

$$T = \sum_{i=1}^{N} \text{emb}_f(f_i) \otimes \text{emb}_r(r_i) \in \mathbb{R}^d \otimes \mathbb{R}^n.$$

If emb_r distributes the role vectors in $\mathbb{R}^n$ such that roles that occur in the same structure are orthogonal to one another, then we can retrieve the filler f_i of r_i by taking $\tilde{f}_i = T \cdot \text{emb}_r(r_i)$. We call this ***unbinding*** r_i. The result of unbinding for r_j not in the structure will be the zero vector. If the role embeddings are not orthogonal, then the fillers of other roles will *intrude* on the result of unbinding proportional to their angle with $\text{emb}_r(r_i)$. If the cumulative intrusion is small enough and we know the embeddings of all possible fillers, then we may retrieve the correct filler by finding the filler embedding with the highest cosine similarity to the result of unbinding, allowing *inversion* of the TPR, or reconstruction of the original symbol structure.

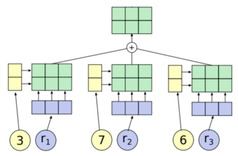


Figure 1: Example of TPR formation. *Role vectors* (in blue) and their corrresponding *filler vectors* are combined via the tensor product and summed.

As the dimension of a vector space goes to infinity, random vectors in that space trend towards orthogonality. While the number of exactly orthogonal vectors in a space is limited by the dimensionality of the space, in sufficiently high-dimensional spaces **it may be possible to approximate orthogonality with random vectors.** In such a case, we may be able to have more bindings in a TPR than dimensions in the role space while retaining accuracy. In this work we look at the properties of emb_r that use this property.

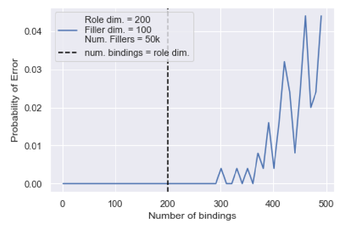
Some simple cases

Let γ be a constant such that for role-embedding dimension γ, structures with up to γn bindings can be inverted with error $< 1\%$.

Fully random TPRs – Figure 2(a) ($\gamma = 2$):
- Fillers and roles randomly selected, embedded randomly
- Mean intrusion is 0 (destructive interference)
- Lower bound on error

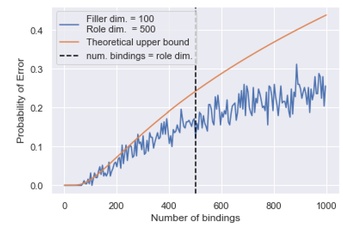
Sentence TPRs – Figure 2(b) ($\gamma = 2$):
- Roles are linear position, fillers are sentences from Reuters news corpus.
- Role vectors are random, filler vectors are GoogleNews embeddings
- Inter-filler and Inter-role correlations
- Constructive interference from Zipfian distribution of words

Maximal intrusion – Figure 2(c):
- All roles except the one being unbound are bound to a single filler
- Constructive interference, upper bound on error
- Error proven to be exponentially decreasing in the role dimension



(a) Random TPRs

(b) word2vec sentences



(c) Maximal intrusion

Figure 2: Error for simple random TPR cases.

