Invertible Tree Embeddings using a Cryptographic Role Embedding Scheme

Coleman Haley[1] and **Paul Smolensky**[1, 2]
[1]Department of Cognitive Science, Johns Hopkins University, Baltimore, MD
[2]Microsoft Research AI, Redmond, WA
{chaley7, smolensky}@jhu.edu

Abstract

We present a novel method for embedding trees in a vector space based on Tensor-Product Representations (TPRs) which allows for inversion: the retrieval of the original tree structure and nodes from the vectorial embedding. Unlike previous attempts, this does not come at the cost of intractable representation size; we utilize a method for non-exact inversion, showing that it works well when there is sufficient randomness in the representation scheme for simple data and providing an upper bound on its error. To handle the huge number of possible tree positions without memoizing position representation vectors, we present a method (Cryptographic Role Embedding) using cryptographic hashing algorithms that allows for the representation of unboundedly many positions. Through experiments on parse tree data, we show a 30,000-dimensional Cryptographic Role Embedding of trees can provide invertibility with error $< 1\%$ that previous methods would require 8.6×10^{57} dimensions to represent.

1 Introduction: Compositional structure in neural representations

One thing that critics and advocates of deep learning (DL) agree on is that DL models would benefit greatly from stronger compositional generalization: the ability to freely generalize, on relatively scant training data, to novel combinations of familiar constituents (Lake and Baroni, 2017; Marcus, 2020; Russin et al., 2020). Symbolic systems achieve strong compositional generalization — especially in language — by leveraging compositional structure within the model's internal representations themselves (Frege and Beaney, 1997). While it is conceivable that DL models in NLP will achieve strong compositional generalization without compositional representations (Baroni, 2020), we believe it is important to investigate neural instantiations of the one method known to underlie such generalization: structured representations. But how can compositional structure be encoded without sacrificing the hallmarks — and sources of power — of neural computation: distributed representation (as opposed to physically separate encodings of constituents) and fully parallel processing (as opposed to operating sequentially on separate constituents)?

The question of distributed neural representations with compositional structure has been debated for decades. That such representations are even possible, claims to the contrary notwithstanding (Fodor and Pylyshyn, 1988), was established early on, by several techniques (Pollack, 1989; Smolensky, 1990; Shastri and Ajjanagadde, 1993; Plate, 1995, a.o.). It turns out that all these techniques are, up to elementwise nonlinear squashing, special cases of a general technique: tensor product representations (TPRs) (Smolensky and Tesar, 2006). This suggests that TPRs may be a universal form of compositional distributed representation, raising the question of whether standard DL models learn them spontaneously. This has recently shown to be true, up to an affine transformation, for simple, highly compositional functions and extensive training data (McCoy et al., 2019; Soulos et al., 2019). Alongside this provocative suggestion that TPRs may be necessary for compositional encoding in neural nets, there are formal demonstrations of the sufficiency of TPRs to enable NNs to compute recursive symbolic functions — strongly compositional behavior (Smolensky, 2012). These representations are typically fully distributed

Proceedings of the 28th International Conference on Computational Linguistics, pages 3671–3683
Barcelona, Spain (Online), December 8-13, 2020

(each neuron contributes to the encoding of each constituent) and the function computation is fully parallel (for many complex functions, a single step).

TPRs introduce bona fide structure to neural representations via the neural embedding of **roles** that define a particular type of compositional structure. For binary trees, one such role could be left-child-of-right-child-of-root. The fillers of these roles, whether they are atomic or structures themselves, are bound to their respective roles by taking the tensor product of the neural embedding of each filler and the embedding of its structural role. The sum of these tensor-product filler-role bindings is the TPR for the whole structure.

When the embeddings of the roles are orthonormal vectors, the filler of any given role r with embedding $\mathbf{r}$ can be exactly recovered from the embedding $\mathbf{S}$ of the structure as a whole simply by taking the inner product: $\mathbf{S} \cdot \mathbf{r}$ (see Sec. 2.2, which also shows that the same invertibility property holds if the role embeddings are merely linearly independent.) This **unbinding** process can lead to an error if another role has an embedding $\mathbf{r}'$ that is not orthogonal to $\mathbf{r}$: unbinding r will induce an *intrusion* of the filler of r' with a magnitude proportional to the inner product $\mathbf{r}' \cdot \mathbf{r}$.

Orthogonality of the role embeddings is of course possible only if their dimension is at least as large as the number of possible roles. Thus for a k-branching tree of depth δ, the size of the TPR is proportional to k^δ. This contrasts with the size of a corresponding symbolic representation, which is proportional to the **occupancy** of the tree: the number of roles with non-null fillers. In NLP, where parse trees are often relatively sparse, this can be a large difference.

The source of this disparity is that, unlike symbolic representations, distributed neural representations must pre-allocate space for all possible items that may need to be encoded: here, a dimension of the TPR for each possible atomic-filler/role pair, which is required for exact invertibility.

However, we know that, in high-dimensional vector spaces, randomly chosen vectors are most likely to be approximately orthogonal (see Sec. 2.3). The work presented here investigates the **Occupancy-Scaling Hypothesis**: *in high-dimensional embedding spaces, TPRs for trees can be invertible to a good approximation provided the occupancy k of the tree is smaller than γ times the role-embedding dimension n, $k < \gamma n$, where $\gamma = O(1)$.* The hypothesis will hold if, under such conditions, role vectors can be assigned so that, with sufficiently high probability, pairs of occupied roles have embedding vectors that are sufficiently close to orthogonal: then intrusion will not lead to unbinding errors. If true, this means that TPRs scale like their symbolic counterparts: the number of *possible* roles is irrelevant; only the number of *occupied* roles matters.

The contributions of the paper consist in a sequence of tests of the Occupancy-Scaling Hypothesis in settings of varying difficulty. In Sec. 3, we first investigate the hypothesis in the case of random symbol strings, with random embeddings both of the roles and of the symbols that fill them. We next examine the case of natural-language sentences, considered as strings of word tokens: now the fillers are not random, although the filled roles are still predictable (positions 1 through N = length of the sentence). We then present experimental verification of a new theoretical worst-case bound on unbinding error (Appendix B).

The main contributions of the paper lie in Sec. 4, where we consider encodings of constituency-parsed sentences in which the roles filled in the parse tree are variable across sentences and the number of possible roles (tree positions) is essentially infinite. The primary innovation of the paper, the **Cryptographic Role Embedding** technique, is shown to effectively encode a huge number of role vectors in a fixed embedding dimension.

