STRUCTURE ASSEMBLY IN KNOWLEDGE BASE REPRESENTATION

by

Matthias Lalisse

A dissertation submitted to Johns Hopkins University in conformity with the requirements for the degree of Doctor of Philosophy

Baltimore, Maryland
September 2021

Abstract

A primary goal of connectionist cognitive science is to provide the technical apparatus for modeling cognitive processes as implemented in brainlike systems. From the structure of classical cognitive theories—alphabets of discrete symbols along with algebraic operations on those primitive symbols—one can derive key properties attributed to "higher-order cognition" by authors like Fodor and Pylyshyn—systematicity, productivity, and compositionality. This fact has led to theories of that type to serve as the enduring backbone of cognitive science. Connectionist networks differ from these classical systems not just in that they are implemented in real numbers while the former operate over discrete sets, but also in that they can represent system-states that cannot be factored into algebraic combinations of primitive symbols that are, in a formal sense, noncompositional.

This dissertation develops, examines, and evaluates a series of models that straddle that divide. Cognitively-oriented connectionist models in the tradition of Vector Symbolic Architectures (VSAs) provide a framework by which neural network models can integrate Fodor and Pylyshyn's insights, as well as the corresponding architectural commitments of cognitive theory. Other processes, based in Harmony Maximization [Smolensky and Legendre, 2006], adjust these structured representations to satisfy learned constraints.

We present three models that use Tensor Product Representations (TPRs) and other VSAs closely related to them, placing these in a common formal framework and then

applying them to Knowledge Base Completion: the task of storing large-scale inventories of facts (e.g. WordNet, Freebase, Wikidata) in representations that allow those databases to be extended via inexact inference. In typical approaches, graph representations are obtained compositionally by taking static vector representations of graph elements (entities and relations) and combining them systematically in order to derive a score. The first two models combine the compositional operations of VSAs with context-modulation processes based in Harmonic Grammar [Smolensky and Legendre, 2006]. A third model examines the proposition that spatial structure implicit in TPRs—with a number of spatial directions equal to the order of the tensor—can be used as an organizing principle for the features encoded in the trilinear tensors occurring in the graph representation setting. Each of the models, we show, performs at the state of the art in Knowledge Base Completion, and we explore the qualitative aspects of the representations that they learn.

Readers: Paul Smolensky, Kyle Rawlins, Alan Yuille, Chris Honey, Hynek Hermansky

Acknowledgments

A dissertation is not the work of any one person. It's the product of the emotional and intellectual labor of an entire community. For his role in ushering this thesis through, the very greatest debt is owed to Paul Smolensky. His ideas, theories, and most of all his commitment to formal development are most clearly threaded throughout this work. Warm thanks as well to the members of my dissertation committee: Kyle Rawlins, Alan Yuille, Chris Honey, and Hynek Hermansky. Without Kyle in particular, and his enduring interest in bridging between formal semantics and computational modeling, this thesis might have taken a very different form. To the members of the Neurosymbolic Computation Lab—Caitlin Smith, Eric Rosen, Paul Soulos, Najoung Kim, and Tom McCoy—thanks are most deserved, especially to Eric Rosen, who collaborated on the work comprising Chapter 5. My parents, Charlotte Marcy and Dominique Lalisse, for always being there when I needed them. My entering class—Jane Lutken, Celia Litovsky, Don Li, and Sadhwi Srinivas—for the welcoming culture they helped to build. Finally, to Teachers and Researchers United, the graduate worker union at Johns Hopkins University.

Dedication

To my grandfather, Freddy Marcy, who died abruptly and far too early while this thesis was still on its way.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

One hallmark of the field cognitive science is the conception of cognitive capacities as systems as computational procedures that operate over domains of symbols to produce behavior. Representations, in this tradition, are typically discrete, in contrast to the great variety of neural models that have emerged to model and perform cognitive tasks, often in an applied setting. To a degree, there is substantive overlap between these two views, though the connections have become more remote over time as the complexity of the ensuing models and the functions involved in building them has markedly increased.

The implications of cognitive models of the classical kind—meaning those comprised of discrete symbols and algebraic rules for combining them—was well summarized by Fodor and Pylyshyn FodorPylyshyn1988. Such models provide, without additional stipulations, explanations for three key properties of "higher cognition"—meaning that evinced by humans. Cognition is *productive* in that in some[1] but not all domains, there is discrete infinity [von Humboldt, 1999]. Cognition is systematic in the sense that there are syntactic links between cognitive representations in virtue of what constituents appear in those

[1] e.g. language.

representations. For example, systematicity says that if Mary loves Kevin is a sentence, then Kevin loves Mary is also an expression. Put another way, there is algebraic closure of the alphabet under the operations of the grammar. Given the grammar and the primitive symbols, one can obtain all recombinations of the constituents as allowed by the combinatorial system. Finally, cognition is *compositional* (a property, in their view, that may be a sub-phenomenon of systematicity) in that there are semantic links between cognitive representations that depend on the set of constituents appearing in them. For instance, from the fact that cats have paws and that panthers are cats, we can infer that paanthers have paws, and we can derive this relationship by rearranging the elements that occurred in the two premises in a systematic way.

As a preliminary characterization, this is a good start. It captures some essential characteristics of cognition that may be viewed as a series of benchmarks that a theory of cognition ought to satisfy. They go beyond this, however, and conjecture that any cognitive theory that satisfies the "benchmarks" are necessarily isomorphic to those systems, meaning that cognition, whether you express it with dots on a page that resolve into discrete letters, or as numbers making a pattern on an array of neurons, is basically symbolic computation with a change of data format. Two questions arise. How can standard symbolic operations be realized in the sorts of mechanisms we find in the brain? This is the Implementationalist Question. Second, are there specifically cognitive phenomena that require structures separate from classical deductive systems? Or phenomena that are relatively much more cumbersome to express in a classical theory relative to available alternatives? This is the Symbolic Describability Question: What is beyond the capacities of symbolic systems to capture readily? There are obvious examples of phenomena that are far more easily expressed in numerical theories than others—things like similarity relations, analogical reasoning, explaining prototype effects, etc. These would require

very onerous rule systems to express.

This dissertation examines and applies a series of models that enforce certain requirements of transparency between the symbolic level of computation and its neural, numerical counterpart. Optimality Theory and Harmonic Grammar [Prince and Smolensky, 2004, Smolensky and Legendre, 2006] provide the basis for these models in terms of their representational formats (varieties of tensor product representations) and computational procedures (structure assembly and optimization of the structures). In the context of neural networks, we refer to the process of associating symbols with structural roles as *binding*. After developing a framework for analyzing them, we fit the models to the task of knowledge base representation, evaluate them on benchmarks and demonstrate state-of-the-art performance.

Chapter 2 provides a common framework for the analysis of Tensor Product Representations/TPRs [Smolensky, 1990] and Holographic Reduced Representations/HRRs [Plate, 1995]—classical solutions to the "filler-role binding problem" [Fodor and Pylyshyn, 1988], each of which has appeared in knowledge base completion, e.g. [Lacroix et al., 2018a, Nickel et al., 2016, Vashishth et al., 2020]. These methods, and many others, all fit under the broader rubric of Multilinear Binding—in particular, Bilinear Binding—in which numerical representations of symbols are bound together using multilinear maps. We prove a number of theorems that not only unify these representational formats, but also show the sense in which they are optimal for structure representation subject to certain constraints, as well as demonstrating the equivalence of well-known models like HRRs to arbitrary bilinear maps, up to a certain level of description. This chapter enriches the traditional answers to the questions surrounding implementation: how, and how well, can compositional systems be realized in neural networks?

Chapter 3 gives theoretical context and a historical review for the second compu-

tational feature of the implemented models: optimization via Harmony Maximization, in which the neural network is designed so as to optimize an internal Harmony function in order to compute representations satisfying structural constraints. We show how to reframe certain structure-completion/structure-correction problems as numerical optimization problems, providing a basis for the subsequent models.

Chapter 4 implements the principles of Harmonic Grammar in a model—*Gradient Graphs*—that combines compositional representation of knowledge base entries with an optimization procedure based on knowledge constraints encoded in the form of a Harmony function. We ask: can knowledge graph representations assembled from discrete representations using compositional functions be augmented with a Harmony-maximization procedure? In the model, structured representations of the knowledge base are optimized with respect to learned semantic constraints, leading the constituents to be adjusted as a function of the other elements occurring in the structure. The mechanisms described therein provide a conceptual solution to linguistic phenomena like copredication and coercion (see Section 4.2.3), and relates to the Symbolic Describability Question: what phenomena can be more elegantly implemented in a numerical system than a symbolic one?

In Chapter 5, we examine knowledge graph representation using all of the principles developed in Chapters 1 and 2, developing *Harmonic Memory Networks* that build structures via sums of pairwise bindings of symbol components using bilinear maps, as well as the representation-optimization procedure. In addition to providing insight into the empirical performance of the binding methods specified in Chapter 1, the model has a number of practical virtues, including scalability due to the model being immediately deployable without retraining as the knowledge base is augmented with new facts.

Chapter 6 develops a model based on Tensor Product Representations that novelly

asks if the volumetric structure implicit in TPRs—with coordinates as 3-tuples of vector components—can be used as an organizing principle for knowledge representation. We use a spatial attention "searchlight" to draw attention to features in the TPR that are relevant to the context of particular queries. Changing the focus of attention in the 3-dimensional tensor, we show, provides a dynamic way of selecting features from multilinear representations of a knowledge entries—a model feature which is usually settled a priori. It also improves the performance on benchmarks.

Chapter 7 provides a summary.

Chapter 2

Models of filler-role binding

Since the origin of connectionist modeling, a literature has developed that describes explicit methods of implementing, in neural networks, the types of operations that classical computers perform, and therefore that feature heavily in classical computational theories of the mind. In this chapter, we briefly review the two most prominent of these methods—tensor product binding and binding via circular convolution-correlation, both of which have appeared in the knowledge base completion literature. We discuss their properties as binding operators, fit both into a broader framework of Bilinear Binding, and prove a number of features of convolution-correlation that render it—subject to certain constraints—optimal among bilinear binding methods for the purpose of structure-assembly in neural networks.

2.1 Tensor Product Binding

The basic requirements on a binding function are that it provide (1) a way of associating vectors with one another to assemble a vector representation of the associations (a

memory vector), and (2) a way of recovering the contents of the memory, i.e. an unbinding/decoding operation.[1] Schematically, a binding operation should provide a binding operation $\mathbb{B}$ and an unbinding operation $\mathbb{U}$ such that:

$$\mathbb{U}(\boldsymbol{r}, \mathbb{B}(\boldsymbol{r}, \boldsymbol{f})) \approx \boldsymbol{f}$$

In words, if a vector $\boldsymbol{r}$ and $\boldsymbol{f}$ are associated in memory, it should be possible to recover the latter by combining $\boldsymbol{r}$ and the memory $\mathbb{B}(\boldsymbol{r}, \boldsymbol{f})$. Since we are generally interested in representing structures that have multiple roles, we should like in addition that fillers be (approximately) retrievable when the structure in question has multiple roles. Thus, given a set $\{\langle \boldsymbol{r}_i, \boldsymbol{f}_i \rangle\}$ of pairwise associations of fillers and role, we desire some method of combining the individual bindings into a structure representing them all, while preserving the decodeability of the individual associations. That is, with Ψ representing an aggregation function applied to *sets* of vector associations, we want that:

$$\mathbb{U}(\boldsymbol{r}_\ell, \Psi\left(\{\mathbb{B}(\boldsymbol{r}_i, \boldsymbol{f}_i)\}\right) \approx \boldsymbol{f}_l \tag{2.1}$$

i.e. reversing the binding operation does not just work on a single vector association, but also on aggregated associations.

The tensor product was the first systematic structure-assembly method introduced to the connectionist representation of cognitive structures to enable fully distributed representations [Smolensky, 1990]. Tensor product binding takes a pair of vectors $\boldsymbol{r} \in \mathbb{R}^m$ (the role) and $\boldsymbol{f} \in \mathbb{R}^n$ (the filler) and returns a tensor indexed by the indices of the two vectors. The ij^{th} element of the tensor product $\boldsymbol{r} \otimes \boldsymbol{f}$ is simply the product of the i^{th}

[1] We shall use the terms *unbinding* and *decoding* interchangeably.

and j^{th} components of the constituent vectors.

$$[\boldsymbol{r} \otimes \boldsymbol{f}]_{ij} = r_i f_j \qquad \textbf{Tensor Product (definition)} \tag{2.2}$$

The aggregation function is summation of tensor products:

$$\Psi\left(\{\mathbb{B}(\boldsymbol{r}_i, \boldsymbol{f}_i)\}\right) = \sum_i \boldsymbol{r}_i \otimes \boldsymbol{f}_i \tag{2.3}$$

The corresponding unbinding operation is the dot product of the role vector with the tensor product. If T is a tensor of the right dimension, then the unbinding of the value in role $\boldsymbol{r}$ from T is:

$$\mathbb{U}(\boldsymbol{r}, T) = \boldsymbol{r} \cdot T \tag{2.4}$$

with its i^{th} component as

$$[\mathbb{U}(\boldsymbol{r}, T)]_i = \sum_j r_j T_{ji} \tag{2.5}$$

In certain conditions, it is possible to guarantee perfect retrieval of the vector associations, i.e. to ensure equality in Eqn. (2.1). Let $\mathcal{R}$ be a set of m roles, and $R = \{\boldsymbol{r}_i\} \subseteq \mathbb{R}^n$ a set of m vectors for each role, with $m < n$. If this set is linearly independent, then we can obtain a set $R^* = \{\boldsymbol{r}_i^*\}$ of dual vectors for R, i.e. vectors with the property that

$$\boldsymbol{r}_i^* \cdot \boldsymbol{r}_j = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases} \tag{2.6}$$

When this holds, there is an unbinding procedure that exactly recovers the vectors bound to each role vector $\boldsymbol{r}_i$: dotting the tensor product $T = \sum_j \boldsymbol{r}_j \otimes \boldsymbol{f}_j$ with the dual vector for the desired role. It follows from Eqn. (2.6) that, provided each role is bound to a single filler, the dot product $\boldsymbol{r}_i^* \cdot T$ outputs $\boldsymbol{f}_i$, the desired filler vector, without error.

In practice, suitable dual vectors can be obtained by arranging the role vectors in R into a matrix M_r and computing its pseudoinverse ${M_r}^\top \left(M_r {M_r}^\top\right)^{-1}$. The columns of this matrix are then the dual vectors for the corresponding roles. In a notable special case, one can use the role vectors $\{\boldsymbol{r}_i\}$ themselves, rather than computing dual vectors. When the vectors are orthonormal—i.e. when each vector has a norm of 1 and a zero dot product with every other vector in the set—then the vectors are self-dual, so that the memory can be queried using just the corresponding role vector.

When insisting on linear independence of the role vectors, representations built using the tensor product can become quite large. A set of k role vectors will require at least k dimensions in order for the vectors therein to be linearly independent, with the size of a tensor product representation with ℓ-dimensional fillers having $k \times \ell$ components. Whether these space requirements are tractable will depend on the setting, but in modern applied contexts where vector dimensions easily number in the hundreds, tensor product bindings can easily overtake hardware constraints. This has motivated the development of tensor product compression methods of the sort described in the next section. However, an often serviceable approximation can be obtained via normalization of role vectors even when these vectors are not linearly independent. When a vector is normalized, its dot product with itself is 1. The tensor product has the property that:

$$\boldsymbol{x} \cdot (\boldsymbol{y} \otimes \boldsymbol{z}) = (\boldsymbol{x} \cdot \boldsymbol{y})\boldsymbol{z}$$

where $\cdot$ is the dot product. In particular, $\boldsymbol{r} \cdot (\boldsymbol{r} \otimes \boldsymbol{f}) = \|\boldsymbol{r}\|^2 \boldsymbol{f}$, and if $\boldsymbol{r}$ has norm 1, the filler is recovered. When the tensor with which $\boldsymbol{r}$ is dotted is not a single tensor product but rather a sum, the result will depend on the similarity of $\boldsymbol{r}$ with the other role vectors which are intermingled in memory. However, if the number of bindings stored in any particular TPR is sufficiently small, and the dimensionality of the role vectors sufficiently capacious, then role vectors sampled from this set are unlikely to be too similar to one another to prevent effective decoding [Haley and Smolensky, 2020].

TPRs have convenient mathematical properties. First, they satisfy the principal requirements on any binding method, namely that they allow encoding and decoding of the contents of given structural positions within a vector-pattern that represents the structure. When the vectors are suitably chosen, the entire structure can be accessed without error. In addition, the TPR binding model maps into an intuitive "algebra of thought" that allows one to easily go between the numerical representations and the symbolic representations encoded in them. This ability to go between symbolic and neural representations, and the attendant prospects for unification of neural implementation with cognitive description, is indeed the basis for their use in the enterprise of Harmonic Grammar and related formalisms [Smolensky and Legendre, 2006].

2.2 Circular Convolution-Correlation and HRRs

The primary challenge for tensor product representations was already mentioned: the size of the required tensors. Holographic Reduced Representations (HRRs) were introduced by [Plate, 1994] as a model of vector computation capable of representing compositional and recursive structure under very strict resource constraints. They are naturally viewed as a form of compressed tensor product, and as such, approximate many properties of

TPRs. From the point of view of the binding problem, they have two properties that have attracted sustained attention. The first is that HRRs can be embedded within other HRRs, enabling recursive representations and hierarchical structure. The second property is that binding two vectors using HRRs leads to a vector that is of the same dimension as its two inputs. Though the expected decoding error grows both with the number of associated pairs, and also with the depth of the structure encoded, this error can, under certain additional assumptions, be treated as noise with a simple distribution, making it easy to reason about how these errors would behave. The cost in decoding error is offset by HRRs' efficient use of spatial memory.

HRRs are based on the operation of *circular convolution*, a variety of vector multiplication that can be used to associate pairs of vectors and store a number of such pairs in memory. Consider two n-dimensional vectors $\mathbf{x}$ and $\mathbf{y}$.

Definition: Circular convolution. The jth element of the circular convolution $\boldsymbol{x} \circledast \boldsymbol{y}$ is

$$[\boldsymbol{x} \circledast \boldsymbol{y}]_j = \sum_{k=0}^{n-1} \mathrm{x}_k \mathrm{y}_{j-k}$$

where the subscripts are computed modulo n.

Algebraic properties: Circular convolution. The circular convolution is associative, commutative, and linear in each argument:

$$\begin{aligned} \textbf{Associative} \quad & \boldsymbol{x} \circledast (\boldsymbol{y} \circledast \boldsymbol{z}) = (\boldsymbol{x} \circledast \boldsymbol{y}) \circledast \boldsymbol{z} \\ \textbf{Commutative} \quad & \boldsymbol{x} \circledast \boldsymbol{y} = \boldsymbol{y} \circledast \boldsymbol{x} \\ \textbf{Linear} \quad & \boldsymbol{x} \circledast (a\boldsymbol{y} + b\boldsymbol{z}) = a\boldsymbol{x} \circledast \boldsymbol{y} + b\boldsymbol{x} \circledast \boldsymbol{z} \end{aligned}$$

A simple way to visualize circular convolution is to imagine it as a wrapped sliding dot product. Each component of the circular convolution of $\boldsymbol{x}$ and $\boldsymbol{y}$ is a dot product

of elements of $\boldsymbol{x}$ and $\boldsymbol{y}$. First, the vector $\boldsymbol{y}$ is *involuted* by reversing all of its elements except for the first entry.