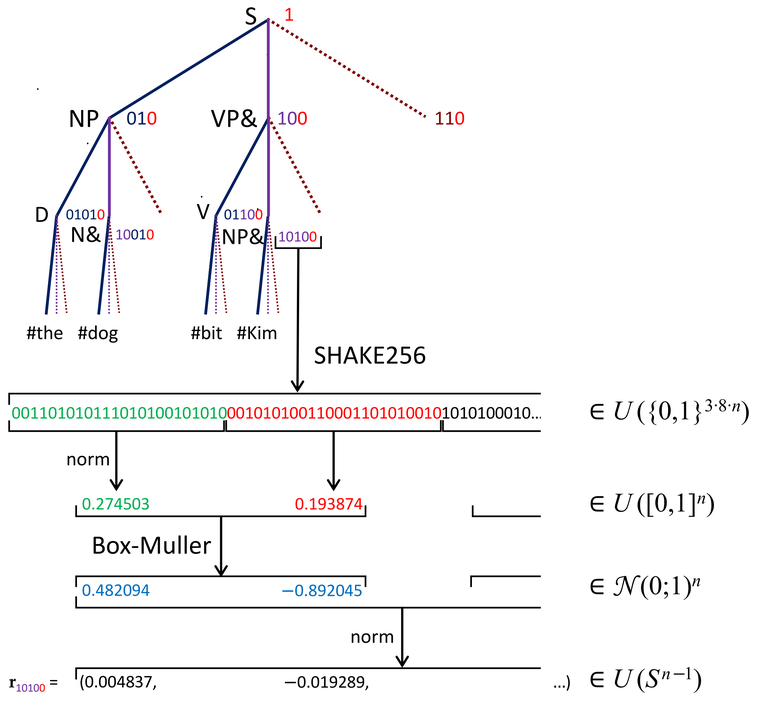


Figure 3: example of **Cryptographic Role Embedding** for ternary branching trees.

Cryptographic Roles

Because there are exponentially many possible positions in a tree, we cannot rely on memoizing random vectors for roles. Thus, we need a deterministic way to map tree positions to pseudorandom unit vectors. To do this, we rely on a five-step process (for roles of dimension n):

1. **Label** the positions in the tree with a bit string
2. **Apply the SHAKE256 cryptographic hash** to generate $3n$'pseudorandom bytes
3. **Norm** 3-byte chunks to obtain random uniform samples on $[0, 1)$
4. **Use Box-Muller** transform to turn these random samples into random samples from the standard normal distribution.
5. **Vector-norm** the concatenation of these bits.

This results in a pseudorandom unit vector which is fully determined by the input position. The cryptographic hash function plays the key role of approximating randomness while allowing a deterministic mapping. This process is fully general, allowing the representation of arbitrary role schemes that can be encoded as strings. Figure 3 provides an overview.

Experiments

Cryptographic roles allow the creation of exponentially many roles with low average dot products, which allows us to abandon prior methods of recursive binding for tree TPRs and use a simple matrix TPR of fixed dimension. Trees from the MASC corpus are used, and the label of each position in the tree is bound to that position. The largest 1% of trees were omitted, resulting in 35,379 trees, with a maximum of 183 nodes. The widest node has 98-ary branching and the deepest tree has depth 29, so in contrast a maximal complete tree would have approximately 4×10^{85} nodes.

Because which roles are bound are not known at unbinding time, unbinding must proceed via breadth first search. The labels of leaf nodes and rightmost children were annotated with a special symbol indicating that unbinding should stop there. When this is unbound incorrectly, it can lead to exponential error, making tree reconstruction particularly challenging.

Because the reconstructed tree and the correct tree may not contain the same set of nodes, F1 is used as accuracy metric. Figure 4 shows the maximum number of nodes for which the F1 is at least .99 when unbinding trees from MASC, with the filler and role dimension varied. Role dimension seems to have a stronger effect than filler dimension (likely due to the orthogonality requirements). For sufficiently large filler dimension, γ ranges from 0.69 to 1, similar to truly orthogonal vectors. We find for trees of size < 150, we may use role dimension 200 and filler dimension 150 to represent them with a 30,000 dimensional TPR. Previous methods relying on orthogonality and recursion would require 8.6×10^{57}-dimensional roles.

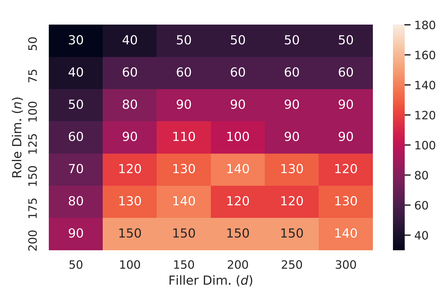


Figure 4: The largest number of nodes (k) for which a given filler and role dimension combination has error $< 1\%$ for all smaller tree sizes.