Our results show that, even when embeddings of symbols are highly compressed and an unbounded set of structural positions are embedded in a modest-sized space, the Occupancy-Scaling Hypothesis holds with a scaling coefficient particular to the type of data being represented. For example, in Sec. 4.3 we show that where exact invertibility would require an embedding dimension of 8.6×10^{57}, a 30,000-dimensional Cryptographic Role Embedding of trees can provide invertibility with error $< 1\%$.

2 Background

2.1 Tensor Product Representations (TPRs)

TPRs provide a principled way of representing information with compositional structure in vector spaces, such as those used as the input and output domains of neural networks (Smolensky, 1990). Developing

a tensor-product-based representational scheme begins by decomposing a compositional structure into structural *roles*, which define a structural type (Newell, 1980); a string can be defined by roles first-position, second-position, etc., and a tree by root, first-child-of-root, etc. A particular instance of the structural type is defined by assigning *fillers* to (some of) these roles. For a specific string, first-position might be filled by `Kim`; for a tree, first-child-of-first-child-of-root might be filled by `the`. A compositional structure can then be represented as the *bindings* of fillers to roles. Once decomposed, roles and fillers are embedded into their respective representational vector spaces. Let some information (e.g., a sentence) be encoded as a particular instance S of a structural type defined by a set of indexed roles $\{r_j\}$, and let the possible fillers constitute an indexed set $\{f_i\}$. Now let b_S be a list of ordered pairs (i, j) representing that in S, the filler with index i (embedded as vector $\hat{\mathbf{f}}_i$) is bound to the role with index j (embedded as vector $\hat{\mathbf{r}}_j$). The *tensor product representation* (TPR) $\mathbf{T}$ of S is then given by

$$\mathbf{T} = \sum_{(i,j)\in b_S} \hat{\mathbf{f}}_i \otimes \hat{\mathbf{r}}_j \tag{1}$$

In certain settings, this TPR may itself be used as a filler and subsequently be bound to another role vector (Legendre et al., 1991). This process results in a TPR that represents hierarchical compositional structure. Here we adopt a setting in which the filler of each role is an atom (e.g., a word), and hierarchical structure, e.g. of a tree, is encoded in the roles themselves, which include embedded roles such as first-child-of-second-child-of root.

2.2 Invertibility: Unbinding from TPRs

TPRs are useful because they embed arbitrary symbolic structure in a vector space in such a way that simple linear algebra operations may be used to retrieve the form of the symbolic structure, including its compositional structure. The core operation in retrieving this structure is called *unbinding*. We may use unbinding to query a role for its filler. When the role vectors are linearly independent, there is a method for exact unbinding (see (Smolensky, 1990) for details). When the dimension of the role-embedding space is smaller than the number of distinct roles, the case we explore below, we must use an approximate unbinding method. This 'self-addressing'[1] unbinding method is what we will use to attempt to invert TPR embeddings to recover the filler of any given role as we test the Occupancy-Scaling Hypothesis, exploring how small a TPR we can use and still retain invertibility to a good degree of approximation.

Self-addressing unbinding retrieves the filler $\tilde{\mathbf{f}}_i$ for the role $\hat{\mathbf{r}}_i$ by simply computing the inner product between the role vector and the TPR: $\tilde{\mathbf{f}}_i = \mathbf{T} \cdot \hat{\mathbf{r}}_i = \sum_{j=1}^{k} (\hat{\mathbf{r}}_j \cdot \hat{\mathbf{r}}_i)\hat{\mathbf{f}}_i$. (Here and henceforth we assume all role vectors have been normalized.) This unbinding is exact if the role vectors are orthogonal. Otherwise, the intrusion of the filler of role j, $\hat{\mathbf{f}}_j$, into the unbound filler of role i, $\tilde{\mathbf{f}}_i$, is $\cos\theta_{ji}\hat{\mathbf{f}}_j$, where θ_{ji} is the angle between the role vectors $\hat{\mathbf{r}}_j$ and $\hat{\mathbf{r}}_i$.

While this unbinding is not exact, often we are interested in the case in which there is a fixed, known filler vocabulary with a given vector embedding. In such a case, it may be possible to use a similarity metric to compare the vector obtained from unbinding to the vectors embedding the vocabulary of fillers and selecting the vocabulary item with the highest value of the metric. Here, the cosine similarity of the two vectors is used as the metric; thus, we say an *unbinding error for role* i has occurred when there exists $j \neq i$ s.t.

$$\frac{\hat{\mathbf{f}}_j \cdot \tilde{\mathbf{f}}_i}{\|\hat{\mathbf{f}}_j\|\|\tilde{\mathbf{f}}_i\|} \geq \frac{\hat{\mathbf{f}}_i \cdot \tilde{\mathbf{f}}_i}{\|\hat{\mathbf{f}}_i\|\|\tilde{\mathbf{f}}_i\|}. \tag{2}$$

2.3 The geometry of S^n

In this section, we briefly present two geometric motivations for the hypothesis that, in high-dimensional spaces, random unit vectors may approximate orthogonality sufficiently for TPR unbinding. We also review a simple method for sampling from S^n, used throughout the paper to generate uniformly distributed unit vectors.

[1] The procedure is so called because the role vector is used to access its own filler.

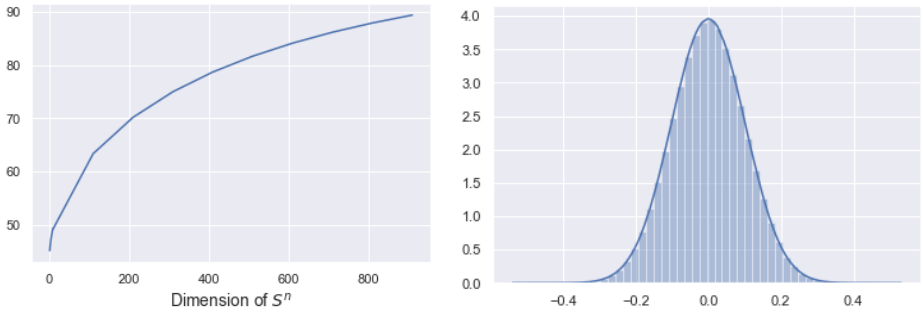

Figure 1: Left: Percentage of S^n forming an angle with the north pole in the range $90° \pm 2.5°$. Right: Empirical distribution (and normal estimate) of the dot products of 5000 random vectors in $\mathbb{R}^{100}$.