Definition: Involution. The ith component of the involution $\tilde{\boldsymbol{y}}$ of the n-dimensional vector $\boldsymbol{y}$ is

$$[\tilde{\boldsymbol{y}}]_i \equiv \tilde{\mathrm{y}}_i \equiv \mathrm{y}_{-i}$$

modulo n. In the 5-dimensional example:

$$\boldsymbol{y} = [\mathrm{y}_0 \quad \mathrm{y}_1 \quad \mathrm{y}_2 \quad \mathrm{y}_3 \quad \mathrm{y}_4] \longmapsto [\mathrm{y}_0 \quad \mathrm{y}_4 \quad \mathrm{y}_3 \quad \mathrm{y}_2 \quad \mathrm{y}_1] \equiv \tilde{\boldsymbol{y}}$$

The particular dot product to be taken at the jth component is obtained by sliding $\tilde{\boldsymbol{y}}$ to the right by j components. When $\tilde{\boldsymbol{y}}$ reaches the end of $\boldsymbol{x}$, there are still $n-j$ unmatched components of $\tilde{\boldsymbol{y}}$. These are matched to the first $n-j$ components of $\boldsymbol{x}$.

As an example, consider two five-dimensional vectors $\boldsymbol{x}$ and $\boldsymbol{y}$. Their circular convolution is a 5-dimensional vector composed of five dot products:

$$[\boldsymbol{x} \circledast \boldsymbol{y}]_0 = \begin{matrix}[\mathrm{x}_0 & \mathrm{x}_1 & \mathrm{x}_2 & \mathrm{x}_3 & \mathrm{x}_4] \\ [\mathrm{y}_0 & \mathrm{y}_4 & \mathrm{y}_3 & \mathrm{y}_2 & \mathrm{y}_1]\end{matrix} = \mathrm{x}_0\mathrm{y}_0 + \mathrm{x}_1\mathrm{y}_4 + \mathrm{x}_2\mathrm{y}_3 + \mathrm{x}_3\mathrm{y}_2 + \mathrm{x}_4\mathrm{y}_1$$

$$[\boldsymbol{x} \circledast \boldsymbol{y}]_1 = \begin{matrix}[\mathrm{x}_0 & \mathrm{x}_1 & \mathrm{x}_2 & \mathrm{x}_3 & \mathrm{x}_4] \\ \mathrm{y}_1] & [\mathrm{y}_0 & \mathrm{y}_4 & \mathrm{y}_3 & \mathrm{y}_2\end{matrix} = \mathrm{x}_0\mathrm{y}_1 + \mathrm{x}_1\mathrm{y}_0 + \mathrm{x}_2\mathrm{y}_4 + \mathrm{x}_3\mathrm{y}_3 + \mathrm{x}_4\mathrm{y}_2$$

$$[\boldsymbol{x} \circledast \boldsymbol{y}]_2 = \begin{matrix}[\mathrm{x}_0 & \mathrm{x}_1 & \mathrm{x}_2 & \mathrm{x}_3 & \mathrm{x}_4] \\ \mathrm{y}_2 & \mathrm{y}_1] & [\mathrm{y}_0 & \mathrm{y}_4 & \mathrm{y}_3\end{matrix} = \mathrm{x}_0\mathrm{y}_2 + \mathrm{x}_1\mathrm{y}_1 + \mathrm{x}_2\mathrm{y}_0 + \mathrm{x}_3\mathrm{y}_4 + \mathrm{x}_4\mathrm{y}_3$$

$$[\boldsymbol{x} \circledast \boldsymbol{y}]_3 = \begin{matrix}[\mathrm{x}_0 & \mathrm{x}_1 & \mathrm{x}_2 & \mathrm{x}_3 & \mathrm{x}_4] \\ \mathrm{y}_3 & \mathrm{y}_2 & \mathrm{y}_1] & [\mathrm{y}_0 & \mathrm{y}_4\end{matrix} = \mathrm{x}_0\mathrm{y}_3 + \mathrm{x}_1\mathrm{y}_3 + \mathrm{x}_2\mathrm{y}_1 + \mathrm{x}_3\mathrm{y}_0 + \mathrm{x}_4\mathrm{y}_4$$

$$[\boldsymbol{x} \circledast \boldsymbol{y}]_4 = \begin{matrix} [\mathrm{x}_0 & \mathrm{x}_1 & \mathrm{x}_2 & \mathrm{x}_3 & \mathrm{x}_4] \\ \mathrm{y}_4 & \mathrm{y}_3 & \mathrm{y}_2 & \mathrm{y}_1] & [\mathrm{y}_0 \end{matrix} = \mathrm{x}_0\mathrm{y}_4 + \mathrm{x}_1\mathrm{y}_3 + \mathrm{x}_2\mathrm{y}_2 + \mathrm{x}_3\mathrm{y}_1 + \mathrm{x}_4\mathrm{y}_0$$

Iterating through each of the 5 components, the vector $\boldsymbol{x}$ is "pushed" along $\boldsymbol{y}$ by one component, and any remaining at the right edge of $\tilde{\boldsymbol{x}}$ are wrapped back around to the beginning of the vector $\boldsymbol{y}$.

As a network. Similar to tensor product representations, binding in HRRs can be realized as weight matrices, specific to a given role, that receive a filler vector and assign it to the role. A role-specific weight matrix $\mathbb{E}_{\boldsymbol{r}}$ encoding the filler-role binding can be defined in terms of the entries of the role vector $\boldsymbol{r}$, with

$$[\mathbb{E}_{\boldsymbol{r}}]_{jk} = \mathrm{r}_{j-k}$$

again computing the difference $k - j$ modulo n.

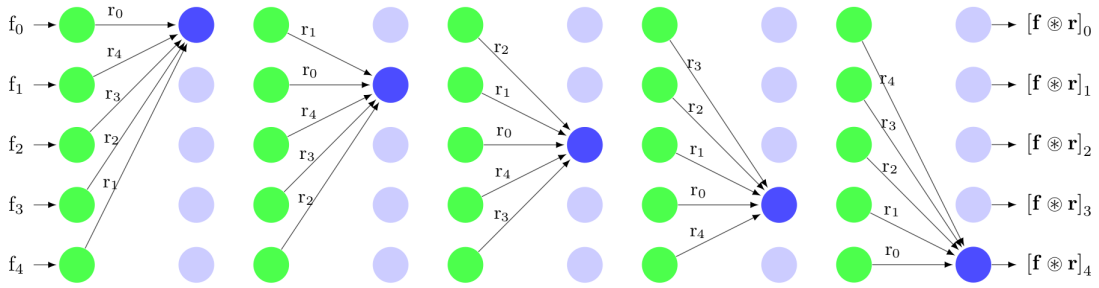


Figure 2.1: Synaptic weights of a 5-dimensional circular convolution (𝔹**inding**) network.

Computing the circular convolution. The naive procedure for computing HRRs is to take the n^2 products occurring in Definition (5.2) and then sum them: n^3 operations. However, HRRs as a binding mechanism are particularly efficient because the circular convolution can be computed in frequency space using the discrete Fourier transform. Where $\mathcal{F}$ is the Fourier transform, $\mathcal{F}^{-1}$ its inverse, and $\odot$ elementwise multiplication, the

convolution can be computed as:

$$\boldsymbol{x} \circledast \boldsymbol{y} = \mathcal{F}^{-1}\left(\mathcal{F}(\boldsymbol{x}) \odot \mathcal{F}(\boldsymbol{y})\right) \tag{2.7}$$

with time complexity $\mathcal{O}(n \log n)$. The unbinding operation also has an efficient algorithm, which we deal with later.

2.2.1 Properties as a binding operator

Unbinding. A vector $\boldsymbol{f}$ bound to role $\boldsymbol{r}$ may be unbound from $\boldsymbol{f} \circledast \boldsymbol{r}$ by using the circular correlation, denoted by the operator $\star$.

Definition: Circular correlation. The jth element of the circular correlation $\boldsymbol{x} \star \boldsymbol{y}$ is

$$[\boldsymbol{x} \star \boldsymbol{y}]_j = \sum_{k=0}^{n-1} \mathrm{x}_k \mathrm{y}_{k+j} \tag{2.8}$$

The subscripts are again calculated modulo n. Correlation can be computed quickly in the frequency domain using the Fourier transform:

$$\boldsymbol{x} \star \boldsymbol{y} = \mathcal{F}^{-1}\left(\overline{\mathcal{F}(\boldsymbol{x})} \odot \mathcal{F}(\boldsymbol{y})\right) \tag{2.9}$$

where $\overline{\mathcal{F}(\boldsymbol{x})}$ is the complex conjugate of $\mathcal{F}(\boldsymbol{x})$, which is obtained by inverting sign of the complex component of $\mathcal{F}(\boldsymbol{x})$.

Like the convolution, circular correlation can be visualized as a wrapped dot product. It is again useful to work through the 5-dimensional example. This time, the vector $\boldsymbol{x}$ is

directly slid along $\boldsymbol{y}$, and there is no involution.

$$
\begin{aligned}
[\boldsymbol{x} \star \boldsymbol{y}]_0 &= \begin{matrix}[\mathrm{x}_0 & \mathrm{x}_1 & \mathrm{x}_2 & \mathrm{x}_3 & \mathrm{x}_4] \\ [\mathrm{y}_0 & \mathrm{y}_1 & \mathrm{y}_2 & \mathrm{y}_3 & \mathrm{y}_4]\end{matrix} &&= \mathrm{x}_0\mathrm{y}_0 + \mathrm{x}_1\mathrm{y}_1 + \mathrm{x}_2\mathrm{y}_2 + \mathrm{x}_3\mathrm{y}_3 + \mathrm{x}_4\mathrm{y}_4 \\
[\boldsymbol{x} \star \boldsymbol{y}]_1 &= \begin{matrix}\mathrm{x}_4] & [\mathrm{x}_0 & \mathrm{x}_1 & \mathrm{x}_2 & \mathrm{x}_3 \\ [\mathrm{y}_0 & \mathrm{y}_1 & \mathrm{y}_2 & \mathrm{y}_3 & \mathrm{y}_4]\end{matrix} &&= \mathrm{x}_4\mathrm{y}_0 + \mathrm{x}_0\mathrm{y}_1 + \mathrm{x}_1\mathrm{y}_2 + \mathrm{x}_2\mathrm{y}_3 + \mathrm{x}_3\mathrm{y}_4 \\
[\boldsymbol{x} \star \boldsymbol{y}]_2 &= \begin{matrix}\mathrm{x}_3 & \mathrm{x}_4] & [\mathrm{x}_0 & \mathrm{x}_1 & \mathrm{x}_2 \\ [\mathrm{y}_0 & \mathrm{y}_1 & \mathrm{y}_2 & \mathrm{y}_3 & \mathrm{y}_4]\end{matrix} &&= \mathrm{x}_3\mathrm{y}_0 + \mathrm{x}_4\mathrm{y}_1 + \mathrm{x}_0\mathrm{y}_2 + \mathrm{x}_1\mathrm{y}_3 + \mathrm{x}_2\mathrm{y}_4 \\
[\boldsymbol{x} \star \boldsymbol{y}]_3 &= \begin{matrix}\mathrm{x}_2 & \mathrm{x}_3 & \mathrm{x}_4] & [\mathrm{x}_0 & \mathrm{x}_1 \\ [\mathrm{y}_0 & \mathrm{y}_1 & \mathrm{y}_2 & \mathrm{y}_3 & \mathrm{y}_4]\end{matrix} &&= \mathrm{x}_2\mathrm{y}_0 + \mathrm{x}_3\mathrm{y}_1 + \mathrm{x}_4\mathrm{y}_2 + \mathrm{x}_0\mathrm{y}_3 + \mathrm{x}_1\mathrm{y}_4 \\
[\boldsymbol{x} \star \boldsymbol{y}]_4 &= \begin{matrix}\mathrm{x}_1 & \mathrm{x}_2 & \mathrm{x}_3 & \mathrm{x}_4] & [\mathrm{x}_0 \\ [\mathrm{y}_0 & \mathrm{y}_1 & \mathrm{y}_2 & \mathrm{y}_3 & \mathrm{y}_4\end{matrix} &&= \mathrm{x}_1\mathrm{y}_0 + \mathrm{x}_2\mathrm{y}_1 + \mathrm{x}_3\mathrm{y}_2 + \mathrm{x}_4\mathrm{y}_3 + \mathrm{x}_0\mathrm{y}_4
\end{aligned}
$$

The network realization of correlation as a linear neural network (for a given role symbol $\boldsymbol{r}$) is similar. The decoding matrix $\mathbb{D}$ has $[\mathbb{D}_{\boldsymbol{r}}]_{ij} = \mathrm{r}_{j+k}$, so that

$$[\mathbb{D}_{\boldsymbol{xy}}]_j = \sum_{k=0}^{n} \mathrm{x}_k \mathrm{y}_{j+k} = [\boldsymbol{x} \star \boldsymbol{y}]_j$$

In the 5-dimensional case, this is visualized as Figure 2.2. To unbind filler $\boldsymbol{f}$ from role $\boldsymbol{r}$, compute:

$$\boldsymbol{r} \star (\boldsymbol{f} \circledast \boldsymbol{r}) = \boldsymbol{f} + \boldsymbol{\varepsilon} \approx \boldsymbol{f}$$

Unlike with TPRs when the set of roles is linearly independent, this unbinding operation is only approximate. The term $\boldsymbol{\varepsilon}$ in the unbound product is Gaussian under several

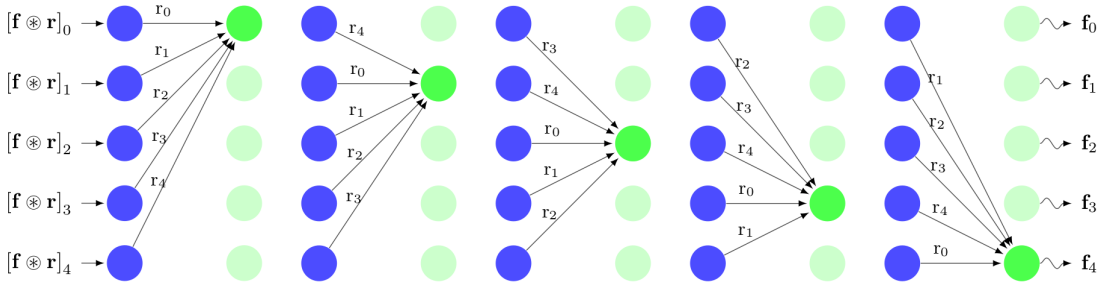


Figure 2.2: Synaptic weights of a 5-dimensional circular correlation ($\mathbb{U}$**nbinding**) network. In the output, the filler $\boldsymbol{f}$ is superposed with noise.

conditions, discussed in [Plate, 1994] and below. In the case of continuous-valued filler and role vectors, it is sufficient for the components of each n-dimensional vector to be independently and identically drawn from the normal distribution with variance $\frac{1}{d}$. The expected norm of each vector is then 1.

We will now specify the sense in which correlation-decoding is an *optimal* approximation.

2.3 Convolution-Correlation binding as special cases of Bilinear binding

While it is well known that the circular correlation and circular convolution serve as approximate inverses of one another—e.g. [Plate, 1995, Nickel et al., 2016]—there is a precise sense in which this is true. We show that the circular convolution and correlation (holographic reduced representations/HRR binding schemes) correspond to procedures for compressing (convolution) and decompressing (correlation) tensor product representations, with the correlation equal to the Moore-Penrose Inverse of a map from the .ensor product space to the d-dimensional space in which the input vectors reside. Thus, the

correlation optimally retrieves a $d \times d$-dimensional TPR from its compression as a d-dimensional vector, specifically with respect to the convolution. Furthermore, this places these operations in the broader setting of *bilinear binding*, of which TPR and HRR binding are special cases. Indeed, any function of two vectors $\boldsymbol{x}, \boldsymbol{y}$ characterized by a linear map on the tensor product of those vectors (a bilinear map), i.e. with the form (2.10),

$$f(\boldsymbol{x}, \boldsymbol{y}) = W\boldsymbol{x} \otimes \boldsymbol{y} \tag{2.10}$$

with $W \in \mathbb{R}^o \otimes \mathbb{R}^n \otimes \mathbb{R}^m$ and the dot product along the trailing dimensions of W, can be rewritten as an encoding and decoding of the tensor product $\boldsymbol{x} \otimes \boldsymbol{y}$, with the Moore-Penrose Inverse providing an unbinding procedure for an arbitrary compression map W.

The circular convolution (forward/encoding) map M for dimension d is defined as:

$$[M]_{kij} = \begin{cases} 1 & \text{if } j = (k - i) \mod d \\ 0 & \text{otherwise} \end{cases} \qquad \textbf{Convolution}$$

The corresponding correlation (backward/decoding) map N is:

$$[N]_{ijk} = \begin{cases} 1 & \text{if } k = (i + j) \mod d \\ 0 & \text{otherwise} \end{cases} \qquad \textbf{Correlation}$$

It is easily verified that these tensor-elementwise conditions are equivalent to the vector-elementwise definitions of the two operations:

$$[\boldsymbol{x} \circledast \boldsymbol{y}]_k = \sum_i x_i y_{(i-k) \bmod d}$$

$$[\boldsymbol{x} \star \boldsymbol{z}]_j = \sum_i x_i z_{(i+k) \bmod d} \qquad \text{from [Plate, 1995]}$$

Specifically,

Theorem 2.3.1. *Where for each index ℓ, $\cdot_\ell$ refers to the dot product along the ℓ index of M, we have that*

$$(M \cdot_i \boldsymbol{x}) \cdot_j \boldsymbol{y} = \boldsymbol{x} \circledast \boldsymbol{y} \tag{2.11}$$

and also,

$$(N \cdot_i \boldsymbol{x}) \cdot_k \boldsymbol{z} = \boldsymbol{x} \star \boldsymbol{z} \tag{2.12}$$

Proof. The dot product (2.11) is:

$$[(M \cdot_i \boldsymbol{x}) \cdot_j \boldsymbol{y}]_k = \sum_{ij} [M]_{kij} x_i y_j$$

First, note that for a given k, the j such that $[M]_{kij} = 1$ is a function of i. call it g_k, which allows us to specify $g_k(i) = (i - k) \mod d$. Thus:

$$\begin{aligned}
[(M \cdot_i \boldsymbol{x}) \cdot_j \boldsymbol{y}]_k &= \sum_{ij} [M]_{kij} x_i y_j \\
&= \sum_i x_i y_{g_k(i)} \\
&= \sum_i x_i y_{(i-k) \bmod d} \\
&= [\boldsymbol{x} \circledast \boldsymbol{y}]_k
\end{aligned}$$

The proof of (2.12) is similar. k such that $k = (i + j) \bmod d$ is a function $h_i(j) = k$. Therefore:

$$\begin{aligned}
[(N \cdot_i \boldsymbol{x}) \cdot_k \boldsymbol{z}]_j &= \sum_{ik} [N]_{jik} x_i z_k \\
&= \sum_i x_i z_{h_i(j)} \\
&= \sum_i x_i z_{(i+j) \bmod d} \\
&= [\boldsymbol{x} \star \boldsymbol{z}]_j
\end{aligned}$$

□

To simplify the following proofs, we have selected an indexation scheme based on ensuring consistency of the indices for the inputs and outputs corresponding to each operation (convolution followed by correlation). Thne notation is suggestive of the context of filler-role binding, but note in passing that the mathematical results themselves are independent of this setting. The intuition is as follows: Let $\boldsymbol{r}^{(i)}, \boldsymbol{f}^{(j)}, \boldsymbol{o}^{(k)}$ stand for the inputs and outputs of convolution and correlation, which are indexed by i, j, k. Starting with convolution ("binding"), $\boldsymbol{r}$ denotes the first input, $\boldsymbol{f}$ the second input, and $\boldsymbol{o}$ the output $\boldsymbol{o} \equiv \boldsymbol{r} \circledast \boldsymbol{f}$. Thus, M leads with the output dimension k, followed by the input dimensions i and j. With respect to correlation ("unbinding"), the inputs are $\boldsymbol{r}$ (the role) and $\boldsymbol{o}$ (the binding), with the filler as the output. Thus, N is indexed as jik. The matrix product NM is shorthand for $N \cdot_k M$, i.e. the dot product of the leading axis of M with the trailing axis of N. The theorem to be proven is:

$$NM\boldsymbol{r} \otimes \boldsymbol{f} \equiv N\boldsymbol{o} \approx \boldsymbol{r} \otimes \boldsymbol{f} \tag{2.13}$$

where the approximation $\approx$ means that MN minimizes the expected retrieval error specified in Theorem 2.3.3 below, taken across all choices of $\boldsymbol{r}$ and $\boldsymbol{f}$. This formulation of N and M as linear maps on the tensor product space $R \otimes F$ is equivalent to the dot products 2.11 and 2.12, i.e. M and N are *bilinear* on R and F. Note that this is a novel and more precise statement of the approximation first explored by Plate's dissertation, as expounded in Section 2.3.2 below.