The first factor to note is that for a unit vector $\mathbf{u} \in \mathbb{R}^n$, as $n \to \infty$, the proportion of the n-dimensional unit sphere S^n with an angle $\phi \leq \Theta$ of $\mathbf{u}$ goes to 0 for all values of $\Theta < 90$ degrees. This has the implication that the proportion of S^n forming an angle of $90 - \varepsilon \leq \phi \leq 90 + \varepsilon$ (thus within ε of orthogonality) goes to 1. We can empirically estimate the rate at which this limit is approached, using Li (2011). As seen in Fig. 1 (left), the rate at which this region grows slows as the dimension increases, but the area is nevertheless large even for fairly small dimensions.

Another manifestation of the increased mass of S^n close to orthogonality to a given vector in higher dimensions can be found by considering the dot products of points selected at random from S^n. As shown in Appendix A, the mean and variance of the dot product of two random unit vectors in $\mathbb{R}^n$ is 0 and $1/n$. In high dimensions, the distribution appears to be well-approximated by a normal distribution: Fig. 1 (right) shows the distribution for $\mathbb{R}^{100}$. Therefore, most dot products are fairly small, and the larger the dot product, the less common it is. Finally, note that it is possible to sample uniformly from S^n (and thus sample a random unit vector) simply by sampling from the standard normal distribution. Samples $Z_1, Z_2, ..., Z_{n-1}$ are taken from the standard normal distribution. Then the ith coordinate of the sampled vector $\mathbf{v} \in S^n$ can be obtained by $Z_1, Z_2, ..., Z_n$, $[\mathbf{v}]_i = Z_i / \sqrt{\sum_{j=1}^{n-1} Z_j}$ (Muller, 1959).

3 Lower and Upper Bounds on Unbinding Error

First, we consider a lower bound on error: fully random TPRs. In this case, both filler vectors and role vectors are drawn uniformly from the unit sphere, and filler-role bindings are selected from the uniform distribution. This eliminates some potential contributions to error: there is no special relation between the representations of filler vectors, and no special co-occurence properties of roles or fillers. In this case, the intrusions of other fillers on the unbinding will typically be destructive, and the expected value of the intrusion term will be 0. Nevertheless, as the number of bound roles becomes large compared to the role dimension, the variance of the intrusion term becomes larger, resulting in errors.

In each simulation run, the size N of the set of possible fillers and the filler embedding dimension d were fixed. We perform a number of samples for each simulation, each time drawing a new set of n-dimensional role vectors $\{\hat{\mathbf{r}}_i\}_{i=0}^{k} \sim \mathcal{U}(S^{n-1})$ that will be bound to fillers (so the occupancy is k). dings k is fixed and the dimension of the role vectors n is varied, while in others n is fixed and k is varied. The former simulations can be thought of as answering the question "How large of role vectors do I need if my symbolic structures are no larger than k?", while the latter answer "How much information can be packed into TPRs using roles of size n?" The filler embedding that will be bound to each $\hat{\mathbf{r}}_i$ is drawn IID from a uniform distribution over the set of N possible fillers. For each

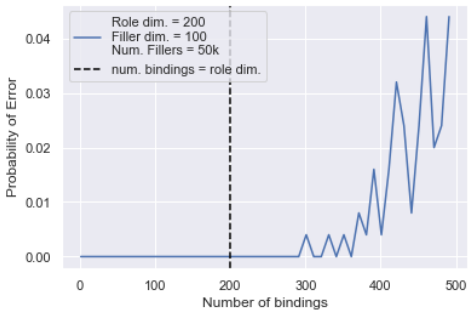

Figure 2: Example error for random TPRs.

fixed set of parameters, we select the filler-role bindings and create the TPR according to Equation (1). We then unbind all the roles using the self-addressing unbinding procedure, compute the similarities between the result of the unbinding $\tilde{\mathbf{f}}$ and each of the filler vectors $\hat{\mathbf{f}}_j$, recording whether an error was made. We divide the number of errors made by the total number of bindings to obtain a simple maximum likelihood estimate of the error probability for any one combination of N, d, n, and k. Simulations were computed in batches using PyTorch (Paszke et al., 2019).

This experiment was conducted with both the role dimension n fixed and the number of bindings k varied and vice-versa, for fixed $n, k = 25, 100, 200$, with $d = 100$, $N = 2000, 10000, 50000$. Results for fixed n and varied k and fixed k and varied n showed analogous patterns of error, suggesting that the ratio of n/k is the relevant factor for error, rather than their independent values. The number of possible fillers N did not seem to substantially effect the error rate. Overall, across all combinations of n and k, the error for $k/n < 2$ was generally less than 1%. This constitutes a confirmation of the Occupancy-Scaling Hypothesis, with scaling coefficient $\gamma = 2$: unbinding error is $< 1\%$ when $k < 2n$. Representative results are shown in Fig. 2.

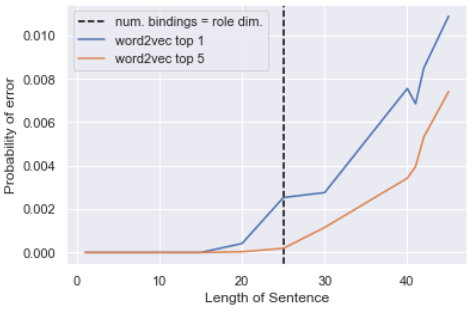

Figure 3: Sentence embedding error with role dimension $n = 25$.

Another relatively simple but more challenging setting is given by embedding English sentences. We use the Reuters corpus from the NLTK Python package (Bird et al., 2009), taking only the sentences of length ≤ 50, yielding 49442 sentences. Here we construct at TPR as follows: if w_i is the ith word in a sentence from the corpus, the filler vector $\hat{\mathbf{f}}_i$ is the embedding of w_i in some vector space; this is bound to an embedding $\mathbf{r}_i$ of the role denoting the ith linear position in the sentence. As before, the role vectors are randomly chosen from the uniform distribution on the unit sphere $S^{n-1} \subset \mathbb{R}^n$. For the word embeddings, we use 300-dimensional word2vec vectors taken from the Google News vectors (Mikolov et al., 2013). There are a number of potential issues here: the fillers cannot be modeled as being drawn from a uniform distribution; since the fillers are words, common words will appear in TPRs more often, being drawn from a distribution which is approximately Zipfian (Zipf, 1949). If a word appears more than once in a sentence (and thus TPR), that increases the chance of constructive interference in the direction of that word. Another challenge is the non-uniformity of word2vec vectors: since word2vec creates vectors on the basis that words that occur near each other (e.g. in the same sentence) should have more similar vectorial embeddings. This means the embeddings of the intruding vectors will be closer to the true filler than a random vector would be expected to be, leading to a potential for errors. Finally, the density of the filler space in this experiment is much greater than in the previous experiments, as there are approximately 3,000,000 GoogleNews vectors, with a dimension of 300 ($N/d = 10000$), increasing the change of a random error. Due to this large filler dimension, we also consider a top-5 setting, which reduces errors. Despite these potential issues, using role dimension $n = 25$ and letting k (here, the number of words in the sentence) vary, we again find $\gamma \approx 2$ with a tolerance of 1% error, as shown in Fig. 3.

Finally, we present a worst-case scenario. Consider a TPR where role vectors are uniformly drawn from the unit sphere S^{n-1} and where each role is bound to one of only two fillers $\hat{\mathbf{a}}$ or $\hat{\mathbf{b}}$. Specifically, $\hat{\mathbf{r}}_0$ is bound to $\hat{\mathbf{a}}$, and for all $i \neq 0$, $\hat{\mathbf{r}}_i$ is bound to $\hat{\mathbf{b}}$. When unbinding a role from a TPR where the fillers are widely scattered, there will be destructive interference causing cancellation between the intrusions of the fillers of other roles; in this case, however, when unbinding $\hat{\mathbf{r}}_0$ there will be no such destructive interference, but instead constructive interference in the direction of $\hat{\mathbf{b}}$. Thus, we call this scenario *maximal intrusion*. The

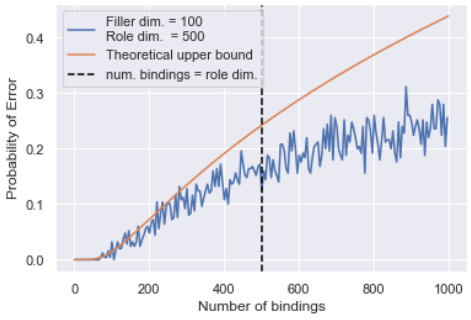

Figure 4: Maximal intrusion error (empirical and upper bound).

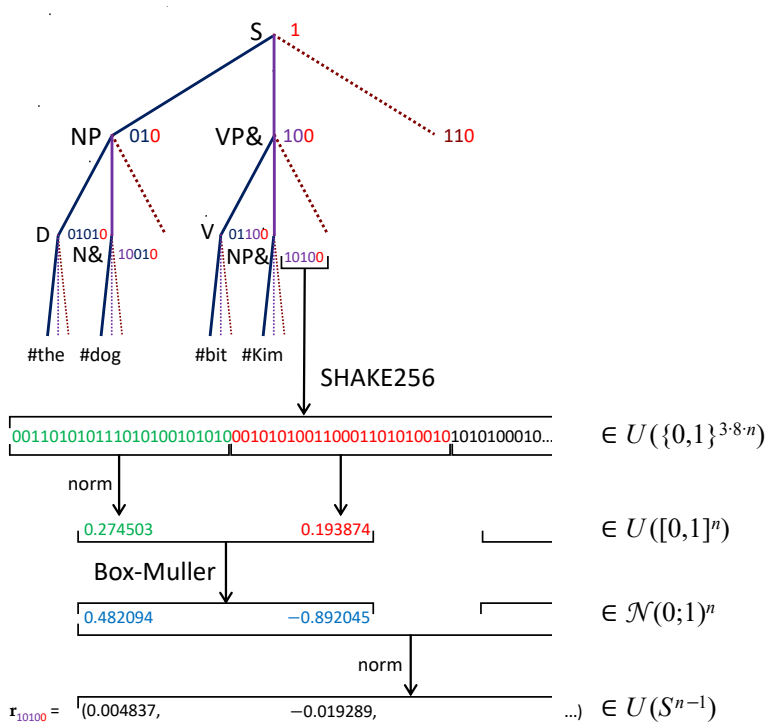

Figure 5: **Cryptographic Role Embedding.** Here the branching factor $b = 3$; dotted edges lead to unoccupied subtrees. Daughters are coded as 01 (left); 10 (center); 11 (right). Nodes are addressed by bit strings encoding the path to the root. Rightmost daughters' labels have a suffix '&'; terminal nodes have a prefix '#'. Together, we refer to these nodes as *final nodes*. The numbers below the tree derive the vector embedding of the node with address 10100. The derivation follows the steps given in Sec. 4.1, with the distribution of quantities at each step shown to the right.

error in this case risks being rather high, although this scenario is very unlikely in real-data settings. The probability that unbinding $\hat{\mathbf{r}}_0$ will erroneously yield $\hat{\mathbf{b}}$ rather than the correct result $\hat{\mathbf{b}}$ is bounded by $P(\text{Error}) < \frac{e^{-n/2k}}{\sqrt{2\pi n/k}}$ (see Appendix B for proof.)

In Fig. 4, we can see that this bound is not tight, with the observed error being substantially lower. Still, the bound has the property of being exponentially decreasing in n–that is, for a fixed number of bindings k, the error drops off at a rate proportional to $\frac{1}{e^n}$, so even in this worst-case scenario it is possible to unbind with a low error rate with a number of roles favorably proportional to the number of bindings, provided k is not $\gg$ than the desired n.

4 Invertible Tree Representations

The TPR embedding of trees is guaranteed to be perfectly invertible if the role embeddings are linearly independent. The roles here are the possible positions in a k-ary tree, so the number of tree roles grows exponentially with the depth of the tree; as such, maintaining the linear independence of all role vectors that is required for exact unbinding would require extremely large role vectors. However, the experiments presented in Section 3 suggest that sufficiently-high-dimensional random role vectors may be close enough to orthogonal that trees can be represented with relatively small role vectors while introducing only a small amount of error in inverting the embedding (i.e., only a small probability of unbinding errors). However, purely random roles are not feasible in the tree context, as the number of roles is potentially infinite and grows exponentially in the depth. In this section, we present a scalable role scheme for the representation of tree TPRs, and demonstrate its efficiency in representing syntax trees with minimal information loss (as demonstrated by reconstruction).