Plate stipulates the following conditions on the input vectors: they are i.i.d. with componentwise mean of zero and variance $\frac{1}{d}$ (standard deviation of $\frac{1}{\sqrt{d}}$), meaning that, when dealing with vectors having unit variance, the binding function is:

$$\left(\boldsymbol{r}/\sqrt{d}\right) \circledast \left(\boldsymbol{f}/\sqrt{d}\right) = M\left(\boldsymbol{r} \otimes \boldsymbol{f}\right)/d \equiv \boldsymbol{o}$$

We use the notation of (2.10) above for products of the bilinear map M with tensor products. In decoding, the role vector must also be transformed, and then, to reset the output to unit variance, the transformation inverted by multiplying with $\sqrt{d}$, yielding:

$$\begin{aligned}
\sqrt{d}\left(\left(\boldsymbol{r}/\sqrt{d}\right) \star \boldsymbol{o}\right) &= \sqrt{d}\left(\left(\boldsymbol{r}/\sqrt{d}\right) \cdot_i \left(N \cdot_k \boldsymbol{o}\right)\right) \\
&= \boldsymbol{r} \cdot_i N\boldsymbol{o} \\
&= \boldsymbol{r} \cdot_i \left(N \cdot_k \left(M\left(\boldsymbol{r} \otimes \boldsymbol{f}\right)/d\right)\right) \\
&= \boldsymbol{r} \cdot_i \left(NM\left(\boldsymbol{r} \otimes \boldsymbol{f}\right)\right)/d
\end{aligned}$$

The same result can be obtained for vectors with unit variance by packing the variance-compression conditions into the binding and unbinding maps, e.g. by dividing N and M

each by $\sqrt{d}$. This yields:

$$[M]_{kij} = \begin{cases} \frac{1}{\sqrt{d}} & \text{if } j = (k-i) \bmod d \\ 0 & \text{otherwise} \end{cases} \qquad \textbf{Convolution} \tag{2.14}$$

$$[N]_{ijk} = \begin{cases} \frac{1}{\sqrt{d}} & \text{if } k = (i+j) \bmod d \\ 0 & \text{otherwise} \end{cases} \qquad \textbf{Correlation} \tag{2.15}$$

We now demonstrate that N is the Moore-Penrose inverse (MPI) of M, i.e. $N = M^\top \left(MM^\top\right)^{-1}$, where transposition is of the first dimension holding the i and j axes fixed, i.e. $[M]_{kij} = [M^\top]_{ijk}$.

Theorem 2.3.2. *The **circular correlation tensor** N—a linear map $\mathbb{R}^d \to \mathbb{R}^d \otimes \mathbb{R}^d$—is the MPI of the **circular convolution tensor** $M : \mathbb{R}^d \otimes \mathbb{R}^d \to \mathbb{R}^d$.*

Proof. First, note that

$$[MM^\top]_{k\ell} = \sum_{ij} [M]_{kij}[M]_{\ell ij}$$

Setting $[M]_{kij} = \frac{1}{\sqrt{d}} = [M]_{\ell ij}$ implies that $j = (k-i) \bmod d = (\ell - i) \bmod d$. This is true if $k = \ell$, and *only* when $k = \ell$.[2] Thus, the off-diagonal elements of $MM^\top$ are zero. Now, take any given k. How many terms in the row-wise sum are nonzero (and thus equal to

[2]Because if $j = (k-i) \bmod d = (\ell - i) \bmod d$ for potentially distinct k and ℓ, then $j = md + k - i = nd + \ell - i$ for some m and n, and therefore $(*)$: $md + k = nd + \ell$.

Given that j, i, k, ℓ are bounded by the integer interval $[1, d]$, m and n must be either 0 or 1. For if $m < 0$ (with -1 the limiting case), then j is at most $-d + k - i$ with i at least 1 and k at most d, whence $j \leq -d + d - 1$, with j a negative integer $\perp$. If on the other hand $m > 1$, with 2 the limiting case, then $j \geq 2d + k - i$ with k at least 1 and i at most d, so that $j \geq d + 1$ $\perp$. The proof for n is the same.

Now suppose $m \neq n$. There is one case without loss of generality: $m = 0, n = 1$. Plugging into $(*)$ above, it follows that $-k = d - \ell$, with $d - \ell$ a negative integer, which is impossible since ℓ is at most d $\perp$.

The one remaining case has $m = n$ and therefore $k = \ell$, as claimed.

$\frac{1}{\sqrt{d}}$)? There is one and only one such term for each value of j. [3] Thus, we remove the i index by expressing it as a function $g_k(j)$ of j, writing

$$\begin{aligned}[MM^\top]_{kk} &= \sum_{j=1}^{d} [M]_{kg_k(j)j}[M]_{kg_k(j)j} \\ &= \sum_{j=1}^{d} \frac{1}{\sqrt{d}}\frac{1}{\sqrt{d}} = 1\end{aligned}$$

This proves each diagonal element is 1. Thus, $MM^\top = I$, and M has full row rank, with right inverse $M^\top$.

Observe furthermore that $N = M^\top$. Take $[M^\top]_{ijk} = [M]_{kij}$. There are two cases. If $[M^\top]_{ijk} = \frac{1}{\sqrt{d}}$, then $i = (j - k) \bmod d$. There are two cases. if $[M^\top]_{ijk} = \frac{1}{\sqrt{d}}$, then $j = (k - i) \bmod d$ and therefore $j = md + k - i$ for some m. This implies that $k = -md + i + j$ and therefore $k = (i + j) \bmod d$, which is the condition for $[N]_{ijk} = \frac{1}{\sqrt{d}}$ 2.15. If, on the other hand, $[M^\top]_{ijk} = 0$, then there is no such m. If we suppose $[N]_{ijk} \neq 0$, then $k = (i + j) \bmod d$, from which we have that $k = (i + j) \bmod d$ and therefore $k = nd + i + j$. This implies that $j = -nd + k - i$, with $-n$ a counterexample to m in the preceding sentence $\perp$. Thus, $M^\top(MM^\top)^{-1} = M^\top I = M^\top = N$, as claimed, and N is the MPI of M. □

N therefore partakes of the properties of the Moore-Penrose Inverse—in particular, the following optimality relation:

Theorem 2.3.3. *The circular correlation is the* ***maximum-fidelity*** *decompressor for a TPR compressed as* $\boldsymbol{r} \circledast \boldsymbol{f} = M\boldsymbol{r} \otimes \boldsymbol{f}$*, in the sense that it minimizes the expected decoding*

[3] To verify this, suppose that $j = (i - k) \bmod \mathrm{d} = (i' - k)$ for potentially distinct i and i'. Then $nd + i - k = md + i' - k$ for some m, n which must be equal to one another (footnote 2). So, $i = i'$.

error (2.16).

$$\mathcal{D}(N, M) = \mathbb{E}\left[\frac{1}{2}\left\|\boldsymbol{r} \otimes \boldsymbol{f} - NM\boldsymbol{r} \otimes \boldsymbol{f}\right\|^2\right] \tag{2.16}$$

given that the following assumptions are satisfied: both $\boldsymbol{r}$ *and* $\boldsymbol{f}$ *are zero-mean and are each have unit isotropic covariance.*

Proof. To simplify the notation, let $\boldsymbol{x} \equiv \boldsymbol{r} \otimes \boldsymbol{f}$. Setting $\partial\mathcal{D}(N, M)/\partial N = \mathcal{Z}$ (with $\mathcal{Z}$ the zero matrix) yields:

$$\begin{aligned}
\mathcal{Z} &= \frac{\partial}{\partial N}\mathbb{E}\left[\frac{1}{2}\left(\boldsymbol{x}^\top\boldsymbol{x} - 2\boldsymbol{x}^\top NM\boldsymbol{x} + \boldsymbol{x}^\top M^\top N^\top NM\boldsymbol{x}\right)\right] \\
&= \mathbb{E}\left[NM\boldsymbol{x}\boldsymbol{x}^\top M^\top - \boldsymbol{x}\boldsymbol{x}^\top M^\top\right] \\
NM\mathbb{E}\left[\boldsymbol{x}\boldsymbol{x}^\top\right] M^\top &= \mathbb{E}\left[\boldsymbol{x}\boldsymbol{x}^\top\right] M^\top \\
N &= \mathbb{E}\left[\boldsymbol{x}\boldsymbol{x}^\top\right] M^\top \left(M\mathbb{E}\left[\boldsymbol{x}\boldsymbol{x}^\top\right] M^\top\right)^{-1}
\end{aligned}$$

To complete the proof, it suffices to show that $\mathbb{E}\left[\boldsymbol{x}\boldsymbol{x}^\top\right] = I$. We assume that $\boldsymbol{r}$ and $\boldsymbol{f}$ are zero-mean and componentwise i.i.d. with unit variance, and are also independent of each other. The expected value of $\boldsymbol{x}\boldsymbol{x}^\top$ is $\mathbb{E}\left[\boldsymbol{r} \otimes \boldsymbol{f} \otimes \boldsymbol{f} \otimes \boldsymbol{r}\right]$ (modulo vectorization). From these distributional assumptions,

$$\mathbb{E}\left[\boldsymbol{r} \otimes \boldsymbol{f} \otimes \boldsymbol{f} \otimes \boldsymbol{r}\right]_{ijk\ell} = \begin{cases} \mathbb{E}\left[r_i\right]\mathbb{E}\left[r_k\right]\mathbb{E}\left[f_j f_\ell\right] = 0 & \text{if } i \neq k \text{ (independence)} \\ \mathbb{E}\left[r_i r_k\right]\mathbb{E}\left[f_j\right]\mathbb{E}\left[f_\ell\right] = 0 & \text{if } j \neq \ell \text{ (independence)} \\ \mathbb{E}\left[{r_i}^2\right]\mathbb{E}\left[{f_k}^2\right] = {\sigma_{r_i}}^2{\sigma_{f_k}}^2 = 1 & \text{if } i = k \text{ and } j = \ell \end{cases}$$

Thus, $\mathbb{E}\left[\boldsymbol{r} \otimes \boldsymbol{f} \otimes \boldsymbol{f} \otimes \boldsymbol{r}\right]_{ijk\ell} = 0$ everywhere except the diagonal, where it is 1. □

It is clear that Theorem 2.3.3 generalizes for arbitrary encoding maps M, with the optimal N always the MPI, given appropriate distributional constraints on the input vectors. The most obvious special case is the tensor product itself, where M is just the identity map on the tensor space $\mathcal{R} \otimes \mathcal{F}$, which also provides the inverse map. Crucially, though, there are in general no constraints on the size of the input vectors or the size of the encoded output, as there are in the special case of HRRs, meaning that TPR-compression provides a general mechanism for binding schemes. The forward map takes the TPR and maps it linearly into a lower-dimensional representation. The backward map—the MPI—optimally recovers the TPR, from which standard TPR unbinding operations (dot products with role vectors) can proceed as usual.

A brief note: the situation does not materially change when we are dealing with *sums* of tensor products rather than tensor products themselves. As long as the individual FRBs are uncorrelated, summing TPRs has the effect of simply dilating the correlation matrix $\Omega \equiv \mathbb{E}\left[\boldsymbol{x}\boldsymbol{x}^\top\right]$ from unit isotropy to an isotropy scaled by the number of summands.[4] The scaling b factor cancels in the matrix inversion for the MPI, so that the decoder becomes $\mathbb{E}\left[\boldsymbol{x}\boldsymbol{x}^\top\right] M^\top \left(M\mathbb{E}\left[\boldsymbol{x}\boldsymbol{x}^\top\right] M^\top\right)^{-1} = bIM^\top \left(bMIM^\top\right)^{-1} = b\frac{1}{b}M^\top \left(MM^\top\right)^{-1} = \text{MPI}(M)$.

This characterization has some interesting consequences for Plate-binding. HRRs come with particular computational advantages—namely, that they can be computed quickly—$\mathcal{O}(d \log d)$—using frequency-domain calculations [Plate, 1994]. Beyond this, are there reasons for preferring the HRR operations—among, say, all bilinear binding

[4]Let the summed FRBs be indexed by m, with each superposed binding $\boldsymbol{x} = \sum_m x^{(m)} = \sum_m \boldsymbol{r}^{(m)} \otimes \boldsymbol{f}^{(m)}$. The entries of the covariance are $[\Omega]_{ij} = \mathbb{E}\left[\left(\sum_m x_i^{(m)}\right)\left(\sum_n x_j^{(n)}\right)\right] = \sum_{mn} \mathbb{E}\left[x_i^{(m)} x_j^{(n)}\right]$. For the off-diagonal elements ($i \neq j$), componentwise independence allow us to rewrite $[\Omega]_{ij} = \sum_{mn} \mathbb{E}\left[x_i^{(m)} x_i^{(n)}\right] = 0$ for all summands. For $i = j$, we have $[\Omega]_{ii} = \sum_m \mathbb{E}\left[x_i^{(m)} x_i^{(m)}\right] + \sum_{m \neq n} x_i^{(m)} x_i^{(n)}$ with the second terms all zero and the terms with $m = n$ the variance of each component of $\boldsymbol{x}$. Together, then, $\Omega = bI$ where b is the number of bindings.

operations obeying the constraint $W : \mathbb{R}^d \otimes \mathbb{R}^d \to \mathbb{R}^d$—on the grounds of expected decoding accuracy (2.16)? In fact, any matrix with full row rank will do.

Theorem 2.3.4. *The decoding error $\mathcal{D}(M, N)$, which is minimized with respect to N (i.e. for fixed M) as $\mathcal{D}(M, M^\top(MM^\top)^{-1}) \equiv \widehat{\mathcal{D}}(M)$, is minimized with respect to both encoding and decoding by any encoding matrix M with full row rank.*

Proof. As a corollary of 2.3.3, for any given encoding matrix M, X has a unique decoding matrix N that minimizes $\mathcal{D}(M, N)$. Thus, we can express the decoding error, minimized with respect to Y, as a function of M:

$$\begin{aligned}\widehat{\mathcal{D}}(X) &= \mathbb{E}\left[\frac{1}{2}\left(\boldsymbol{x}^\top\boldsymbol{x} - 2\boldsymbol{x}^\top NM\boldsymbol{x} + \boldsymbol{x}^\top M^\top N^\top NM\boldsymbol{x}\right)\right] \\ &= \mathbb{E}\left[\frac{1}{2}\left(\boldsymbol{x}^\top\boldsymbol{x} - 2\boldsymbol{x}^\top M^\top(MM^\top)^{-1}M\boldsymbol{x} + \boldsymbol{x}^\top M^\top(MM^\top)^{-1}MM^\top(MM^\top)^{-1}M\boldsymbol{x}\right)\right]\end{aligned}$$

If M has full row rank so that $MM^\top$ is positive definite and $(MM^\top)^{-1}MM^\top = I$, this simplifies to:

$$\begin{aligned}\widehat{\mathcal{D}}(X) &= \mathbb{E}\left[\frac{1}{2}\left(\boldsymbol{x}^\top\boldsymbol{x} - 2\boldsymbol{x}^\top M^\top(MM^\top)^{-1}M\boldsymbol{x} + \boldsymbol{x}^\top M^\top(MM^\top)^{-1}M\boldsymbol{x}\right)\right] \\ &= \mathbb{E}\left[\frac{1}{2}\left(\boldsymbol{x}^\top\boldsymbol{x} - \boldsymbol{x}^\top M^\top(MM^\top)^{-1}M\boldsymbol{x}\right)\right]\end{aligned}$$

Taking the derivative with respect to M, we obtain:

$$\begin{aligned}\frac{\partial\widehat{\mathcal{D}}(X)}{X} &= (MM^\top)^{-1}M\mathbb{E}\left[xx^\top\right] - 2(MM^\top)^{-1}M\mathbb{E}\left[x^\top x\right]M^\top(MM^\top)^{-1}M \\ &= (MM^\top)^{-1}M\mathbb{E}\left[xx^\top\right] - 2(MM^\top)^{-1}M\mathbb{E}\left[x^\top x\right]M^\top(MM^\top)^{-1}M \\ &= (MM^\top)^{-1}M - (MM^\top)^{-1}MM^\top(MM^\top)^{-1}M \\ &= (MM^\top)^{-1}M - (MM^\top)^{-1}M\end{aligned}$$

$$= \mathcal{Z}$$

Thus, the derivative is everywhere zero, and each M is optimal—given that its decoder is optimized. □

The meaning of this result is that, with respect to the primary encoding-decoding objective—reconstruction error, which is the principal objective that any binding method ought to satisfy—convolution-correlation is just as good and no better than any other TPR-encoding map of the right size.

In the proofs, we made several assumptions about the distributions of filler and role vectors. We now address some of these caveats.

2.3.1 Effect of normalization

The above results being in hand, we can reanalyze the role of convolution-correlation binding and its value. HRRs amount to the following procedure:

(1) a. Bind and compress with the forward map M. Input: tensor product $\boldsymbol{r} \otimes \boldsymbol{f}$. Output: vector $\boldsymbol{o} = \boldsymbol{r} \circledast \boldsymbol{f}$.

b. Unbind and decompress with the reverse map N. Input: vector $\boldsymbol{o}$. Output: an array $\boldsymbol{T}$ in the tensor product space $R \otimes F$, with the property that $\boldsymbol{T} \approx \boldsymbol{r} \otimes \boldsymbol{f}$ (for the case of a single binding) in the sense of Theorem 2.3.3.

c. Apply dot product between the retrieved TPR and the role vector (or its dual). Inputs: role vector $\boldsymbol{r}$ and tensor $\boldsymbol{T}$. Output: vector $\tilde{\boldsymbol{f}}$ with $\tilde{\boldsymbol{f}} \approx \boldsymbol{f}$.