4.1 Cryptographic Role Embedding: A pseudorandom, deterministic role scheme[2].

This tree representation scheme is designed with a few goals in mind. First, in conformity with the Occupancy-Scaling Hypothesis, the size of the representation should scale not with the depth, but with the number of filled nodes in the tree: the occupancy of the tree. Thus, sparse trees should enable a smaller representation than complete trees, even if the sparse trees are much deeper. Additionally, the roles should be close to random. independent samples from the unit sphere. The theoretical and empirical work presented here indicates that while orthogonality of role vectors is required for guaranteed perfect accuracy in role unbinding, a high degree of accuracy across varied scenarios is possible when roles are randomly drawn from the unit sphere of sufficiently high dimension. We seek then to represent positions in a tree as random points on the unit sphere; however, simply drawing randomly from the unit sphere is not scalable. For each position in the tree, a unique random vector is needed. The number of such positions in a k-ary branching tree of depth d is $(k^d - 1)/(k - 1)$. To avoid storing an exponential number of vectors and requiring a maximal depth, we propose a system in which no vectors need be stored, but the (pseudo)random vector for any position can instead be generated on-demand repeatedly.

The generation of an n-dimensional role vector in the proposed scheme can be divided into 4 steps (see Fig. 5.):

1. Tree position $\rightarrow$ bit string
2. Bit string $\rightarrow 3n$ pseudorandom uniform bytes (SHAKE256)
3. $3n$ pseudorandom uniform bytes $\rightarrow n$ pseudorandom independent Gaussian samples (Box-Muller)
4. n pseudorandom independent Gaussian samples $\rightarrow n$-dimensional pseudorandom unit vector

Each role (tree-node position) is addressed by variable-length bit string which encodes the path from the node to the root according to a simple set of rules.[3] This string must then be used to deterministically generate a sequence of pseudorandom bits. Since the string representation is such that similar roles have similar strings (in terms of, e.g., edit distance), it is essential that this generation process not map similar strings to similar sequences of bits if the independence of roles is to be maintained.

Cryptographic hash functions are a class of functions designed to solve exactly this problem. These functions map input strings of any length to a fixed number of output bits such that 1) it is not feasible to find 2 inputs which map to the same output, 2) a small change to the input results in a large change in the output, and 3) the process is fully deterministic, relying on no source of randomness. These attributes of hash functions point towards the output bits of different tree position strings being random and uncorrelated. In this work, the SHAKE256 variable-length hash function was applied to the tree position strings to generate a sequence of output bits (FIPS 202, 2015).

The output bits of the SHAKE256 hash function are effectively uniformly distributed; however, in order to obtain a uniform sample of unit *vectors*, random samples from a *Gaussian* distribution are needed (see Section 2.3). The **Box-Muller transform** takes uniform samples on the interval $[0, 1]$ and deterministically produces independent normally-distributed samples. To obtain samples on the interval $[0, 1]$ from the output of the hash function (a sequence of random bits), 3 (8-bit) bytes of output were taken at a time and normalized by dividing by $2^{(3\cdot 8)}$. This results in n uniform samples to which the Box-Muller transform may be applied [4]. The Box-Muller transform takes 2 (pseudo)random uniform

[2]Code for the representations developed here, as well as the other experiments and results in this paper is available at `https://github.com/ColemanHaley/InvertibleTreeRepresentations`

[3]The root node was represented by "`1`", all other strings are constructed right-to-left and begin with "`0`". Let the *arity* of the tree (maximum allowed amount of branching) be denoted k; then the remainder of the tree representation is divided into sections of $\lceil \log_2(k + 1) \rceil$ bits. Each section contains the binary value of the distance from the leftmost child of the parent of the node (plus one). For example, consider in a binary tree the root's right child's left child's right child (RLR for short). In this case, each section is 2 bits long (plus the rightmost 0 representing the root), so the string representing this position is `1001100`.)

[4]The SHAKE256 hash function was selected due to its ability to produce variable length outputs, allowing the appropriate number of output bits to be created; note however, that the amount of entropy of the output is *fixed*, meaning that for extremely long output lengths (for large n), there is a chance the function may be insufficiently random.

samples U_1 and U_2 as input, and produces 2 (pseudo)random Gaussian-distributed samples Z_1 and Z_2 as output according to the following formulae:

$$Z_1 = \sqrt{-2 \ln U_1} \cos(2\pi U_2) \qquad Z_2 = \sqrt{-2 \ln U_2} \cos(2\pi U_1)$$

While the formulae are similar, it has been proven that Z_1 and Z_2 are independent samples. The Box-Muller transform is applied iteratively to pairs of the uniform samples derived from the hash function output until n independent Gaussian samples are generated. With these Gaussian samples $Z_1, Z_2, ..., Z_n$, the i-th coordinate of the node's role vector $\mathbf{r}$ is given by $[\mathbf{r}]_i = Z_i / \sqrt{\sum_{j=1}^{n} Z_j}$.

To test how close to uniformly distributed these role vectors are, the branching factor was set to 3 and the vectors for all tree positions up to depth 5 were generated (1093 vectors). The dot products of all pairs of vectors were computed, and the distribution was compared with an analogous distribution for 1093 randomly sampled unit vectors using Levene's test for variance equality, which showed equal variances with $p > 0.99$.

4.2 The filler space

We assume the vocabulary of words (terminal labels) and nonterminal labels is fixed and known, and relatively small compared to the number of tree positions. Thus, we memoize the filler vectors, and we need not use any sort of generation scheme in contrast to the roles. Each filler vector is an independent random sample from the unit sphere in $\mathbb{R}^d$, S^{d-1}, where d is the filler vector dimension.

In inverting a tree embedding, determining which positions are actually present in the tree and which are empty is a non-trivial issue — unless a node has the maximum number of children, it may have further children to the right of the child being processed; it is similarly possible that a leaf node may have further children. In the TPR scheme these unfilled roles are implicitly bound to the **0** filler vector; however, empirically no threshold was identified for filler magnitude to reliably distinguish filled positions and unfilled positions. As a result, special fillers were created representing each filler when it is the rightmost child of a parent (adding the suffix "&") and when it is a leaf (adding the prefix "#"); these fillers and their associated vectors are used instead of the true fillers in the representations. When inverting the representation, the prefixes are used to guide the search through the tree to only filled positions and are stripped from the fillers in order to reconstruct the true tree.

4.3 Experiments

In order to test the invertibility of this representation scheme and its utility in representing naturalistic data, we carry out experiments on syntactic trees from the MASC dataset (Ide et al., 2010). This dataset contains approximately 500,000 words of text from diverse domains, separated into sentences which are annotated into parse trees in the Penn Treebank format. This dataset contains a wide variety of trees — its most branching node has 98 children, and its deepest tree has a depth of 43 nodes. This means there are $98^{43} \approx 4.19 \times 10^{85}$ possible tree positions to be represented by role vectors. Extreme outliers in terms of number of nodes were removed by taking all trees of size (occupancy) < 183 (approx. 99% of trees), yielding 35,379 syntax trees. Inversion of the representation is accomplished by conducting a breadth-first tree traversal by enqueuing possible tree positions, then at each position producing the appropriate role vector through the process outlined in Section 4.1, unbinding and seeing if the bound symbols are marked as rightmost or leaf symbols, and using that to keep empty sibling and child positions out of the queue.

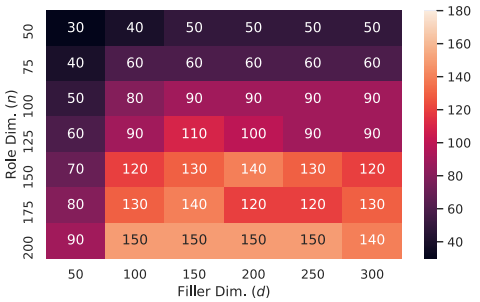

Figure 6: The largest number of nodes (k) for which a given filler and role dimension combination has error $< 1\%$ for all smaller tree sizes.[5]

The reconstructed tree and the original tree may not contain the same set of nodes — there may be, for instance, positions found in the reconstruction that are not present in the true tree. Due to the multiple types of errors possible, to produce a unified metric we take the F-score over the pairs of positions and node labels (thus, an incorrect filler for a correct role is treated as 2 errors). $1 - F$ is then treated as the "error." Within each experiment, the role and filler dimensions are held constant and evaluated over all sentences in MASC. Sentences are grouped by the total number of nodes in their tree representations. The role and filler dimensions were then varied independently. Figure 6 shows the largest tree size (in terms of number of occupied nodes) for which error is below $< 1\%$ *and* the error for all smaller trees is $< 1\%$–the last point at which error is consistently below 1%.

The data indicates that while filler dimension plays a role in representation quality, there seems to be a "threshold size" in these experiments, 150, above which the filler dimension does not substantially aid performance. In contrast, no such upper bound was identified for increasing role dimension. In terms of the Occupancy-Scaling Hypothesis, we find a more complex story here. For sufficiently large filler dim. $d \geq 150$, values of $\gamma = k/n$ range from 0.69 to 1. Further experimentation is needed to definitively confirm whether the Occupancy-Scaling Hypothesisholds in this case; if it does, its value likely lies within this range.

We found that for trees of length < 150, Error was consistently below 1% for role dimension 200 and filler dimension 150 ($\gamma = 0.75$). Since the deepest tree of length ≤ 150 has depth 29, the widest branching has 98-ary branching, and there, it would take roles of dimension $(98^{29} - 1)/(98 - 1) \approx 5.7 \times 10^{55}$ to achieve the linear independence required to represent this exactly, yielding representations of approx 8.6×10^{57} units if using the 150-dimensional filler-vectors used here.

The reconstruction of tree-structured data presents a significant challenge that may account for the higher error in this case than previous cases — in a tree, a large percentage of nodes are either the rightmost child of their parent or a leaf. This means that there are many opportunities to make an error in reconstructing these nodes — when such an error is made, it often results in a final node being mistaken as a non-final node or vice-versa, leading to spurious unbinding of roles that are unbound or not traversing an entire subtree. In addition, the branching nature of tree structure means that such errors easily result in an exponential number of additional errors. This is an issue with the nature of the task, not the representation. Preliminary investigation by the authors into augmenting the unbinding process with an oracle determining which nodes are and are not final, indicated $\gamma \approx 2$, similar to the random TPR and sentence TPR results in Section 3; however, this is not pursued here *because* it a less challenging task that requires burdensome assumptions for use.

5 Summary and conclusion

Can trees of potentially large depth be encoded as distributed representations in such a way as to enable fully parallel processing and high-accuracy decoding without requiring representations with a dimensionality that grows rapidly with the maximum possible depth? General hand-designed methods for neural encoding of compositional structure, and learned internal neural representations for strongly compositional tasks, have been shown to be cases of tensor product representations (TPRs), which provide distributed representations that enable parallel processing and accurate decoding — if the dimension of the representation is very large, growing exponentially with tree depth. A method is presented here for designing distributed tree embeddings (TPRs) that, by contrast, have the same scaling properties as symbolic tree representations: they grow with the number of labelled tree nodes, independently of tree depth. This technique uses Cryptographic Role embeddings to encode tree positions. This is essentially a means of hashing tree positions onto a high-dimensional unit sphere such that nearby tree positions are not mapped to nearby unit vectors, minimizing interference between decoding symbols that are close together in the tree. The use of a cryptographic hash means there are no burdensome requirements of memoization, allowing the flexible handling of extremely large or unbounded role schemes for complex compositional structures. Experiments show that trees that would require TPRs of size 8.6×10^{57} to enable exact

[5]To mitigate noise and indicate the true error trends in the data, trees were binned by length to the nearest 10. E.g., all trees of length 140, 141, ..., 149 contribute to a common average error for 140.

decoding can, with Cryptographic Role embeddings, be embedded as vectors of dimension 30,000 while keeping the probability of error in decoding a tree position less than 1%. Because the representation size is fixed, any of the $80000\binom{98\cdot150}{150} = 8.34 \times 10^{213}$ trees of 150 nodes can be represented with this scheme. This success, coupled with the upper and lower bounds on error shown in Section 3, point to a powerful potential for TPR-based representations of many types of structure in language and other compositional domains.

Acknowledgements

The authors would like to thank Colin Wilson and the Neuro-Symbolic Computation Lab at Johns Hopkins University for their helpful comments. This work was in part supported by a Provost's Undergraduate Research Award at Johns Hopkins University. This work utilizes resources supported by the National Science Foundation's Major Research Instrumentation program, grant #1725729, as well as the University of Illinois at Urbana-Champaign.

References

Marco Baroni. 2020. Linguistic generalization and compositionality in modern artificial neural networks. *Philosophical Transactions of the Royal Society B*, 375(1791):20190307.

Steven Bird, Edward Loper, and Ewan Klein. 2009. *Natural Language Processing with Python*. O'Reilly Media Inc.

Jerry A Fodor and Zenon W Pylyshyn. 1988. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1-2):3–71.

Gottlob Frege and Michael Beaney. 1997. *The Frege Reader*. Blackwell.

Nancy Ide, Collin Baker, Christiane Fellbaum, and Rebecca Passonneau. 2010. The manually annotated subcorpus: A community resource for and by the people. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 68–73, Uppsala, Sweden, July. Association for Computational Linguistics.

Brenden M Lake and Marco Baroni. 2017. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *Proceedings of the 35th International Conference on Machine Learning*, pages 2879–2888.