In typical implementations of tensor product representations, it is required that the role vectors be—ideally—orthonormal. When the full TPR is at issue, this guarantees lossless

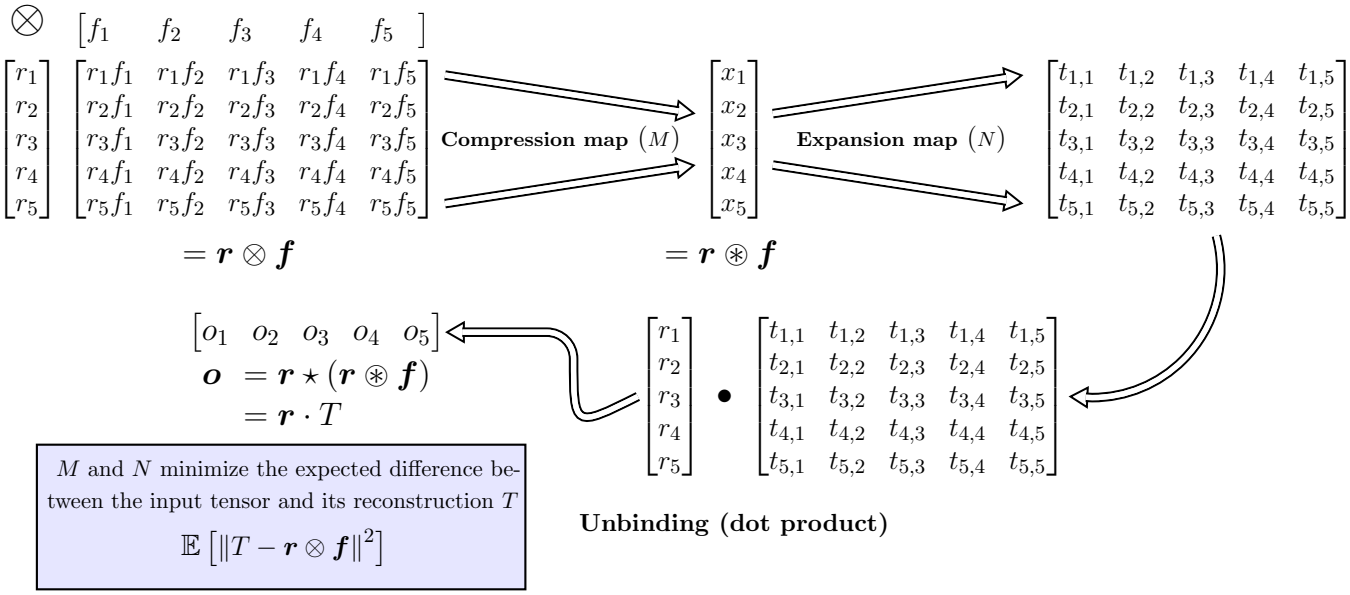


Figure 2.3: **The relation between convolution-correlation decoding and TPR binding:** Applying the convolution tensor to a TPR $\boldsymbol{r} \otimes \boldsymbol{f}$ is equivalent to the circular convolution of $\boldsymbol{r} \circledast \boldsymbol{f}$. Decoding the compressed tensor with the correlation tensor N optimally reconstructs the input tensor, after which standard TPR unbinding operations—dot products with role vectors when these are normalized, or with the duals of the role vectors—are applied. The relationships depicted here are general. They pertain not just to HRR binding, but to any suitable TPR encoding map M (i.e. one with a valid left inverse).

addressing using the original role vectors: if the tensor $\boldsymbol{T} = \sum_i \boldsymbol{r}_i \otimes \boldsymbol{f}_i$, then $\boldsymbol{r}_j \cdot \boldsymbol{T} = \boldsymbol{f}_j$. Barring this, it is often sufficient in practice for the role vectors to be normalized. This encoding scheme has the property that $\boldsymbol{r}_j \cdot \boldsymbol{r}_j \otimes \boldsymbol{f}_j = \boldsymbol{f}_j$, i.e. lossless addressing from the TPR consisting of a single FRB. Distributions for normalized vectors are more difficult to calculate because of inter-component correlations arising from the normalization process. For instance, if $d-1$ components are zero, then the final component must be 1. We now show that the distributional requirements imposed on vectors in the optimality Theorem 2.3.3 are satisfied when the role vectors are random *normalized* vectors, i.e. where each role vector $\overline{\boldsymbol{r}} = \frac{\boldsymbol{r}}{\|\boldsymbol{r}\|}$, when $\boldsymbol{r}$ is sampled from a symmetric distribution with mean zero, and has isotropic covariance that is solely a function of d.

Theorem 2.3.5. *Let* $\overline{X} = \left\{ \frac{\boldsymbol{x}}{\|\boldsymbol{x}\|} \right\}$ *be the set of vectors obtained as the normalization of each d-dimensional vector in* X *where* X *is distributed* symmetrically *with mean zero.*[5] $\overline{X}$ *is also distributed with mean zero and covariance* $\frac{1}{d}I$.

Proof. The expected variance of $\overline{x}_i$ is

$$\begin{aligned}
\mathbb{E}\left[{\overline{x}_i}^2\right] &\equiv \mathbb{E}\left[\frac{{x_i}^2}{\|\boldsymbol{x}\|^2}\right] \\
&= \mathbb{E}\left[\frac{{x_i}^2}{\sum_j {x_j}^2}\right] \\
&= \mathbb{E}\left[\frac{{x_i}^2}{{x_i}^2 + \sum_{j \neq i} {x_j}^2}\right]
\end{aligned}$$

${x_i}^2$ and $\sum_{j \neq i} {x_j}^2$ are independent chi-squared distributions with degrees of freedom 1 and $d-1$. The variance is therefore distributed as a ratio of chi-squared distributions. The

[5] Symmetry means that $p(x_i) = p(-x_i)$ for all x_i. The normal distribution $\mathcal{N}(0, cI)$ for any c is a special case.

ratio is a beta-distribution with shape parameters $\frac{1}{2}$ and $\frac{d-1}{2}$. The expected value is:

$$\mathbb{E}\left[{\overline{x}_i}^2\right] = \mathbb{E}\left[\frac{{x_i}^2}{{x_i}^2 + \sum_{j\neq i} {x_j}^2}\right] = \frac{1}{d}$$

The variance may be set to 1 by rescaling the normalized vector $\overline{x}_i$ by $\sqrt{d}$, which is simply the inverse of Plate's transformation from unit-variance to expected-norm 1 vectors.

The off-diagonal terms are zero, as we prove using the symmetry of the expectation.

$$\begin{aligned}
\mathbb{E}\left[\overline{x}_i\overline{x}_j\right] &\equiv \mathbb{E}\left[\frac{x_i x_j}{\|\boldsymbol{x}\|^2}\right] \\
&= \mathbb{E}\left[\frac{x_i x_j}{{x_i}^2 + {x_j}^2 + \sum_{k\neq i,j} {x_k}^2}\right] \\
&= \int_{-\infty}^{\infty} p(x_1, x_2 ... x_i ... x_d) \frac{x_i x_j}{{x_i}^2 + \sum_{k\neq i} {x_k}^2} d\boldsymbol{x} \\
&= \int_{-\infty}^{\infty} p(x_i) \int_{-\infty}^{\infty} p(x_1, x_2 ... x_{i-1}, x_{i+1} ... x_d) \frac{x_i x_j}{{x_i}^2 + \sum_{k\neq i} {x_k}^2} d\boldsymbol{x}_{/i} dx_i \qquad \text{(independence)}
\end{aligned}$$

where $\boldsymbol{x}_{/i}$ denotes the vector $\boldsymbol{x}$ excluding the ith component. $p(x_i)$ is symmetric (i.e. $p(x_i) = p(-x_i)$), yielding:

$$\begin{aligned}
\mathbb{E}\left[\overline{x}_i\overline{x}_j\right] &= \int_0^{\infty} p(x_i) \int_{-\infty}^{\infty} p(\boldsymbol{x}_{/i}) \frac{x_i x_j}{x_i + \sum_{k\neq i} {x_k}^2} d\boldsymbol{x}_{/i} dx_i + \\
&\qquad \int_{-\infty}^{0} p(x_i) \int_{-\infty}^{\infty} p(\boldsymbol{x}_{/i}) \frac{x_i x_j}{x_i + \sum_{k\neq i} {x_k}^2} d\boldsymbol{x}_{/i} dx_i \\
&= \int_0^{\infty} p(x_i) \int_{-\infty}^{\infty} p(\boldsymbol{x}_{/i}) \frac{x_i x_j}{x_i + \sum_{k\neq i} {x_k}^2} d\boldsymbol{x}_{/i} dx_i + \\
&\qquad \int_0^{\infty} p(-x_i) \int_{-\infty}^{\infty} p(\boldsymbol{x}_{/i}) \frac{-x_i x_j}{(-x_i)^2 + \sum_{k\neq i} {x_k}^2} d\boldsymbol{x}_{/i} dx_i \\
&= \int_0^{\infty} p(x_i) \int_{-\infty}^{\infty} p(\boldsymbol{x}_{/i}) \frac{x_i x_j}{x_i + \sum_{k\neq i} {x_k}^2} d\boldsymbol{x}_{/i} dx_i -
\end{aligned}$$

$$\int_0^\infty p(x_i) \int_{-\infty}^\infty p(\boldsymbol{x}_{/i}) \frac{x_i x_j}{x_i + \sum_{k \neq i} {x_k}^2} d\boldsymbol{x}_{/i} dx_i$$

$$= 0$$

That the expected mean of $\overline{\boldsymbol{x}}$ is zero is proven by the same symmetry considerations. □

As a corollary, $\overline{X}$ satisfies the requirements for the MPI (including the specific case of the convolution-correlation pairing) to be optimal. Therefore, using the conditions assigned to tensor product representations in self-addressing binding schemes (i.e. with *normalization* of the role vectors) in the convolution-correlation setting is tantamount to the same procedure as in (1), with the only difference being that the input role vectors are normalized, so that decoding is perfect in the single-binding case. Importantly, optimality of TPR decoding is preserved.

2.3.2 Relation to Plate's statement of the approximation

Plate provides the following account of the relation between convolution and correlation. The result of composing the convolution and correlation operations is to retrieve the convolved vector with the following componentwise error pattern:

$$[\boldsymbol{r} \star (\boldsymbol{r} \circledast \boldsymbol{f})]_i = (1 + \varphi_i) f_i + \varepsilon_i \tag{2.17}$$

where $\varphi_i = \|\boldsymbol{r}\|^2 - 1$ is the squared norm of the role vector (mean-centered with the expected norm 1), and $\varepsilon_i = \sum_{j \neq i} \sum_{k,\ell} r_k r_\ell f_j$.

Plate's requirements on the distribution of filler and role vectors mean that—due to the central limit theorem, and in the limit as $d \longrightarrow \infty$—both φ_i and ε_i can be

characterized as zero-mean noise, respectively distributed as:

$$\varphi_i \sim \mathcal{N}\left(0, \frac{2}{d}\right) \qquad\qquad \varepsilon_i \sim \mathcal{N}\left(0, \frac{d-1}{d^2}\right)$$

These facts (see [Plate, 1994] for proofs) establish the basic procedure outlined by Plate for convolution-binding: unbinding retrieves an approximation to the original input, which is then compared with the inventory of vectors (the candidate fillers) to decode using the closest match—so-called "clean-up" memories—in order to strip the noise terms.

The result established here provides a tighter connection between tensor product and convolution memories, explaining the sense in which correlation inverts the convolution: correlation is the *best linear decoder* for a tensor product compressed with convolution. In addition, it demonstrates the equivalence of convolution—in terms of decoding accuracy—to any other row-independent encoding matrix with the right constraints on the dimensions of the encoder and decoder (input dimension of d^2, output dimension of d).

2.3.3 Computational efficiency of convolution-correlation

We have shown that convolution-correlation decoding is equivalent to bilinear encoding-decoding with respect to the coarse squared-error criterion (2.16). That perspective, therefore, does not grant convolution a privileged position among bilinear binding operations. With respect to that criterion, all bilinear maps with full row rank and having the right dimensional constraints are optimal solutions. However, convolution has a distinct advantage in terms of computation-time due to the fact that it can be computed using the Discrete Fourier Transform in $\mathcal{O}(d \log d)$ time, in comparison with $\mathcal{O}(d^3)$ time for naively computing the tensor product $\boldsymbol{r} \otimes \boldsymbol{f}$ and multiplication with the d^3-element tensor M [Plate, 1994]. This advantage, however, is somewhat overstated.

HRRs are optimal only when their strict distributional requirements are met—the most important of which, in terms of time complexity, is isotropy. This holds not just theoretically, but also empirically, as we show in Section 2.3.5. If all of the filler-role bindings (vectors $\boldsymbol{x}$), or a sufficiently large sample, are available, it is always possible to transform these vectors so that unit isotropy ($\mathbb{E}\left[\boldsymbol{x}\boldsymbol{x}^\top\right] = I$) is enforced. The involved transformation requires computing (or estimating under simplifications like as role-filler independence, cf. the next section) the covariance matrix Σ, and multiplying the memory states $\boldsymbol{x}$ by the square root of its inverse, $\Sigma^{-\frac{1}{2}}$, which may be obtained from its singular value decomposition. Computing the SVD of this square matrix has complexity $\mathcal{O}(d^3)$. In applied settings where the distributions of roles, fillers, and their products must be learned (i.e. where embeddings must be learned), $\Sigma^{-\frac{1}{2}}$ must be re-estimated at each step, with the implication that each iteration (in the learning phase, at any rate) also has cubic time complexity. Thus, the computational advantages of HRRs come at the cost of accuracy if the algorithm is naively applied.

It is worth reflecting on what are meant by the distributions whose covariances must be obtained. It is not sufficient to estimate the parameters of the distribution of the role and filler vectors separately (though this is a heuristic we employ in Chapter 5 for computational reasons), since even if the constituent vectors are normalized, covariances among the components of the *tensor products* (which are the objects on which the isotropy assumptions prevail in all the preceding theorems) will arise if there are correlations between the role and filler vectors themselves. Those correlations can be estimated directly, but only at great cost. In a train-test setting, this involves first assembling TPRs for all of the training inputs, and calculating a covariance matrix that has dimension $d_r \times d_f$—all for a single step of gradient ascent. This will be a computationally intensive procedure in virtually any circumstance.

2.3.4 Non-isotropy and FRB non-independence

Strict isotropy ($\Sigma_R \propto I$ and $\Sigma_F \propto I$) and independence ($\boldsymbol{r}$ is uncorrelated with $\boldsymbol{f}$) are required in order for convolution to satisfy Theorem 2.3.3. In general, vectors entering into binding may not have this structure, particularly in empirical settings where component-wise correlations are in fact a virtue of learned representations, allowing them to capture such relations as similarity. However, the very design of typical binding schemes presupposes various independence conditions on the various numerical representations, and for good reason. These methods arose in the course of attempts to characterize numerical/neural representations that would share the key characteristics of cognitive systems like language—especially, their systematicity. From the point of view of strict grammaticality or semantic well-formedness conditions, expressions can be arranged freely, with statistically odd symbol-role associations standing equivalently to statistically congruent (i.e. with respect to world knowledge/experience) counterparts—given the right level of abstraction—as long as certain basic type constraints are satisfied. One has little trouble, for instance, understanding the sentence *The ball kicked the boy*—at least to the point of identifying the syntactic and semantic roles associated to each NP—and with respect to this fundamental capacity of role-association/retrieval, there is no difference between it and the statistically more congruent *The boy kicked the ball*.

Due to their a priori formal commitment to, coarsely put, "treating all fillers the same", cognitive theories allow exactly these sorts of free combinations of contents with structural roles, indiscriminate of the identity of those fillers. TPRs under appropriate constraints of course have this property, as a consequence of perfect decoding. As soon as TPRs are remapped into spaces smaller than the space of origin ($d_R \times d_F$), however, some information about conjunctions of features of fillers with features of roles (which is what

the elements of a TPR record) is bound to be lost. As an illustrative example, consider a model of binding of fillers to the semantic roles of Agent and Patient [Levin and Rappaport Hovav, 2005]. Reprising the example of *kick* from above, we might imagine instantiating the encoding/decoding problem as follows: there are role vectors $[0, 1]$ (Agent) and $[1, 0]$ (Patient) for the designated structural roles. In addition, there are vectors for *boy* and *ball* which we take to correspond to meaningful features of those objects:

$$\begin{array}{lccccc} \overrightarrow{boy} = & [& 1 & 0 & 0 & 1 &] \\ \overrightarrow{ball} = & [& 0 & 1 & 1 & 0 &] \\ & & \texttt{consciousness} & \texttt{stillness} & \texttt{smoothness} & \texttt{autonomous\_motion} & \end{array}$$

We aim to remap the 8-dimensional TPR vectors obtained from taking the tensor product of the fillers with the roles (in the space $R_2 \otimes F_4$) into a smaller representation, say into a 4-dimensional space (call it O_4). One plausible strategy might be to ignore role-feature conjunctions that are rare, since the value of these features will often be zero and thus can be zeroed out in recovering the TPR. Let Agent be highly correlated with `consciousness` and `autonomous_motion` (typical properties of biological entities), and let Patient be correlated with `stillness` and `smoothness` (typical properties of human-made artifacts). An appropriate remapping would be to take only those features that are highly correlated, since those are the ones that we are most likely to need to retrieve. The map from $R_2 \otimes F_4 \rightarrow O_4$ is:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \longrightarrow o_1 \qquad\qquad \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \longrightarrow o_2$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \longrightarrow o_3 \qquad\qquad \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \longrightarrow o_4$$

where $\boldsymbol{o} = [o_1, o_2, o_3, o_4]$ is the output vector, which is obtained by taking the dot product of the input TPR "matrix" with each element of the "matrix" on the left. The first element of $\boldsymbol{o}$ records the role-bound feature Agent-and-conscious, the second Patient-and-still, etc. Inverting this map leads to perfect recovery when *boy* is bound to *Agent* and *ball* to *Patient*, but extracts only the zero vector when those roles are reversed.

Such a scheme would perhaps be optimal from the point of view of recovering the identities of the two participants in the event when the notion of optimality depends on real-world statistics, e.g. when the role of Agent is highly correlated with consciousness and evincing autonomous motion and that of Patient with stillness and being smooth.[6] It, however, lacks the core feature of symbolic cognitive systems: their compositional productivity, i.e. the freedom to combine the full inventories of symbols with roles.

It takes some reinterpretation to align these classical concepts with a statistical characterization, and there is no perfect fit. What is intended is something like this: all features should be treated the same. There should be no systematic difference in the decodeability of individual characteristics of the representation expressed numerically in terms of features. Each feature, on aggregate, should be equally decodeable. Thus, the

[6]We leave aside for present purposes another possible scheme where the maps in question are *affine* rather than strictly linear, representing a bias term for the highly correlated features (e.g. Agent-and-conscious and Patient-and-still), assuming their values to be 1 (since this is true on average), leaving only the extremely "rare" features to be extracted. In that case as well, though, featural homogeneity is not satisfied, since the decoding error would be perfect for the rare features selected for decoding, but not for the common features whose values are set by the bias.

expected decoding error should be the same for all features ("*featural homogeneity*"):

$$\mathbb{E}\left[\left([NM\boldsymbol{x}]_k - x_k\right)^2\right] = \mathbb{E}\left[\left([NM\boldsymbol{x}]_\ell - x_\ell\right)^2\right] \qquad \forall k, \ell \qquad (2.18)$$

This is not a generic property of linear maps, including those with full row rank. As a very simple example, take the linear map from $M : \mathbb{R}^{25} \longrightarrow \mathbb{R}^5$ that takes the first 5 components of the input and copies them to $\mathbb{R}^5$, with zeros elsewhere, i.e.

$$M_{ij} = \begin{cases} = 1 & \text{if } i \leq 5 \text{ and } i = j \\ = 0 & \text{otherwise.} \end{cases}$$

This has a well-defined pseudo-inverse, which is just $M^\top$. However, it is easily verified that Eqn. (2.18) is not satisfied in this case, since the expected retrieval error is zero for the first five components, and for the remaining 20, is the variance of the corresponding component of $\boldsymbol{x}$.