Géraldine Legendre, Yoshiro Miyata, and Paul Smolensky. 1991. Distributed recursive structure processing. In *Advances in Neural Information Processing Systems*, pages 591–597.

Song Lin Li. 2011. Concise formulas for the area and volume of a hyperspherical cap. *Asian Journal of Mathematics & Statistics*, 4:66–70.

Gary Marcus. 2020. The next decade in AI: Four steps towards robust artificial intelligence. *arXiv preprint arXiv:2002.06177*.

R Thomas McCoy, Tal Linzen, Ewan Dunbar, and Paul Smolensky. 2019. RNNs implicitly implement tensor product representations. In *International Conference on Learning Representations*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Mervin E. Muller. 1959. A note on a method for generating points uniformly on n-dimensional spheres. *Commun. ACM*, 2(4):19–20, April.

Allen Newell. 1980. Physical symbol systems. *Cognitive Science*, 4(2):135–183.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8026–8037. Curran Associates, Inc.

Tony A Plate. 1995. Holographic Reduced Representations. *IEEE Transactions on Neural networks*, 6(3):623–641.

Jordan B Pollack. 1989. Implications of recursive distributed representations. In *Advances in Neural Information Processing Systems*, pages 527–536.

Jacob Russin, Jason Jo, Randall O'Reilly, and Yoshua Bengio. 2020. Compositional generalization by factorizing alignment and translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 313–327, Online, July. Association for Computational Linguistics.

Lokendra Shastri and Venkat Ajjanagadde. 1993. From simple associations to systematic reasoning: A connectionist representation of rules, variables and dynamic bindings using temporal synchrony. *Behavioral and brain sciences*, 16:417–417.

Paul Smolensky and Bruce B. Tesar. 2006. Symbolic computation with activation patterns. In Paul Smolensky and Géraldine Legendre, editors, *The harmonic mind: From neural computation to optimality-theoretic grammar*, volume 1, pages 235–270. MIT Press.

Paul Smolensky. 1990. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence*, 46(1-2):159–216.

Paul Smolensky. 2012. Symbolic functions from neural computation. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 370(1971):3543–3569.

Paul Soulos, Tom McCoy, Tal Linzen, and Paul Smolensky. 2019. Discovering the compositional structure of vector representations with role learning networks. In *NeurIPS Workshop on Context and Composition in Biological and Artificial Neural Systems, arXiv:1910.09113*.

G. K. Zipf. 1949. *Human behavior and the principle of least effort*. Addison-Wesley Press.

A Proof that the variance σ_n^2 of $s_{1,n}$ as $n \to \infty$ is $1/n$

Let $s_{1,n} \equiv \hat{\mathbf{r}}_i \cdot \hat{\mathbf{r}}_0$ be a random function of $\hat{\mathbf{r}}_i \in \mathbb{R}^n \sim \mathcal{U}(S^{n-1})$. Its variance is

$$\sigma_n^2 = \int_0^\infty (\hat{\mathbf{r}}_i \cdot \hat{\mathbf{r}}_0)^2 p(\hat{\mathbf{r}}_i \cdot \hat{\mathbf{r}}_0 = \cos\theta) d\theta = \int_0^\infty \cos^2\theta p(\hat{\mathbf{r}}_i \cdot \hat{\mathbf{r}}_0 = \cos\theta) d\theta \tag{3}$$

Represent $\hat{\mathbf{r}}$ vectors in an orthonormal basis in which $\hat{\mathbf{r}}_0 = \hat{\mathbf{e}}_0$. Then

$$\begin{aligned}
\{\hat{\mathbf{r}}_i | \hat{\mathbf{r}}_i \cdot \hat{\mathbf{r}}_0 = \cos(\theta)\} &= \{\hat{\mathbf{r}}_i | [\hat{\mathbf{r}}_i]_0 = \cos(\theta)\} && (4)\\
&= \left\{ \hat{\mathbf{r}}_i \middle| 1 = \sum_{q=0}^{n} [\hat{\mathbf{r}}_i]_q^2 = \cos^2\theta + \sum_{q=1}^{n} [\hat{\mathbf{r}}_i]_q^2 \right\} && (5)\\
&= \left\{ \hat{\mathbf{r}}_i \middle| \sum_{q=1}^{n} [\hat{\mathbf{r}}_i]_q^2 = 1 - \cos^2\theta = \sin^2\theta \right\} && (6)\\
&= S^{n-2}(\sin\theta) && (7)
\end{aligned}$$

Letting the hyper-surface area of the radius-r sphere $S^{n-1}(r) \subset \mathbb{R}^n$ be $A^{n-1}(r) = C_{n-1} r^{n-1}$, we have

$$\begin{aligned}
\sigma_n^2 &= \int_0^\infty \cos^2\theta p(\hat{\mathbf{r}}_i \cdot \hat{\mathbf{r}}_0 = \cos\theta) d\theta && (8)\\
&= \int_0^\pi \cos^2\theta \frac{A^{n-2}(\sin\theta)}{A^{n-1}(1)} d\theta && (9)\\
&= 2\frac{C_{n-2}}{C_{n-1}} \int_0^{\pi/2} (1 - \sin^2\theta)(\sin^{n-2}\theta) d\theta && (10)\\
&= 2\frac{C_{n-2}}{C_{n-1}} \left[I(n-2) - I(n) \right] && (11)
\end{aligned}$$

with

$$C_{n-1} = \frac{2\pi^{n/2}}{\Gamma(n/2)} \Rightarrow C_{n-2} = \frac{2\pi^{[n-1]/2}}{\Gamma([n-1]/2)} = \frac{2\pi^{n/2-1/2}}{\Gamma([n/2-1]+1/2)} \tag{12}$$

and

$$\begin{aligned} I(m) &\equiv \int_0^{\pi/2} \sin^m \theta d\theta & (13)\\ &= \frac{\sqrt{\pi}}{2}\frac{\Gamma(\frac{m}{2}+\frac{1}{2})}{\Gamma(\frac{m}{2}+1)} & (14)\end{aligned}$$

Assume henceforth that n is even:

$$u \equiv \frac{n}{2} \in \mathbb{Z} \Rightarrow \frac{n-2}{2} = u - 1 \in \mathbb{Z} \tag{15}$$

Then we can use, for $v \in \mathbb{Z}$:

$$\Gamma(v + \frac{1}{2}) = \frac{(2v)!}{4^v v!}\sqrt{\pi} \tag{16}$$

which together with

$$\Gamma(v) = (v-1)! \tag{17}$$

and $v \equiv m/2$ entails

$$\begin{aligned} I(m) &= \frac{\sqrt{\pi}}{2}\left(\frac{(2v)!}{4^v v!}\sqrt{\pi}\right)\frac{1}{v!} & (18)\\ &= \frac{\pi}{2}\frac{(2v)!}{4^v (v!)^2} & (19)\end{aligned}$$

hence

$$I(m-2) = \frac{\pi}{2}\frac{(2[v-1])!}{4^{v-1}([v-1]!)^2} \tag{20}$$

Note that when $n = 2u$ is even, (12) becomes

$$C_{n-1} = \frac{2\pi^u}{(u-1)!} \tag{21}$$

Then, from (11), (15), (19) and (20) we get

$$\begin{aligned} \sigma_n^2 &= 2\frac{C_{n-2}}{C_{n-1}}\frac{\pi}{2}\left[\frac{(2[u-1])!}{4^{u-1}([u-1]!)^2} - \frac{(2u)!}{4^u (u!)^2}\right] & (22)\\ &= \pi\frac{C_{n-2}}{C_{n-1}}\frac{(2[u-1])!}{4^{u-1}([u-1]!)^2}\left[1 - \frac{(2u)(2u-1)}{4(u)^2}\right] & (23)\\ &= \pi\frac{C_{n-2}}{C_{n-1}}\frac{(2[u-1])!}{4^{u-1}([u-1]!)^2}\frac{1}{2u} & (24)\\ &= \pi\frac{C_{n-2}}{C_{n-1}}\frac{(2x)!}{4^x (x!)^2}\frac{1}{2(x+1)} & (25)\end{aligned}$$

where

$$x \equiv u - 1 = n/2 - 1 \tag{26}$$

Further, from (12), (16) and (21), we have

$$\begin{aligned} \frac{C_{n-2}}{C_{n-1}} &= \frac{2\pi^u \pi^{-1/2}}{\Gamma([u-1]+1/2)}\frac{(u-1)!}{2\pi^u} & (27)\\ &= \frac{1}{\sqrt{\pi}}\frac{4^{u-1}(u-1)!}{\sqrt{\pi}(2[u-1])!}(u-1)! & (28)\\ &= \frac{4^x (x!)^2}{\pi(2x)!} & (29)\end{aligned}$$

Then

$$\sigma_n^2 = \pi \frac{4^x (x!)^2}{\pi (2x)!} \frac{(2x)!}{4^x (x!)^2} \frac{1}{2(x+1)} \tag{30}$$

$$= \frac{1}{2(x+1)} = \frac{1}{n} \tag{31}$$

B Proof of bound on maximal intrusion error

In the restricted worst-case scenario of maximal intrusion, we can prove an upper bound on a restricted case of error. We will consider it a *Type I error* when the unbinding $\tilde{\mathbf{f}}_0$ of $\hat{\mathbf{r}}_0$ is closer to $\hat{\mathbf{b}}$ than the correct filler vector $\hat{\mathbf{a}}$. What will call a *Type II Error* arises if there exists any $j < N$ such that $\hat{\mathbf{f}}_j$ is closer to $\tilde{\mathbf{f}}_0$ than $\hat{\mathbf{a}}$ is. Note that Type II Error is the type of error we have been considering so far. In this case, all intrusion is in the direction of $\hat{\mathbf{b}}$, so we expect that Type I errors will constitute the majority of Type II errors. We can express the unbinding of $\hat{\mathbf{r}}_0$ as $\tilde{\mathbf{f}} = \hat{\mathbf{a}} + s_{k,n}\hat{\mathbf{b}}$ where $s_{k,n} \equiv \sum_{i=1}^{k} i \cdot \hat{\mathbf{r}}_0$ is a random variable with distribution determined by the independently uniform distribution of $\{\hat{\mathbf{r}}\}_{i=0}^{k} \subset S^{n-1}$. A Type I error occurs iff

$$1 + s_{k,n}c < c + s_{k,n} \Leftrightarrow 1 - c < s_{k,n}(1-c) \Leftrightarrow s_{k,n} > 1, \tag{32}$$

where $c \equiv \hat{\mathbf{a}} \cdot \hat{\mathbf{b}} \in [-1, 1]$. Thus,

$$P(\text{Type I error}) = P\Big(s_{k,n} \equiv \sum_{i=1}^{k} \hat{\mathbf{r}}_i \cdot \hat{\mathbf{r}}_0 > 1 \,\Big|\, \hat{\mathbf{r}}_i \sim \mathcal{U}(S^{n-1})\Big). \tag{33}$$

Under the assumption $k \gg 1$, we can derive an upper bound on this error probability as a function of k and n. By the Central Limit Theorem, if the distribution of $s_{1,n} \equiv \hat{\mathbf{r}}_i \cdot \hat{\mathbf{r}}_0$ has mean zero and variance $\sigma_n^2 (\forall i = 1, ..., k)$, then as $k \to \infty$,

$$P\left(\frac{1}{k}\sum_{i=1}^{k} \hat{\mathbf{r}}_i \cdot \hat{\mathbf{r}}_0 > 0\right) \to P(X > a \,|\, X \sim \mathcal{N}(0, \sigma_n^2)) \tag{34}$$

$$= P\Big(Y > \frac{a}{\sigma_n} \,|\, Y \sim \mathcal{N}(0,1)\Big). \tag{35}$$

So

$$P(\text{Type I error}) = P(\sum_{i=1}^{k} \hat{\mathbf{r}}_i \cdot \hat{\mathbf{r}}_0 > 1) \tag{36}$$

$$= P(\frac{1}{\sqrt{k}}\sum_{i=1}^{k} \hat{\mathbf{r}}_i \cdot \hat{\mathbf{r}}_0 > \frac{1}{\sqrt{k}}) \tag{37}$$

$$\to P\left(Y > \frac{1}{\sqrt{k}\,\sigma_n} \,|\, Y \sim \mathcal{N}(0,1)\right) \tag{38}$$

for $k \gg 1$. As shown in Appendix A, $\sigma_n^2 = 1/n$, so

$$P(\text{Type I error}) \to P(Y > \sqrt{n/k} \,|\, Y \sim \mathcal{N}(0,1)) \tag{39}$$

$$< \frac{e^{-n/2k}}{\sqrt{2\pi n/k}}. \tag{40}$$

While the bound is on Type I error and not the more general Type II error, we found empirically almost errors are of Type I, as all intrusion is in the direction of a Type I error.