Theorem 2.3.6. *Let M be an encoding matrix with full row rank, N its pseudoinverse, and USV any singular value decomposition of M. Further, let the vectors $\boldsymbol{x}$ be distributed with covariance $\mathbb{E}\left[\boldsymbol{x}\boldsymbol{x}^\top\right] = I$. Then M and N exhibit* featural homogeneity *(2.18).*

Proof. The expected componentwise error $\mathcal{D}_i(\boldsymbol{x})$ with respect to encoding matrix $M \in \mathbb{R}^{m \times n}$ is

$$\mathbb{E}\left[\left([NM\boldsymbol{x}]_i - x_i\right)^2\right] = \mathbb{E}\left[\boldsymbol{n}_{i\cdot}{}^\top M\boldsymbol{x}\boldsymbol{x}^\top M^\top \boldsymbol{n}_{i\cdot} - 2\boldsymbol{n}_{i\cdot}{}^\top M\boldsymbol{x}x_i + {x_i}^2\right]$$

where $\boldsymbol{n}_{i\cdot}$ is the i^{th} row of N. M is assumed to have full row rank. We begin by expressing

M in terms of its singular value decomposition:

$$M = USV^\top$$

where U and V are the matrices of left and right singular vectors of M, and S the square matrix with singular values along the diagonal. Its pseudoinverse N is

$$N = VS^{-1}U^\top$$

where S^{-1} has the reciprocals of M's singular values along the diagonal. Rewrite $\boldsymbol{n}_{i\cdot}$ as $US^{-1}\boldsymbol{v}_{i\cdot}$, with $\boldsymbol{v}_{i\cdot}$ the i^{th} row of V. Thus,

$$\begin{aligned}
\mathbb{E}\left[([NM\boldsymbol{x}]_i - x_i)^2\right] &= \boldsymbol{n}_{i\cdot}^\top M\mathbb{E}\left[\boldsymbol{x}\boldsymbol{x}^\top\right] M^\top \boldsymbol{n}_{i\cdot} - 2\boldsymbol{n}_{i\cdot}{}^\top M\mathbb{E}\left[\boldsymbol{x}x_i\right] + \mathbb{E}\left[{x_i}^2\right] \\
&= \boldsymbol{v}_{i\cdot}{}^\top S^{-1}U^\top USV^\top \mathbb{E}\left[\boldsymbol{x}\boldsymbol{x}^\top\right] VSU^\top US^{-1}\boldsymbol{v}_{i\cdot} \\
&\quad -2\boldsymbol{v}_{i\cdot}{}^\top S^{-1}U^\top USV^\top \mathbb{E}\left[\boldsymbol{x}x_i\right] + \mathbb{E}\left[{x_i}^2\right]
\end{aligned}$$

Setting $\mathbb{E}\left[\boldsymbol{x}\boldsymbol{x}^\top\right] = I$ (which can always be achieved by transformation) implies that $\mathbb{E}\left[\boldsymbol{x}x_i\right] = \mathbf{1}_i$, the vector with 1 at component i and zeros elsewhere, such that $V^\top\mathbb{E}\left[\boldsymbol{x}x_i\right] = V^\top\mathbf{1}_i = \boldsymbol{v}_{i\cdot}$, the i^{th} row of V. Due to the properties of the SVD, $U^\top U = I = V^\top V$. Therefore,

$$\begin{aligned}
\mathbb{E}\left[([NM\boldsymbol{x}]_i - x_i)^2\right] &= \boldsymbol{v}_{i\cdot}{}^\top\boldsymbol{v}_{i\cdot} - 2\boldsymbol{v}_{i\cdot}{}^\top\boldsymbol{v}_{i\cdot} + 1 \\
&= 1 - \|\boldsymbol{v}_{i\cdot}\|^2
\end{aligned}$$

Supposing that that the expected componentwise decoding error $\mathbb{E}\left[\mathcal{D}_i(\boldsymbol{x})\right] = \mathbb{E}\left[\mathcal{D}_j(\boldsymbol{x})\right]$ is identical for each i, j, it follows that each row of V has the same norm. To get the other

direction, follow the equal signs in reverse. □

Corollary 2.3.6.1. *When its assumptions are satisfied, convolution-correlation decoding has featural homogeneity under criterion (2.18).*

Proof. The rows of M are orthonormal—a corollary of the fact that $MM^\top = I$ (see Theorem 2.3.2). This being so, we have $M = IIM$ with I having orthonormal columns and likewise M for the rows. Thus, IIM is a singular value decomposition of M, with $M = V^\top$ in the standard formula. We now show that the columns of M (rows of V) all have the same norm. The columns of M are indexed by ij (Eqn. 2.14). The squared norm of each column is $\sum_k \left([M]_{kij}\right)^2$. For a given i, j, there is one and only one k such that $j = (k - i) \bmod d$ (footnote 2), and the value of that entry is $\frac{1}{\sqrt{d}}$. The norm, $\frac{1}{d}$, is the same for each column of M. □

2.3.5 The empirical performance of binding methods

The theoretical results provide an account of (1) the way in which correlation decodes convolution, including the sense in which that pairing is optimal, and (2) the reasons why convolution-correlation decoding—beyond the utility derived from the fast DTFT procedure for computing it—may have a preferred place among bilinear binding operations when at issue is the modeling of cognitive operations that, at a certain level of abstraction, do not seem to depend on a representation's featural content. However, we depended on idealizations like isotropy and independence, idealizations that do not tend to hold in empirical settings. We now ask: what effect does departure from the assumptions have on decoding accuracy in well-controlled simulations in which we vary the distributions of vectors that enter into binding, and inspect the performance of different bilinear binding methods in decoding the results.

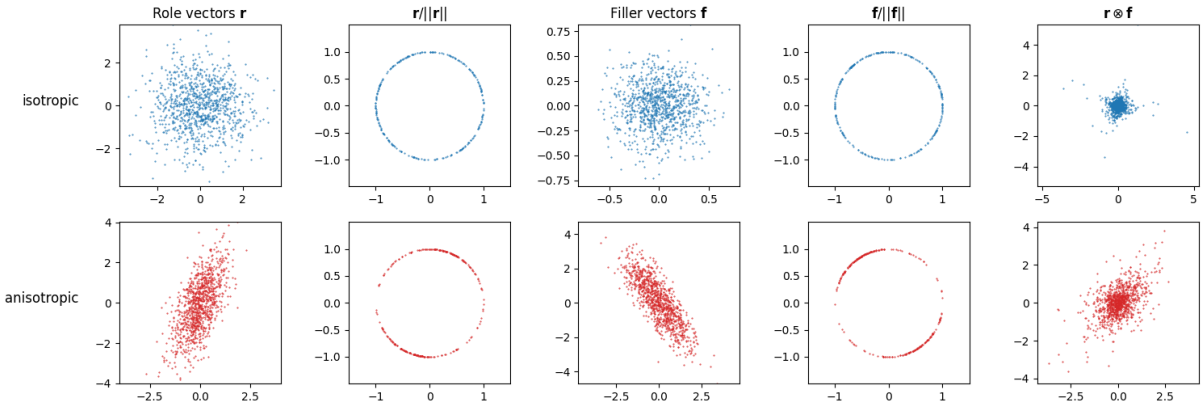


Figure 2.4: Samples from a distribution of 25d role vectors, 25d filler vectors, and $25^2 = 625$d tensor product filler-role bindings, projected onto a plane. The second column from the left depicts a distribution of *normalized* role vectors sampled from the distributions on the left—normalized in PC space for the purposes of visualization (in actuality these are projected onto the 25d hypersphere), and similarly for the filler vectors (second plot from the right). The first sample (blue) is drawn from an isotropic distribution. The second (red), anisotropic.

Even from a purely empirico-theoretical perspective, this analysis is important to do for a number of other reasons. The most obvious is that the results enumerated above refer specifically to the most convenient criterion available for the analysis of linear models: the sum of squared componentwise error. In the deployment of these models, however, the situation is more complicated. Accuracy in applications typically depends on additional operations applied to the retrieved vectors, specifically those itemized in Section 2.3.1. The usual computation stream—using the bilinear map notation developed in this chapter, which encompasses convolution-correlation as well as a broader landscape of models where dimensions are allowed to vary—is the following:

(2) **Filler-role binding computation stream**: To retrieve filler vector $\boldsymbol{f}_i \in \mathcal{F}$

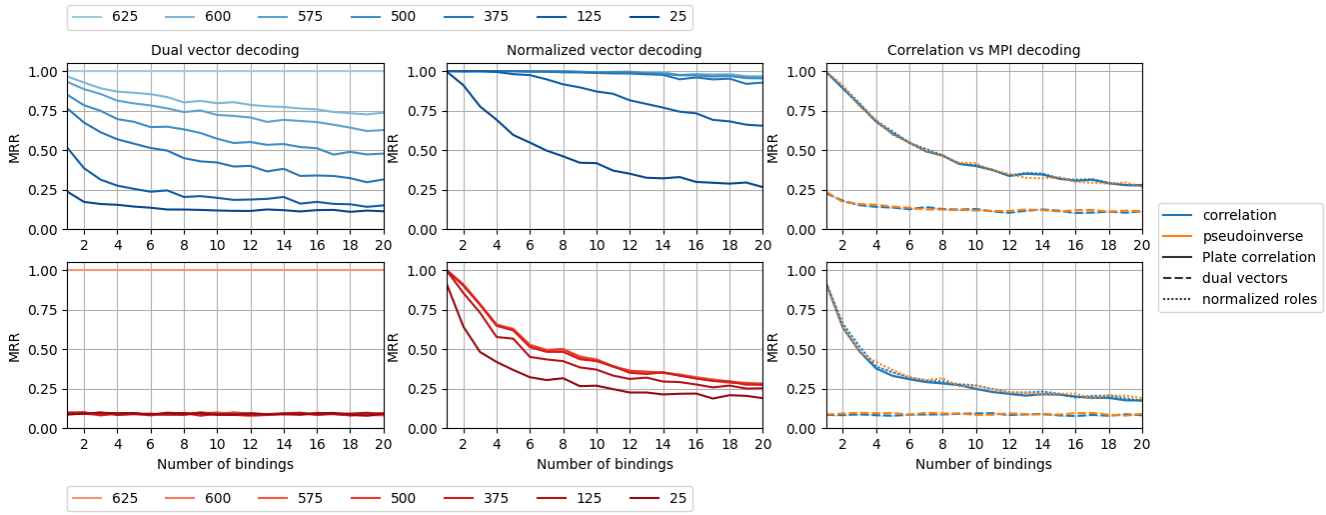


Figure 2.5: Decoding accuracy for filler-role bindings sampled from two distributions (Figure 5.7). The color-coding legends for the left-two columns are at the top (blues/isotropic) and bottom (reds/anisotropic), and indicate the dimensionality of the output vectors. Comparisons between correlation and MPI decoding—the rightmost column—are labeled as in the legend on the very right. Details: For 15 iterations, we sampled encoding matrices M and solved for their decoding matrices N using the Moore-Penrose inverse. The matrices were varied for the output dimensions (i.e. the amount of compression), with a minimum of 25d (i.e. the output dimension of the convolution). Varying these, we assessed the decoding accuracy as a function of the number of bindings superposed in memory. The rightmost graphs depict the results for just the models with 25d output dimensions—both those with convolution-correlation, and with sampled encoding/decoding matrices—using the color/style scheme indicated in the rightmost box. The colors/styles are shared between the top (isotropic) and bottom (anisotrophic) graphs.

bound to a given role r_i, do:

$$\text{argmin}_j \;\; \text{dist}\left(\boldsymbol{f}_j, \boldsymbol{r}_i \cdot \text{unvec}\left(NM \;\text{vec}\left(\sum_i \boldsymbol{r}_i \otimes \boldsymbol{f}_i\right)\right)\right)$$

where vec is the vectorization function for multi-indexed tensors, unvec the inverse of that operation, and dist some distance function, (e.g. Euclidean, negative cosine or Pearson similarity).

Using these methods, then, involves role-extraction of fillers via a dot product post-compression, comparison with an external inventory of vectors, as well as employment of a distance function. Given the variability introduced by these additional parameters, they are best investigated empirically.

In Figure 5.7, we show the results of running an array of binding-unbinding algorithms on vectors generated from a number of different distributions. There are two main categories: the isotropic distribution (red) was sampled from two zero-mean Gaussian (one for fillers and for roles) with covariance matrix I. The resulting distribution of tensor products $\boldsymbol{r} \otimes \boldsymbol{f}$, which is isotropic but non-Gaussian, is also displayed. The anisotropic distribution for $\boldsymbol{r}$ and $\boldsymbol{f}$ were generated from randomly sampled covariance matrices $\Sigma_{\boldsymbol{r}}$ and $\Sigma_{\boldsymbol{f}}$, each divided by the trace of the matrix in order to match the expected norm of the isotropic distribution. The resulting distribution of TPRs is, as well, anisotropic. Additionally, for each of these sampling sources, we generated a distribution from projecting the role vectors onto the unit hypersphere (vector normalization). These vectors were entered into an iterative sampling process for memory vectors (sums of TPRs) where 1 to 20 25-d role vectors were bound to any of 50 25-d filler vectors. The inventory of roles and fillers was resampled 15 times. Starting with an inventory of 25 role and 50 filler

vectors sampled from the (an)isotropic distributions, there are four basic models:

1. **Dual vector decoding**: Assemble $\mathtt{M}$ as $\mathtt{M} = \sum_i \boldsymbol{r}_i \otimes \boldsymbol{f}_i$. Encode-then-decode the TPR as $\tilde{\mathtt{M}} = \text{unvec}\left(NM \text{ vec}\left(\mathtt{M}\right)\right)$ (M the encoding matrix and N its pseudoinverse), and then dot the result with the dual vector $\boldsymbol{r}_i{}^*$. Observe that, in this model, decoding is exact for the uncompressed TPR, i.e. $\boldsymbol{r}_i{}^* \cdot \mathtt{M} = \boldsymbol{f}_i$.

2. **Normalized vector decoding**: $\mathtt{M} = \sum_i \frac{\boldsymbol{r}_i}{\|\boldsymbol{r}_i\|} \otimes \boldsymbol{f}_i$, with decoding as $\tilde{\boldsymbol{f}}_i = \frac{\boldsymbol{r}_i}{\|\boldsymbol{r}_i\|} \cdot \tilde{\mathtt{M}}$. In this case, decoding from the original memory is inexact due to overlap between the role vectors.

3. **Plate decoding**: Role and filler vectors are rescaled by $\frac{1}{d}$ to ensure the Plate conditions (componentwise variance of d) are met in the isotropic case. The same transformation is applied for the vectors sampled from the anisotropic distribution, which have an expected norm of 1 due to trace-normalization, but with varying componentwise variances. The compressed memory is assembled as $\tilde{\mathtt{M}} = \sum_i \frac{\boldsymbol{r}_i}{\sqrt{d}} \circledast \frac{\boldsymbol{f}_i}{\sqrt{d}}$, with decoding using the correlation and the rescaled role vector, the result again rescaled to invert the Plate transformation: $\tilde{\mathtt{f}}_i = \sqrt{d}\left(\frac{\boldsymbol{r}_i}{\sqrt{d}} \star \tilde{\mathtt{M}}\right)$.

4. **Normalized correlation decoding**: Here, role vectors were first normalized, which yields a distribution that satisfies the Plate conditions (componentwise variance of $\sqrt{d}$) in the isotropic case. Filler vectors are simply rescaled. The memory is $\mathtt{M} = \sum_i \frac{\boldsymbol{r}_i}{\|\boldsymbol{r}_i\|} \circledast \frac{\boldsymbol{f}_i}{\sqrt{d}}$, with decoding as $\tilde{f}_i = \frac{\boldsymbol{r}_i}{\|\boldsymbol{r}_i\|} \star \tilde{\mathtt{M}}$.

Whereas in (1) and (2) we vary the output dimension of the encoding matrices, (3) and (4) are restricted to just those with 25-d output vectors. To rank the encoding-decoding outputs with respect to the inventory of fillers, we used the negative cosine similarity as a distance measure.

Results. Figure 2.5 shows the results for all models. We first examine the isotropic distribution (top row, Blue). As expected, the models with greater output-dimension performed better than those with tighter resource constraints, with a gradual cline as the number of stored bindings increases. Across the board, normalized vector decoding performed better than dual vector decoding, and with a more gradual decline in performance in all cases. Most starkly, performance with 25-d output vectors in the normalized setting is near perfect (.997) with a single binding, whereas it is only .231 in dual-vector decoding. Observe that, since in each iteration we sampled 25 role vectors from a 25-d space, the role vectors tended to have low overlap—with a mean absolute dot product of .16 (determined empirically)—which accounts for the empirical sufficiency of normalized vector decoding in memories with multiple superposed bindings.

The rightmost panel shows the results of comparing all 25-d output vectors, separating out the randomly generated encoding maps from convolution-correlation decoding. We find that correlation-decoding performs equivalently to the sampled encoding maps. This is expected given Theorem 2.3.4, though there we used a different criterion for evaluating model error. The empirical result extends this finding to a more realistic setting in which vectors are bound, retrieved, and then distance-matched to a given inventory.

The results for the anisotropic distribution are displayed in the three bottom graphs. Dual vector decoding is perfect in the case of 625-d vectors (the full TPR), but at the chance level when memories were remapped into a space smaller than the original TPR.[7] This is probably due to the near-singularity of the role-vector matrix—from which we compute the dual vectors via the pseudoinverse—when its rows (the role vectors) are highly correlated.[8] The situation is somewhat better for self-addressing vectors. Re-

[7]The chance level is MRR=.0899, which was verified empirically by sampling ranks uniformly from 50 (for the 50 filler options).

[8]Inspecting the dual vectors derived from this procedure showed duals with components whose values

trieval with normalized roles provides reasonable decoding accuracy for a small number of bindings, but drops to the range of .19 to .28 (across all memory embedding sizes) as the number of stored bindings increases (compared to the range .27 to .97 for the same models, 20 bindings, under isotropy). The severe degradation of performance in this case shows the importance of satisfying the isotropy conditions in order for MPI decoding (including correlation decoding) to be accurate.

Observe that decoding error in anisotropic distributions comes from a number of sources. The first is the non-optimality of the MPI, leading to error in retrieval of the tensor product in the decompression phase. Second, error arises in unbinding due to role vectors being tightly packed together with other role vectors, meaning that unbinding retrieves from neighboring roles when those roles are summed together in memory. Similarly, error is introduced from the fact that some filler vectors are extremely close together due to correlations. Combined, these factors drastically affect MPI decoding.

This section provides an empirical evaluation of correlation/correlation and other methods of bilinear vector-binding in controlled simulations. In Chapter 5, using an applied model developed therein, we will additionally ask: (2) how do the different binding methods behave in empirical settings where the assumptions—especially, role-filler independence—are not satisfied? This is important for a number of reasons. The first is that, while the above simulations vary the distributions of input vectors, which has consequences for the distribution of tensor products entering into the bilinear binding operations, it is noteworthy that correlations between the components of the constituent vectors (the fillers and the roles) are not the only source of correlations between components of the input tensors. As mentioned in 2.3.3, another source is correlations between

were in the hundreds or thousands, meaning that small numerical differences in the retrieved TPRs led to large differences in their dot products with the duals.

the choices of fillers and roles. It is not sufficient that the *sentient* feature be uncorrelated with the *animate* feature on the filler vectors—correlations which can always be corrected for using the coordinate transformation specified in Section 2.3.3 above. Reprising an example from earlier, the filler *boy* should be uncorrelated with the role *Agent*, and the filler *ball* uncorrelated with the role *Patient*. It should be clear that these conditions do not typically hold and, that where they do not, the transformations required to enforce them come at great computational cost.

2.4 Summary

In this chapter, we have reviewed the two primary methods proposed for implementing analogues of symbolic structure-assembly in neural network models. In restricting ourselves to these classical models, we have not treated the wealth of models that rely on significant nonlinearities. The upshot is that these models are amenable to theoretical analysis. We have provided such an analysis in terms of a broader framework of Bilinear Binding, which encompasses both the tensor product and convolution-correlation decoding, as well as a number of implementations of Tensor Product Representations that have coped with the dimensional requirements of TPRs by reducing these to tractable size with learned linear maps. We have shown that, with respect to certain theoretical criteria, these methods are equivalent, and that convolution-correlation is on equal footing with arbitrary bilinear maps, both in terms of accuracy and in terms of computing resources required. Empirical results suggest that the flexibility in parameter-learning available to bilinear models tends to improve performance relative to the fixed inductive biases implied in the convolution-correlation setup.

Chapter 3

Characteristics of self-optimizing networks

There is good evidence for the existence of self-optimizing networks as the neural basis of cognitive computation in certain domains. Niessing and Friedrich [Niessing and Friedrich, 2010] examined the neural classification of olfactory stimuli in the olfactory bulb of explanted zebrafish brains. Imaging activation in response to stimuli with compounds gradually varied in their concentration of several amino acid odors, they extracted temporally-varying neural patterns on a set of mitral cells with quarter-second temporal resolution. They found that neural patterns were initially highly correlated across all inputs, and then gradually decorrelated and binned into distinct classes corresponding to the dominant scent, arriving at steady-state patterns for each category at around 1500 milliseconds. Small initial pattern differences were gradually magnified, with clearly-discriminable patterns emerging in the steady state, leading to categorical encoding of continuously-varying inputs—a result that replicates similar experiments in rats [Barnes et al., 2008] showing categorical grouping of neural patterns with linearly-varying input

characteristics. The results also accord well with the categorical structure of scent perception in humans [Castro et al.,], with clear analogues in perceptual studies showing categorical perception in cognitive domains such as color [Hanley and Roberson, 2011] and speech-sound perception [Altmann et al., 2014].

The authors remark that the results are consistent with point-attractor models of neural computation, in which neural circuits are configured such that a neural population's temporal dynamics lead to patterns gravitating to designated points in the activation space (see Fig. 3.3 for a qualitative comparison of modeling results with the results of Nissing and Friedrich). At a conceptual level, network architectures of this kind can be characterized as performing "pattern-completion"—taking input patterns that correspond imperfectly with stored patterns representing discrete stimulus classes, and gradually warping the inputs into the target patterns.

Thus, the available neuroscientific evidence recommends the pursuit of network architectures based on attractors—networks that compute representations across time with various steady states corresponding to discrete stimulus classes. In other words, categorization of sensory inputs does not just consist of a discriminative procedure (feedforward), but rather of (feedback-mediated) pattern modification that is well characterized as modulation of input patterns into patterns corresponding to the prototypes for each class.

We start from this intuition that computation in neural networks can often be modeled as a time-dependent process in which a network's internal representations of a stimulus are operated on by a network's internal dynamics. In many cases, it is convenient to begin by defining a measure—henceforth, the *Harmony*—that assigns a numerical value to every possible state of the network, and which can also be used to directly define self-optimizing dynamics that have the right general properties. An appropriate Harmony

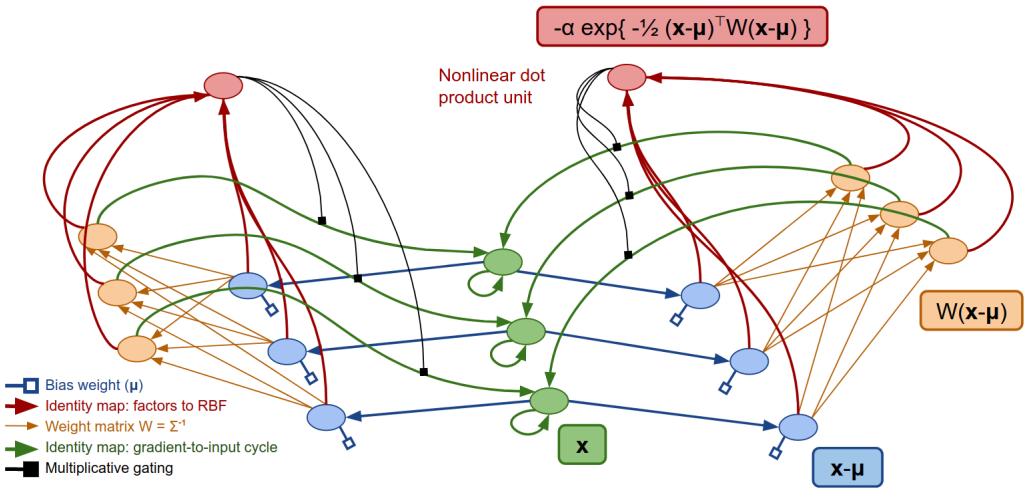


Figure 3.1: Schematic for a two-class radial pattern-completion network, a composition of two RBF circuits. Green nodes hold the value of the main state variable $\boldsymbol{x}$, with blue nodes and orange nodes implementing the transformations $\boldsymbol{x} - \boldsymbol{\mu}$ and $W(\boldsymbol{x} - \boldsymbol{\mu}_i)$, respectively. The weight matrix W is set to ${\Sigma_i}^{-1}$. The red node computes the main nonlinearity: the dot product of the blue nodes and orange nodes (Mahalanobis distance between $\boldsymbol{x}$ and $\boldsymbol{\mu}$), passing through the exponential function. This computes $\psi(\boldsymbol{x}; \boldsymbol{\mu}_i, \Sigma_i)$, a scalar that gates the green weights. The input to the state variable $\boldsymbol{x}$ is the desired gradient.

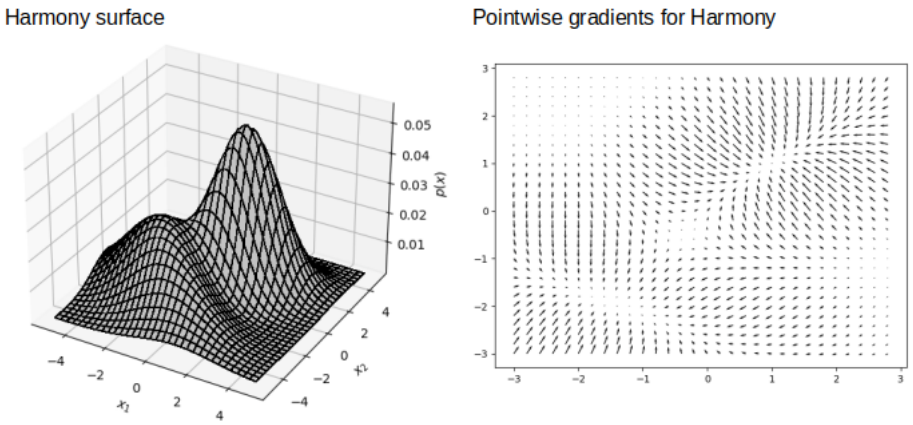


Figure 3.2: Left: Harmony surface for a two-distribution Radial Basis network over two state space components x_1 and x_2, where α and β are set to the normalization constants for the corresponding distribution, multiplied by $\frac{1}{2}$ (uniform priors). Each distribution has a distinct mean μ_i and covariance Σ_i, yielding two local optima, one at each mean. Right: pointwise gradients for the state variable $\boldsymbol{x} = [x_1, x_2]$, with stable equilibria at the two means.

function has the property that its stable maxima are associated with all and only the target patterns—e.g. the prototype patterns for the target classes. We show by example how an appropriate choice of Harmony function allows one to derive networks with the right dynamics.

3.1 Example: Radial Basis Networks

Let's examine the simple case of a multi-class classification problem. We first define a neural network architecture—organized along the lines of the principles of self-optimization—that performs a simple multi-class classification problem in a manner that accords with the qualitative facts observed by Niessing and Friedrich. We set the Harmony to a weighted sum of radial basis outputs. Sticking with the two-class example for simplicity, this is:

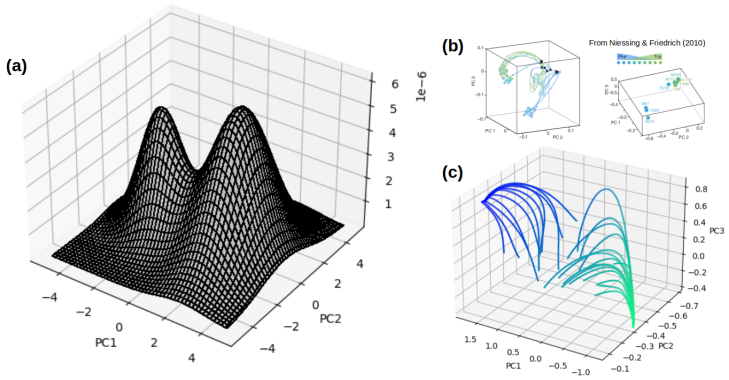


Figure 3.3: Optimization dynamics of an RBF with a 10-dimensional state space and the Harmony function (a) projected onto the first two principal components. (b) Visualization of zebrafish mitral cell patterns in response to chemical stimulation of the olfactory bulb, with the concentration of the chemical stimulus varied gradually. Stimuli were composed of a pair of chemicals (Tryptophan/Trp and Phenylalanine/Phe) in varying ratios ([100% Trp, 0% Phe]; [90% Trp, 10% Phe], etc.). The figure illustrates steady states (right) and temporal trajectories (left) of neural responses to stimuli thus varied. (c) Temporal trajectories of points—projected onto the first 3 PCs—sampled from the pair of distributions that generated (a) according to RBF dynamics (Direction of time is from the interior to the boundaries of the state-space cube).

$$(1) \qquad \mathcal{H}_{\text{BRF}}(\boldsymbol{x}) = \alpha \cdot \exp\left\{-\tfrac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_1)^\top {\Sigma_1}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1)\right\} + \beta \cdot \exp\left\{-\tfrac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_2)^\top {\Sigma_2}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_2)\right\}$$
$$\equiv \alpha \cdot \psi\left(\boldsymbol{x}; \boldsymbol{\mu}_1, {\Sigma_1}^{-1}\right) + \beta \cdot \psi\left(\boldsymbol{x}; \boldsymbol{\mu}_2, {\Sigma_2}^{-1}\right)$$

where α and β reflect priors on each distribution, and may be set to the prior-weighted normalization constants for the multivariate normal distribution—$p_i(2\pi)^{-\frac{d}{2}}|\Sigma_i|^{-\frac{1}{2}}$ with $\sum_i p_i = 1$, in which case $H(\boldsymbol{x})$ is exactly the marginal probability of $\boldsymbol{x}$ under a mixture of Gaussians with means $\boldsymbol{\mu}_i$, precision matrices ${\Sigma_i}^{-1}$, and priors α, β—or arbitrarily. Local maximization of the value of $H_{\text{RBF}}(\boldsymbol{x})$ with respect to $\boldsymbol{x}$ can be viewed as assigning the input to one of the i available classes, each with a corresponding "prototype" $\boldsymbol{\mu}_i$. The discriminative procedure is straightforward: compute output values for the terms corresponding to each class, and pick the class that is most likely to have generated the input. However, the neuroscientific evidence enumerated above suggests that, in broad

strokes, the computational procedure that is actually implemented amounts to having a network set up such that running the network forward performs class assignment by morphing the input pattern into an output pattern that is prototypical for one of the classes. Our premise is that the morphing function can be characterized by gradient ascent on a network Harmony function. For H_{RBF}, the relevant network is given by setting the time-derivative of the activation values for $\boldsymbol{x}$ to the derivative of H_{RBF} with respect to $\boldsymbol{x}$:

$$(2) \qquad \frac{d\boldsymbol{x}}{dt} = \frac{\partial \mathcal{H}_{\mathrm{RBF}}}{\partial \boldsymbol{x}} = -\alpha \cdot \psi\left(\boldsymbol{x}; \boldsymbol{\mu}_1, {\Sigma_1}^{-1}\right) \cdot {\Sigma_1}^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_1) - \beta \cdot \psi\left(\boldsymbol{x}; \boldsymbol{\mu}_2, {\Sigma_2}^{-1}\right) \cdot {\Sigma_2}^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_2)$$

In words, the velocity of $\boldsymbol{x}$ is the sum of two affine transformations of $\boldsymbol{x}$, weighted by the current values of the corresponding scalar $\psi\left(\boldsymbol{x}; \boldsymbol{\mu}_i, \Sigma_i\right)$.

This function may be implemented using one of the *RBF circuits* depicted on each side of Fig. 3.1, which illustrates a network whose spreading activation yields the desired dynamics. The activation of each unit n obeys the rule

$$(3) \qquad \frac{dx_n}{dt} = f(\iota_n(t)) - x_n(t)$$

where $f(\iota_u(t))$ denotes a function of the input to u at time t, and the second term denotes a decay in proportion to the current activation value of the unit—requiring that the unit have a constant external input to maintain its current level of activation. For all but one unit in each circuit, f is the linear identity $f(\iota_u) = \iota_u$, returning the input. First, linear input units (green nodes). The goal is to compute the value of the main state variable $\boldsymbol{x}$ as it evolves across time. We give $\boldsymbol{x}$ self-wires (an input maintaining the current level of

activity) and add a derivative yielded by the computation stream, i.e.:

$$\begin{aligned}
\iota_{\boldsymbol{x}}(t) &= \boldsymbol{x}(t) + \frac{\partial \mathcal{H}_{\text{RBF}}}{\partial \boldsymbol{x}(t)} \\
\frac{d\boldsymbol{x}}{dt} &= \frac{\partial \mathcal{H}_{\text{RBF}}}{\partial \boldsymbol{x}(t)} && \text{sum of } \iota_{\boldsymbol{x}} \text{ and rule (3)} \\
\boldsymbol{x}(T) &= \boldsymbol{x}(0) + \int_0^T \frac{\partial \mathcal{H}_{\text{RBF}}}{\partial \boldsymbol{x}(t)}\, dt
\end{aligned}$$

In the second layer, the value of linear input units—shared across distinct circuits representing each maximum—is transmitted via the identity map (blue wires) to a set of units implementing mean-subtraction, distinct for each circuit, with biases $-\mu_i$ for each unit u_i. For units $\boldsymbol{u}$ that are linear functions of $\boldsymbol{x}$ with weight matrix W, this is:

$$\begin{aligned}
\frac{d\boldsymbol{u}}{dt} &= \frac{d}{dt}[W\boldsymbol{x}(t)] + \iota_{\boldsymbol{u}}(t) - \boldsymbol{u}(t) \\
&= W\frac{\partial \mathcal{H}}{\partial \boldsymbol{x}(t)} + \iota_{\boldsymbol{u}}(t) - \boldsymbol{u}(t)
\end{aligned}$$

The layer 2 units are initialized to $\boldsymbol{x}(t) - \boldsymbol{\mu}$, with a bias of $-\boldsymbol{\mu}$, yielding $\frac{d\boldsymbol{u}}{dt} = \frac{\partial H}{\partial \boldsymbol{x}(t)}$, and similarly for layer 3, which has weight matrix Σ^{-1} (orange wires)—the precision matrix for the corresponding distributions $\left(\frac{d\boldsymbol{u}}{dt} = \Sigma^{-1}\frac{\partial \mathcal{H}}{\partial \boldsymbol{x}(t)}; \therefore \boldsymbol{u}(T) = \Sigma^{-1}(\boldsymbol{x}(0) - \boldsymbol{\mu}) + \int_0^T \Sigma^{-1}\frac{\partial \mathcal{H}}{\partial x(t)}dt = \Sigma^{-1}\boldsymbol{x}(t)\right.$. A single (red) unit implements the key nonlinearity, an exponential of the dot product of the layer 2 and layer 3 inputs, which multiplicatively gates connections (green wires) transmitting the gradient $\frac{\partial H}{\partial \boldsymbol{x}(t)} \propto \Sigma^{-1}(\boldsymbol{x}(t) - \boldsymbol{\mu})$, with the multiplicative gate yielding an input to $\boldsymbol{x}$ of $\psi\left(\boldsymbol{x}; \boldsymbol{\mu}, \Sigma^{-1}\right) \cdot \Sigma^{-1}(\boldsymbol{x}(t) - \boldsymbol{\mu})$, as desired.

Via the green wires, the current value of the orange nodes passes to the input nodes via an identity map, multiplicatively gated by the output of the nonlinear unit, such that the instantaneous input to the green nodes (the input) reproduces the desired gradient (2)

for one of the two distributions: $\alpha\,\psi(\boldsymbol{x};\boldsymbol{\mu}_u, W_i)\cdot W(\boldsymbol{x}-\boldsymbol{\mu}_i)$. The other side of the network completes the computation by summing in the gradient with respect to the parameters for the second distribution $(\beta, \boldsymbol{\mu}_2, {\Sigma_2}^{-1})$.

With appropriate settings of the parameters, the resulting network has the property that (1) the network Harmony is the probability density of the current pattern $\boldsymbol{x}$ being generated by either of the distributions, and (2) the network dynamics perform local optimization of the network state to yield a pattern prototypical for one of the two distributions. Composing multiple circuits of this type allows us to derive pattern-mapping networks solving classification problems with arbitrary numbers of classes.

A network with this structure models the qualitative dynamics observed in brain studies of pattern categorization—namely, approaching an attractor that sharply categorizes inputs by clustering them as a function of the original input. Figure 3.3 illustrates the resulting dynamics from a discrete-time simulation with two distributions and difference equation

$$\begin{aligned}\boldsymbol{x}(t+1) = \boldsymbol{x}(t) &- \tau\cdot\alpha\cdot\psi\left(\boldsymbol{x}(t);\boldsymbol{\mu}_1,{\Sigma_1}^{-1}\right)\cdot{\Sigma_1}^{-1}(\boldsymbol{x}(t)-\boldsymbol{\mu}_1)\\ &- \tau\cdot\beta\cdot\psi\left(\boldsymbol{x}(t);\boldsymbol{\mu}_2,{\Sigma_2}^{-1}\right)\cdot{\Sigma_2}^{-1}(\boldsymbol{x}(t)-\boldsymbol{\mu}_2)\end{aligned}$$

for some τ, which is the sum of derivatives for the two distributions with respect to $\boldsymbol{x}(t)$.

This illustration links optimization processes to the likely biological solutions to certain computational problems—in this case, the assignment of sensory input to classes grouping like stimuli. In the characterization developed here, those inputs are taken to stand in for neural patterns. However, it is also possible to think in terms of conceptual attributes of objects or other stimuli that vary essentially smoothly and continuously in their values, and yet exhibit fairly sharp category boundaries in most cases. Human

can readily produce judgments of the prototypicality of stimuli with respect to particular categories, and that prototypicality correlates with other observables like reaction time in classification tasks, the rate with which children acquire categories with/without natural prototypes, and the probability that subjects will produce particular instances of a category—e.g. "dog" versus "dolphin" as instances of "mammal" [Rosch, 1978].

Self-optimizing networks of the indicated sort—where points in a continuous space of attributes are assigned to local maxima—can explain these broad qualitative characteristics in fairly straightforward ways with appropriate linking hypotheses—linking, for instance, processing time in classification tasks to the time that is required to map a point to its corresponding prototype, with computation time defined by optimization dynamics quite literally; or the relative sharpness of category membership yielded by the presence of stable equilibria only at local maxima. Classic findings about the categorical perception of speech sounds—showing, for instance, that stimuli with continuously-varying physical quantities such as Voice Onset Time (VOT) are nevertheless sorted into discrete percepts with precise VOT boundaries [Goldstone and Hendrickson, 2010] also accord well with this picture.

3.1.1 Harmony networks

Self-optimizing networks provide a plausible basis for the development of models that reside at the interface of cognition and neural computation. They key difficulty of applying them in a field like language has to do with the daunting productivity of domains where combinatoric possibilities come into play. The central free parameter in the RBF network architecture is the selection of the mean vectors $\boldsymbol{\mu}$, which specify the maxima associated with each attractor, each of which is defined by one of the subcircuits. It is

easy to see how classification problems with arbitrarily large numbers of classes can be built by composing arbitrary numbers of subcircuits.

Additionally, groups of independent classification tasks can be solved by solving each task in orthogonal subspaces of the input space, with distinct subnetworks handling each classification task. For instance, one can associate a maximum with each sequence of two licit English words (e.g. $[\texttt{cat}, \texttt{paw}]$, $[\texttt{fish}, \texttt{scale}]$, and $[\texttt{fish}, \texttt{paw}]$ are all sequences, but $[\texttt{tac}, \texttt{awp}]$ is not) by concatenating two networks, each of which solves the independent task "map the input in position $i \in \{1, 2\}$ to the nearest word" Formally, the resulting network has the property that $H([s_1, s_2]) = H_1(s_1) + H_2(s_2)$, where H_1 and H_2 are Harmony functions (1) for each subnetwork. This network has maxima all points where both H_1 and H_2 are at equilibrium—thus, a maximum associated with each pair of licit strings.

Take, however, the problem of choosing between competing combinatorial structures with the property of *mutually dependent choices*—e.g. the same task as above but now restricted to only plausible pairings. Now, $[\texttt{cat}, \texttt{paw}]$ and $[\texttt{fish}, \texttt{scale}]$ are sequences but $[\texttt{fish}, \texttt{paw}]$ and $[\texttt{cat}, \texttt{scale}]$ are not—commonsense semantics of two-word phrases. In this case, the selection of s_1 depends on the selection of s_2, and we should like for the following relations to hold:

$$H(\texttt{cat}, \texttt{paw}) > H(\texttt{fish}, \texttt{paw}) \qquad H(\texttt{fish}, \texttt{scale}) > H(\texttt{cat}, \texttt{scale}) \tag{3.1}$$

If Harmonies $H(s_1, s_2)$ sum linearly as $H(s_1) + H(s_2)$—as they do in RBF networks segmented into orthogonal optimization problems in the indicated way—then $H(\texttt{cat}) + H(\texttt{paw}) > H(\texttt{fish}) + H(\texttt{paw}) \Rightarrow H(\texttt{cat}) > H(\texttt{fish})$ and also $H(\texttt{fish}) + H(\texttt{scale}) >$

$H(\texttt{cat}) + H(\texttt{scale}) \Rightarrow H(\texttt{fish}) > H(\texttt{cat})$.[1]

3.2 Self-optimizing networks in combinatorial domains: Harmonic Grammar and Gradient Symbolic Computation

The tradition of Harmonic Grammar [Smolensky and Legendre, 2006] (HG) deals primarily with cases of this kind—where the choice of value for a given variable depends on the choice for another variable—via the introduction of connections between elements representing each variable.[2] HG combines two main elements: (a) a method for assembling structures from inventories of representations of the parts of the structure, and (b) a computational procedure—based on optimization of input patterns—for resolving input patterns into output patterns that satisfy certain constraints, which include mutual-

[1] This is isomorphic to the XOR problem.

[2] In the HG formalism, this is done by using quadratic functions network states. For instance, the $\texttt{fish} - \texttt{paw}$ problem can be solved in a 4-unit network with a quadratic Harmony function. There is one unit for each assignment of a words to a role, e.g. $[1, 0, 0, 0]$ for `fish`-as-first-word, $[0, 1, 0, 0]$ for `cat`-as-first-word, $[0, 0, 1, 0]$ for `scale`-as-second-word, and `paw`-as-second-word. There are symmetric weights between each of the four units in the network, yielding a weight matrix W:

$$\mathrm{W} = \begin{bmatrix} 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 \end{bmatrix}$$

W is built up systematically by placing a 1 in cells where the the corresponding vector elements "go together", and a -1 otherwise. For instance, $W_{1,3} = W_{3,1} = 1$ because "`fish`-as-first-word" goes with "`scale`-as-second-word", and $W_{2,3} = W_{3,2} = -1$ because `cat`-as-first-word" does not go with `scale`-as-second-word". The Harmony function—analogous to Eqn. (1) for the RBF network—is $\mathcal{H}_{\texttt{fish-paw}} = \boldsymbol{x}^\top W \boldsymbol{x}$. It can be verified that the inequalities in Eqn 3.1 hold for each of the corresponding network states. This shows how networks with this quadratic architecture can encode optimization problems exhibiting mutual dependence, allowing us to correctly score competing alternative structures. The additional problems of designing a network that computes the optimization, and does so while assigning steady states only to valid structures, is addressed in the subsequent text.

dependence relations such as those of 3.1. Element (a)—combinatorial assembly—is specified by an inventory of roles (structural positions), an inventory of fillers (say, the lexical inventory of a language), and an operation—the tensor product $\otimes$—which serves to associate fillers with roles. Let symbols be denoted in `typewriter` font and the corresponding vectors in **bold**, with $\mathtt{S} = \{\mathtt{f}_{i/\mathtt{r}_i}\}$ denoting a set of associations of fillers $\mathtt{f}_i$ to corresponding roles $\mathtt{r}_i$. Its embedding is a vector $\boldsymbol{s}$ assembled as

$$\boldsymbol{s} = \sum_i \boldsymbol{r}_i \otimes \boldsymbol{f}_i \tag{3.2}$$

with the role and filler vectors typically chosen to be linearly independent sets. Each $\boldsymbol{r}_i$ and $\boldsymbol{f}_i$ are vectors in the sets $\{\boldsymbol{r}_i\}, \{\boldsymbol{f}_i\}$ of embeddings of each symbol. Licit structures are thus required to have each role bound to exactly one filler, and each filler-role binding $\boldsymbol{r}_i \otimes \boldsymbol{f}_i$ must have coefficient 1 in the sum. To encode structures where only a subset of roles are bound to fillers (e.g. where structures are sequences of symbols with a given max-length), one can introduce a "null" filler as one of the vectors in the filler space, so that the constraint of one filler per role is satisfied. Element (b)—encoding of constraints on structures—is provided by parameters W and b—a weight matrix and bias vector, each in the vector space $R \otimes F \otimes F \otimes R$ and $R \otimes F$, respectively. States of the network reside in the product space $R \otimes F$, and the well-formedness of structures—the result of often conflicting constraints—is generated by second-order interactions between the components of the state space (the symmetric matrix W). Together, these parameters define the well-formedness measure Harmony over any $\boldsymbol{y} \in R \otimes F$ in the state space:

$$\mathcal{H}_0(\boldsymbol{y}) = \frac{1}{2}\left(\boldsymbol{y}^\top W \boldsymbol{y} + b^\top \boldsymbol{y}\right) \tag{3.3}$$

Sometimes, W is directly set to be negative-definite, which guarantees concavity across the entire state space. Alternatively, one can implement "Faithfulness" constraints penalizing the network state if it drifts too far from the initial input vector $\boldsymbol{x}$: $-\lambda(\boldsymbol{y}-\boldsymbol{x})^\top(\boldsymbol{y}-\boldsymbol{x})$. As long as the hyperparameter λ is set to be greater than the largest eigenvalue of W, the result is a concave quadratic function of $\boldsymbol{y}$. Then, $\mathcal{H}(\boldsymbol{y})$ has a unique maximum for any input $\boldsymbol{x}$. From these condition, networks with the self-optimization property can be derived. In reality, these formulations are largely interchangeable and come down to the definition of W, since in each case the resulting function hinges on a negative definite matrix—W itself in case 1 where the constraint is imposed on W directly, and $W - \lambda I$ in case 2 where the constraint is set on λ and referred to as Faithfulness. We show how this is so in Appendix A of Chapter 4.

While the concavity of H means that the unique $\boldsymbol{y}$ can analytically solved, it is readily seen that, although $\mathcal{H}_0$ defines well-formedness for all licit structures of the correct form 3.2, it also defines well-formedness for all other states in the product space, which are not in general thus expressible. In particular, the optimal network state $\hat{\boldsymbol{y}} = \text{argmax}_{\boldsymbol{y}}\mathcal{H}(\boldsymbol{y})$ will not. Relative probabilities between specific discrete structures are established by their location in the vector space and corresponding Harmonies, but the network does not generally resolve into the encoding of a single discrete structure. Hence, at the level of the neural implementation, establishing rankings between candidate structures was possible only by iterating across all of the candidates and calculating their harmonies—a procedure mimicked in the organization of symbolic tableaux for the purposes of linguistic analysis, but quite at odds with the core interest of Harmonic Grammar as a formalism that mediates between connectionism and grammar—that interest being in its promise of neurobiologically-plausible implementation of structure-selection in combinatorically-expressive domains, implemented as Harmony-maximization by gradient ascent.

Implementation of the right optimization dynamics is the product of more recent work under the name Gradient Symbolic Computation (GSC) [Smolensky et al., 2014, Tupper et al., 2018]. Here, the standard dynamics of Harmonic Grammar are coupled with additional dynamics that show preference for structures that are licit in having the form of 3.2. The architecture provides a systematic way of deriving structured networks with harmony landscapes that have two properties: (a) the Harmony function 3.3 ranks candidate structures according to their satisfaction of constraints encoded in the network parameters, and (b) equilibrium states of the network (local optima) are to be found only at points representing licit structures. This implies that network dynamics designed to follow the gradient of H will converge to—and only to—local optima (see fig. 3.4). Paired with a procedure analogous to simulated annealing—where the strength of the quantization parameter is gradually raised as the network computes along the gradient, with random perturbations—the network will, in the limit, tend to converge to the *global* optimum [Tupper et al., 2018].

Let $X_r \in \mathbb{R}^{n_r \times |R|}$ and $X_f \in \mathbb{R}^{n_f \times |F|}$ denote the matrices with the role vectors and filler vectors as rows. Stipulate that the sets of role and filler vectors $\{\boldsymbol{r}_i\}, \{\boldsymbol{f}_j\}$ span the corresponding spaces. When the roles and fillers are set to be linearly independent—and thus form a basis for each space—both are invertible, with ${X_r}^{-1}$ and ${X_f}^{-1}$ having as columns the dual vectors ${X_r}^{-1}{}_{[:,i]} = \boldsymbol{r}_i^*$ and ${X_f}^{-1}{}_{[:,i]} = \boldsymbol{f}_i^*$ for the corresponding fillers and roles. The *Quantization Harmony* $\mathcal{H}_q$ is defined [Smolensky et al., 2014] as the quartic function over $\boldsymbol{x} \in R \otimes F$:

$$\mathcal{H}_q(\boldsymbol{x}) = -\sum_i \left(-1 + \sum_j \left(\boldsymbol{r}_i^* \cdot \boldsymbol{x} \cdot \boldsymbol{f}_j^* \right)^2 \right)^2 - \sum_{ij} \left(\boldsymbol{r}_i^* \cdot \boldsymbol{x} \cdot \boldsymbol{f}_j^* - 1 \right)^2 \left(\boldsymbol{r}_i^* \cdot \boldsymbol{x} \cdot \boldsymbol{f}_j^* \right)^2 \quad (3.4)$$

Observe that each $\boldsymbol{x}$ in the network state space can be expressed as some $\boldsymbol{x} = \sum_{ij} \alpha_{ij} \boldsymbol{r}_i \otimes$

$\boldsymbol{f}_j$, where each $\alpha_{ij} = \boldsymbol{r}_i^* \cdot \boldsymbol{x} \cdot \boldsymbol{f}_j^*$. The first term says that the sum of the squared coefficients $\sum_j {\alpha_{ij}}^2$ for each role i is 1, and the second says that each $\alpha_{ij} = \boldsymbol{r}_i^* \cdot \boldsymbol{x} \cdot \boldsymbol{f}_j^*$ is either 0 or 1. Together, these terms set equilibria at points where where one and only one α_{ij} is 1 for each role $\mathtt{r}_i$, i.e. the points $\left\{\sum_i \boldsymbol{r}_i \otimes \boldsymbol{f}_j\right\}$ for any of the $|F|^{|R|}$ assignments of a single filler to each role. The total Harmony $\mathcal{H}$ is the sum of $\mathcal{H}_0$—the structural well-formedness measure—along with the quantization term $\mathcal{H}_q$—which requires states to have the right form 3.2:

$$\mathcal{H}(\boldsymbol{y}) = (1-q)\mathcal{H}_0(\boldsymbol{y}) + q\mathcal{H}_q(\boldsymbol{y}) \tag{3.5}$$

where $q \in [0,1]$ is a scalar known as the *quantization strength*. The sum of these terms yields a composite well-formedness measure with peaks at each discrete structure, the height of those peaks determined primarily by the *Core Harmony* 3.3 (fig. 3.4 provides an example).

GSC was devised for the general purpose of selecting structural descriptions in discrete combinatorial domains, i.e. where the outputs of computation necessarily correspond to discrete structures. The introduction of Quantization dynamics into the framework enforces this constraint while providing an explicit mechanism: gradient ascent paired with annealing—for resolving arbitrary inputs into a selection of discrete structures[3].

This is one of several methods by which one can obtain discrete-domain elements from continuous vector representations, the first being iteration over the possible structures. In the applied domain that is the subject of this thesis, in order to link arbitrary inputs to corresponding discrete structures, we use another: content retrieval via unbinding operations, combined with a cleanup memory. Instead of requiring that representations

[3]Given that the quantization strength q is sufficiently high, each terminus of computation (each optimum) uniquely identifies one of the licit structures.

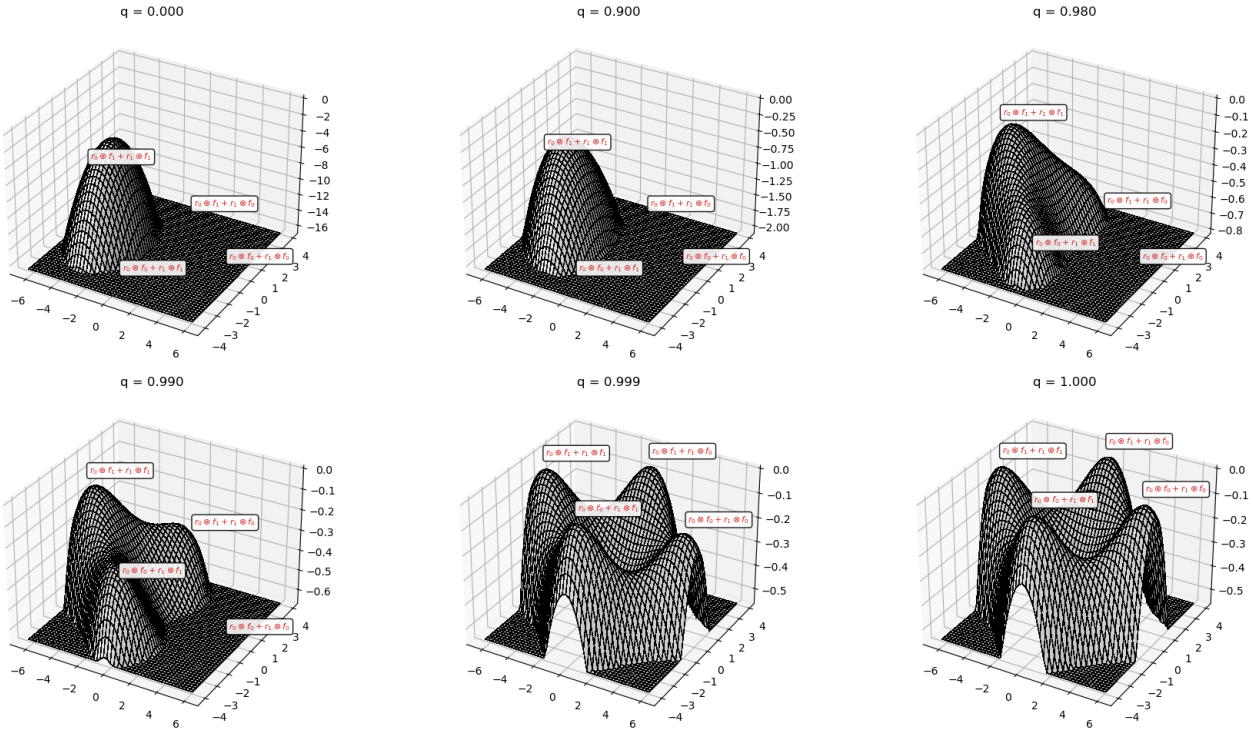


Figure 3.4: Harmony surface for a Harmonic Grammar defined over a filler-role inventory of two fillers and two roles, each component being 4-dimensional—leading to a 16-dimensional network state space for representing filler-role bindings. The network Harmony is depicted after projection of all points onto two principal components, at various settings of the quantization parameter q.

at the output of computation factor, one can obtain a discrete structure by simply forcing the network to produce an answer using a sort of "memory retrieval" operation modeled on the filler-role binding framework. In the standard case where the binding operation is the tensor product, and the dot product with dual vectors providing the corresponding unbinding operation, this allows an arbitrary vector $\boldsymbol{s}$ that lives in the right space—i.e. with $\boldsymbol{s} = \sum_{ij} s_{ij} \boldsymbol{r}_i \otimes \boldsymbol{f}_j$ and $\{\boldsymbol{r}_i \otimes \boldsymbol{f}_j\}$ being a basis for that space—to be addressed with each role, yielding $\boldsymbol{r}_i^* \cdot \boldsymbol{s} = \sum_j s_{ij} \boldsymbol{f}_j \equiv \tilde{\boldsymbol{f}}_i$. The "best" filler for the given role is then selected by choosing the filler that is closest to the result—for instance by taking $\text{argmin}_j \left\| \boldsymbol{f}_j - \tilde{\boldsymbol{f}}_i \right\|^2$.

A version of this procedure is used in the Harmonic Memories model elaborated in Chapter 5, where we adopt the two-way binding (fillers and roles) model of representation as a basis for representing graph components (specifically, entity "memories"), with a subsequent Harmony maximization procedure which models the satisfaction of learned semantic constraints on licit structures. In that case, the intuition is the same as the one described in the GSC framework above: an input representation taking the form of superposed filler-role bindings is subjected to optimization of an associated Harmony function. There are, however, significant departures with respect to the form of the computational *process*—modifications due in large part to the computational efficiency requirements of our applied setting: knowledge base completion. Maximizing the utility of GSC networks requires stochastic optimization of a highly nonlinear function with a (generally large) number of local minima that is exponential in the size of the memory (the number of superpositions allowed). For the m roles (with m the memory capacity) and a basis of fillers with size n, the set of licit structures has size n^m, with the baseline algorithm (iteration) for computing Harmonies for each structure having exponential time. Usually, structure-selection is performed using stochastic optimization of the current network state, with stochasticity introduced in order to escape local minima, as in Hopfield networks [Hopfield, 1982].

Stochasticity also has utility in computational modeling of cognitive processes. While the strict Harmony values of network states—its local and global optima and structure-rankings derived therefrom—are associated with grammatical *knowledge* rather than processing, i.e. computationally specifying the set of linguistically licit structures, stochastic optimization can model such variables as computation/processing time as well as comprehension or production errors. The GSC model has been profitably used in both domains. On the grammatical side, [Smolensky and Goldrick, 2016] characterize the phenomenon

of French laison using constraints encoded in a Harmony function, finding that the model can be parametrized in such a way as to yield grammatical predictions about which French phrases should be liaised. On the processing side, [Whan Cho et al., 2017] model incremental syntatic parsing using a GSC network. In addition to defining grammatical knowledge (the inventory of locally optimal states and their scores), they show how a network with different policies for augmenting the quantization parameter q across time lead to different processing dynamics as new words are added to an evolving parse, combining core grammatical knowledge with the computation speedup obtained from parsing a sentence as it is built up incrementally. They find that different q-update policies lead to different predictions about how a GSC parser will process garden path sentences—i.e. whether the network converges to local (garden path parses) or global minima (the correct parse). Stochasticity makes erroneous states available, and the q parameter models how the network interacts with noise to produce correct vs. incorrect parses in different proportion.

When the interest is primarily in modeling grammar, parsing or other cognitive phenomena within restricted domains, it is tractable to define network weights analytically and to perform optimization via simulation of the implied dynamical system. Our applications, however, require speedy training as well as inference.

Consider first what is required in order to define network weights for a GSC network. The key function is 3.4, in which each sum of filler-role bindings is identified with an optimum. Expanding out the first and largest term of 3.4 leads to a polynomial with $|\mathcal{R}|^2|\mathcal{F}|^2$ terms. For even a small optimization problem having, say, 10 roles and 100 fillers, this is a million products, each of which must be recomputed for all timesteps in the state-optimization during inference. This component of the GSC Harmony is the one that imposes the constraint that all output states be superpositions of discrete filler-

role bindings, and the result of suspending is that output states no longer correspond to discrete structures. However, as we discuss in Chapter 4, allowing the network to represent semantic modulation of input symbols as a function of their contexts leads to output representations whose structure is both useful and interpretable (e.g. Table 4.4). Thus, this gradience in output representations is in fact a virtue.

Most significantly from a computational point of view, it is not generally possible to provide linearly independent filler-role bases where relations number in the tens to hundreds and entities number in the tens of thousands. The required embedding sizes are not accessible. In addition, there are numerous *representational* constraints imposed by our application setting, in which roles correspond to relations which may have multiple target entities (e.g. feline has both cat and leopard as hyponyms), meaning that the desired outputs are not one-to-many (a function, as required by definition 3.4), but rather many-to-many (a relation). The simplifications we make are, therefore, largely unavoidable.

3.3 Conclusion

In this chapter, we have reviewed an array of neural networks that perform pattern-completion in the form of optimizing a dynamical system, elaborating on applications of these network to solving of combinatorial problems stated in terms of filler-role bindings. Networks of this general form provide the foundation for the models we develop in Chapters 4 and 5. What the models share is the definition of a Harmony function $\mathcal{H}$ assigning a score to each state of the network. Under appropriate constraints to the form of $\mathcal{H}$, the corresponding networks define dynamical systems with optima at each valid solution.

The overall framework of self-optimizing networks is quite general, and there are many

degrees of freedom in their design. The Radial Basis Networks developed in Section 3.1 rely on bilinear operations with a nonlinear transfer function and multiplicative connection weights. The GSC networks of Section 3.2 depend on $\mathcal{H}_q$, a quartic function, whose computation stated in neural terms requires synapses that integrate four-way products. An interesting question—but one that is ancillary for our purposes—is *which* networks of this general form are appropriate for modeling cognitive phenomena? A central constraint on such networks, as models of neural computation, should be their neural plausibility, and it is not straightforwardly clear that the networks reviewed in this chapter can meet the threshold of compatibility with known bioneural principles. We mention in passing, however, that the networks deployed in the subsequent two chapters have very clear neural models in the fairly strict sense that they rely only operations familiar from classical neural networks: symmetric bidirectional connection weights, with the linear transfer function defining the activation of each neural unit. While the focus here is primarily applied, it should not be understated that these networks can serve as a unifying principle for knowledge representation in a way that bridges the divide between cognition and its neural implementation.

Chapter 4

Gradient Graphs

4.1 Introduction

As they are conventionally analyzed, representations of semantic or linguistic data are "compositional": the meanings of complex representations are built up from the meanings of their constituent parts.[1] This idea has motivated numerous models of graph data deployed in knowledge base completion (KBC), in which embeddings of entities and relations are combined into composite representations—pairs of entities in a particular relation with one another—that are built up systematically from the constituent parts. But what happens when the whole is *not* a simple function of the parts? A natural case arises in the interpretation of Noun-Noun compounds. The contrasting senses of *vampire cat* (*a-cat-that-is-a-vampire*) and *vampire stake* (*a-stake-used-to-kill-a-vampire*) has as much to do with the compatibility of the constituent nouns occurring in a given relation than with the meanings of the individual components of the phrase.

Pursuing this line of thought, we propose **Gradient Graphs**, a neural network model

[1] An earlier version of this work was published in the Proceedings of the Society for Computation in Linguistics [Lalisse and Smolensky, 2019].

for KBC built on the principle that compositionally-obtained representations of semantic objects can be optimized to reflect context-specific aspects of the meanings of their constituents. The issue of context-conditioned, tokenized semantic representations has received little explicit attention in the KBC literature. However, precedents do exist. [Bordes et al., 2011] model context-sensitive entity senses by embedding relations as pairs of matrices (R_{lhs}, R_{rhs}) that linearly transform entity embeddings into pairs of embeddings defined by the relation and the entities' positions within it (the left-hand-side or right-hand-side). The distances of the resulting embeddings are then compared. [Socher et al., 2013] cope with the context-sensitivity of relation meanings by learning a $k \times d \times d$-dimensional tensor embeddings for each relation, letting their model represent polysemy by learning k versions of the relation represented in the k slices of its embedding tensor. The intuition underlying this approach is that, for instance, the relation `has_part` has a different sense when applied to a biological organism than when predicated of a company. While the former has parts like organs and limbs, the latter has parts like subsidiaries and workers, which occupy very different parts of the semantic space. Each relational slice is then responsible for learning the compatibility of arguments within particular semantic subspaces.

In contrast to these other works, our approach is more radical in the sense that our context-sensitive representations of knowledge base entries are not just computed from the entries' constituent elements (entity and relation embeddings), but are instead the result of a representation-optimization procedure that balances compositionally-derived representations with general knowledge about the characteristics of well-formed semantic structures. We show that this additional "supracompositional" processing, in addition to yielding sizable accuracy improvements over the compositional models we apply it to, leads to embeddings of entity tokens with interpretable characteristics.

4.1.1 Layout of the chapter

Section 4.2 lays out the general framework, which is compatible with a variety of implementations. Section 4.3 presents two compositional embedding models proposed in the literature. We adapt these models to construct compositional embeddings, and in Section 4.4 report evaluations of Gradient versions of these models. Section 4.5 discusses the characteristics of the resulting semantic representations in greater detail, as well as their role in assisting inference. Section 4.6 concludes. Technical details about the model and the implementations are given in the Appendices.

4.2 Optimization of semantic tokens

The hypothesis underlying the approach we propose is that noncompositional effects in knowledge base data can be modeled by subjecting candidate facts to a process of optimization with respect to a set of learned semantic coherence conditions. These semantic coherence conditions, encoded in a symmetric matrix, map out the covariance structure of the semantic space, indicating which semantic features are likely to co-occur with one another. The embedding of a given triplet is then the vector obtained by optimizing the semantic coherence of the the triplet embedding.

We first lay out the model in abstract form, before introducing particular implementations. Let $x \in \mathbb{R}^d$ be a d-dimensional embedding of a knowledge base triplet (e_ℓ, r, e_r) obtained as some function f_{comp}—the **composition function**—of the embeddings of the left and right entities as well as the relation $\boldsymbol{r}$. Section 4.3 provides several models for constructing the triplet embedding $\boldsymbol{x}$. Also, let h be a d-dimensional vector giving the internal ("hidden") state of the network. The *Harmony* of an internal state h of the

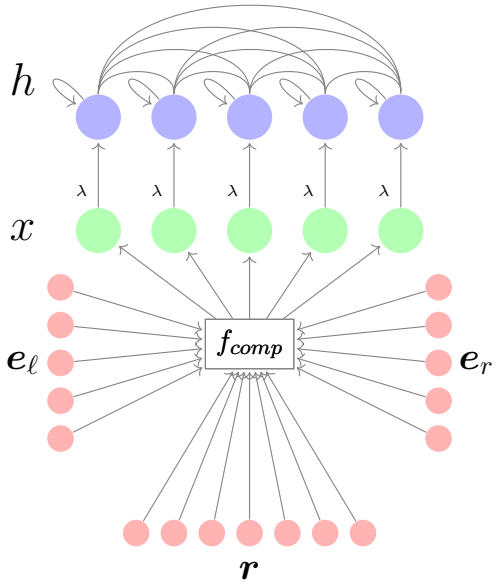


Figure 4.1: Gradient Graph as a recurrent neural network. In addition to bias terms (omitted in the figure) and self-connections, hidden units are densely connected to one another via a layer of connections with symmetric (undirected) weights, and receive constant input weighted by λ from a single unit in the input layer. The composition function $f_{comp}(\boldsymbol{e}_\ell, \boldsymbol{r}, \boldsymbol{e}_r)$, which differs between implementations, computes a **compositional embedding** $\boldsymbol{x}$, which is fed into a **hidden layer** $\boldsymbol{h}$ of the network. The continuous-time dynamics of this network compute an internal representation $\hat{\boldsymbol{h}}$ of the input triplet that is optimal with respect to the Harmony 4.1—a measure of the triplet's semantic well-formedness.

network with respect to the triplet embedding x is

$$\mathcal{H}(\boldsymbol{h}, \boldsymbol{x}) = \frac{1}{2}\left[\boldsymbol{h}^\top \mathbb{W}\boldsymbol{h} + \boldsymbol{b}^\top \boldsymbol{h} - \lambda(\boldsymbol{h}-\boldsymbol{x})^\top(\boldsymbol{h}-\boldsymbol{x})\right] \tag{4.1}$$

where $\mathbb{W}$ is a $d \times d$ weight matrix with $\mathbb{W} = \mathbb{W}^\top$ and $\boldsymbol{b}$ is a bias vector, both learned. 4.1 is composed of two terms: **Core Harmony**, a measure of the semantic coherence of the state vector $\boldsymbol{h}$, and **Faithfulness**, a penalty incurred due to the state $\boldsymbol{h}$'s deviation from the compositional triplet embedding $\boldsymbol{x}$. λ is a hyperparameter that controls the magnitude of the penalty incurred for straying from $\boldsymbol{x}$.

$\mathcal{H}(\boldsymbol{h}, \boldsymbol{x})$ may be rewritten as 4.2.

$$\mathcal{H}(\boldsymbol{h}, \boldsymbol{x}) = \frac{1}{2}[\boldsymbol{h}^\top (\mathbb{W} - \lambda I)\, \boldsymbol{h} + (\boldsymbol{b} + 2\lambda \boldsymbol{x})^\top \boldsymbol{h} - \lambda \boldsymbol{x}^\top \boldsymbol{x})] \tag{4.2}$$

If λ is greater than the largest eigenvalue of $\mathbb{W}$, then $V = \mathbb{W} - \lambda I$ is negative-definite, and $\mathcal{H}(\boldsymbol{h}, \boldsymbol{x})$ has a unique global optimum $\hat{\boldsymbol{h}} = \text{argmax}_{\boldsymbol{h}\mathcal{H}(\boldsymbol{h}, \boldsymbol{x})}$ for each $\boldsymbol{x}$. In closed form, this global optimum is

$$\mu(\boldsymbol{x}) = -V^{-1} \left(\frac{1}{2}\boldsymbol{b} + \lambda \boldsymbol{x} \right) \tag{4.3}$$

which depends only on the network parameters and on $\boldsymbol{x}$. The expression $\mu(\boldsymbol{x})$ comes from observing that $\hat{\boldsymbol{h}}$ is the mean of a Gaussian distribution with inverse covariance matrix V, which implies that $\hat{\boldsymbol{h}}$ is the most probable state $\boldsymbol{h}$ of the network with respect to the probability distribution over the state space defined by $p(\boldsymbol{h}|\boldsymbol{x}) \propto \exp\{\mathcal{H}(\boldsymbol{h}, \boldsymbol{x})\}$ (see Appendix A). We take the token embedding for a triplet $\boldsymbol{x}$ to be $\mu(\boldsymbol{x})$, which is the most semantically coherent triplet embedding given the compositional triplet $\boldsymbol{x}$. In the limit as $\lambda \to \infty$, $\mu(\boldsymbol{x})$ is just $\boldsymbol{x}$ itself. Let $\lambda_{\mathbb{W}}$ denote the largest eigenvalue of $\mathbb{W}$; then as $\lambda \to \lambda_{\mathbb{W}}$, $\mu(\boldsymbol{x})$ may become arbitrarily far from the triplet embedding $\boldsymbol{x}$.

A Gradient Graph may be viewed as a neural network with weight matrix $\mathbb{W}$ and bias vector $\frac{\boldsymbol{b}}{2}$, where the synaptic weights $\mathbb{W}$ specify a feedback layer through which the values of the hidden state units affect one another. The construction is as follows. We stipulate that the hidden state of the network follows the gradient of Harmony over time:

$$\frac{d\boldsymbol{h}}{dt} = \frac{\partial \mathcal{H}(\boldsymbol{x}, \boldsymbol{h})}{\partial \boldsymbol{h}} \tag{4.4}$$

Therefore,

$$\begin{aligned}\frac{dh_i}{dt} &= \frac{d}{dh_i}\frac{1}{2}\left[h^\top \mathbb{W} h + b^\top h - \lambda \left\|x - h\right\|^2\right] \\ &= \frac{d}{dh_i}\frac{1}{2}\left[\sum_{jk} h_j \mathbb{W}_{jk} h_k + b_i h_i - \lambda (x_i - h_i)^2\right] \\ &= \frac{1}{2}\left[\sum_j h_j \mathbb{W}_{ji} + \sum_k \mathbb{W}_{ik} h_k\right] + \frac{b_i}{2} + \lambda x_i - \lambda h_i\end{aligned}$$

The above specifies the connectivity of a network whose hidden units have the linear transfer function ($f(\iota) = \iota$), bias $\frac{\boldsymbol{b}}{2}$ and external input $\boldsymbol{x}$ (weighted by λ). Each $\boldsymbol{h}_i$ also receives self-inhibitory input weighted by $-\lambda$, as well as inputs $\mathbb{W}_{ij}\boldsymbol{h}_j$ from each $\boldsymbol{h}_j$. The symmetry of $\mathbb{W}$ implies that each term $\mathbb{W}_{ij}\boldsymbol{h}_j = \boldsymbol{h}_j \mathbb{W}_{ji}$ occurs twice, so that the factor of $\frac{1}{2}$ cancels. This connectivity structure is illustrated in Figure 4.1.

4.2.1 Relation to Harmonic Grammar

In addition to being globally optimal with respect to the Harmony function $\mathcal{H}(\boldsymbol{h}, \boldsymbol{x})$ conditioned on a particular input $\boldsymbol{x}$, $\mu(\boldsymbol{x})$ is the unique fixed point of this network's state-evolution dynamics. It is interesting to note that such networks are the connectionist foundation for Harmonic Grammar (HG) and Optimality Theory (OT) in linguistics [Smolensky and Legendre, 2006], where the dynamics of a neural network perform optimization over internal representations of an input structure. Appropriate output representations are then selected in accordance with well-formedness constraints encoded in the network parameters. There, the output representation balances Faithfulness to the input (an Underlying Form) and the network's knowledge about the characteristics of well-formed structures in general.

Similarly, it is appealing to conceptualize the hidden layer of a Gradient Graph network as cleaning up a knowledge base triplet by subjecting it to semantic well-formedness conditions. The optimal triplet $\mu(\boldsymbol{x})$ is then the point to which the network converges in the limit of infinite computation time. However, our model differs from typical implementations of HG and OT in that the optimal structure $\mu(\boldsymbol{x})$ does not, in general, decompose into a unique combination of the input constituents (entity and relation embeddings). The resulting representations are in this sense gradient, rather than being the product of a combination of discrete objects. Furthermore, Gradient Graphs are, to our knowledge, the first application of these ideas to the automatic learning of an appropriate semantic optimization function from a large amount of data.

4.2.2 Comparison with translation-based approaches

Like a large class of *Translation*-based models [Bordes et al., 2011, Yoon et al., 2016, Lin et al., 2015, Ji et al., 2016], our inference procedure consists of the application of an affine transformation to an input x (Equation 4.3), which is then scored using some regular operation. In our case, this scoring function is quadratic. A particular close cousin is the bilinear Semantic Matching Enery (SME) method of [Bordes et al., 2014], which learns a global three-mode tensor $\mathbb{W}$ that, when dotted along the third mode with a relation embedding $\boldsymbol{r}$, yields a relation-specific matrix $\mathbb{W}_r$. Along with learned left and right bias vectors $\boldsymbol{b}_\ell$ and $\boldsymbol{b}_r$, this weight matrix is fed into the bilinear scoring function 4.5:

$$\text{score}_{\text{SME}}(\boldsymbol{e}_\ell, \boldsymbol{r}, \boldsymbol{e}_r) = (\mathbb{W}_r \boldsymbol{e}_\ell + \boldsymbol{b}_\ell)^\top (\mathbb{W}_r \boldsymbol{e}_r + \boldsymbol{b}_r) \tag{4.5}$$

Expanding out this expression, we get 4.6.

$$\boldsymbol{e}_\ell{}^\top \mathbb{W}_r{}^\top \mathbb{W}_r \boldsymbol{e}_r + \boldsymbol{b}_\ell{}^\top \mathbb{W}_r \boldsymbol{e}_r + \boldsymbol{b}_r{}^\top \mathbb{W}_r \boldsymbol{e}_l + \boldsymbol{b}_\ell{}^\top \boldsymbol{b}_r \tag{4.6}$$

The relation-specific bilinear form $\mathbb{W}_r{}^\top \mathbb{W}_r$ is, like our global $\mathbb{W}$ matrix, symmetric. The remaining terms, apart from the constant $\boldsymbol{b}_\ell{}^\top \boldsymbol{b}_r$, compute a pair of relation-specific bias vectors $\boldsymbol{b}_\ell{}^\top \mathbb{W}_r$ and $\boldsymbol{b}_r{}^\top \mathbb{W}_r$ applied to the pair of entity embeddings. The resulting Energy function used to score triplets has more than a passing similarity to our Harmony function 4.1 when $\lambda = \infty$ and, thus, no optimization takes place.

A distinctive characteristic of our approach in relation to these structurally similar models is that the transformation undergone by a Gradient Graph triplet is directly connected to the well-formedness criterion according to which triplets are evaluated in inference. As illustrated in the Discussion, our transformation of a compositional triplets using learned well-formedness criteria leads to two kinds of triplet embeddings: compositionally obtained *type* embeddings, and contextually optimized *token* embeddings. In qualitative and quantitative analyses of the learned representation, we see (1) that the space of compositionally obtained triplet embeddings has a reasonable structure, independently of the optimizing transformation, that is already sensitive to the context supplied by the relation, and (2) that semantic optimization improves these compositional representations in recognizable ways. Interestingly, improving triplets with respect to the Harmony function does *not* uniformly place them in regions that are high-Harmony in a global sense. In fact, we find that whereas positive triplets end up close to other positive triplets, plausible but negative triplets tend to be detained in clusters with other negative instances (Tables 4.2 and 4.4).

4.2.3 Harmony-Maximization as context-modulation: An intuition

Although our main problem domain is knowledge representation (a "language of thought") rather than language itself, a version of the token-type distinction we use here has precedent appears in the linguistic analysis of nouns in constructions where they are considered in certain aspects. Take for instance the following examples, adapted from [Asher, 2011]:

(1) **Copredication**

a. Lunch was delicious but took forever.

b. The books are heavy but informative.

Clearly, *lunch* can appear as the argument of both predicates *being delicious* and *taking forever*. But the predicates applied to it are predicates of categorically different types of things. Lunch *was delicious* in its aspect as a physical substance, but lunch *took forever* in its aspect as an event. Similarly for *the books*: they are *heavy* as physical substances, but *informative* when considered as abstract informational entities. "Copredicative" sentences like these are interesting because they require interpreting *lunch* as the same entity when it is *being delicious* and *taking forever*. Yet the event of *lunch* does not have a taste, nor does the meal itself have a duration [Liebesman and Magidor, 2019]. Indeed, the licitness of copredication is a key datum in establishing that differing senses of a word are examples of polysemy and not homophony. Take the well-known case of *bank* as in an issuer of credit and *bank* as in a riverbank, which are homophones. Copredication fails in this case:

(2) a. *The bank that holds my mortgage is very slippery when it rains.

b. The bank that holds my mortgage is on 4th Avenue.

Put in our terms, while *bank* the physical building is sufficiently close to *bank* the abstract legal entity that instances of the two can be morphed into one another, this is not the case for the homophonous senses of *bank*.

A related phenomenon is coercion. Take the two sentences sentences below, also from [Asher, 2011]:

(3) **Coercion**

a. Julie enjoyed the book.

⇒ Julie read the book.

b. The goat enjoyed the book.

⇒ The goat ate the book.

Clearly *the book* in the context of *Julie* refers to the abstract object with its informational content, and in the context of *the goat*, to the physical substance of which the book is composed. This difference explains the inferences that can be drawn from each of these sentences: the word *enjoyed* has accommodated the context of the head noun.

A characterization of these phenomenon in our terms distinguishes between *enjoy* in both predications in the following way. *Enjoy* is initialized to a common representation (vector) $\boldsymbol{e}_{\texttt{enjoy}}$. It occurs in two predication structures: $(\texttt{julie}, \texttt{enjoy}, \texttt{book})$, and $(\texttt{goat}, \texttt{enjoy}, \texttt{book})$. There are corresponding vectors for each element, with `enjoy` in both contexts initialized to a common vector $\boldsymbol{e}_{\texttt{enjoy}}$. When occurring in the structure, $\boldsymbol{e}_{\texttt{enjoy}}$ is subjected to the semantic type requirements of the other arguments in the

structure. julie is capable of consuming the book's most salient characteristic—the information—while the goat can only interact with the book as a physical object. So while the initial representation of enjoy is the same, the final representation must take into account the selectional constraints associated with the other vectors. Our mechanism for instantiating this difference is optimization of the lunch vector with respect to these semantic constraints as embodied in the weights of $\mathbb{W}$. In terms of the model, the Harmony surfaces for the two compositional inputs are different because of the Faithfulness term, and so the optimized version of the Julie triplet will end up somewhere close to "Julie read the book", and the goat version ends up close to "The goat ate the book". This example is depicted in Figure 4.2.

An interesting aspect of our model is that it does not only produce a score for the involved predications, but directly modifies the input vector representation of enjoy into a one that is modulated by the context (e.g. *enjoy-as-reading* and *enjoy-as-eating*), with the contextualized facts involving enjoyment now evaluated with $\boldsymbol{e}_{\texttt{enjoy}}$ in its optimized form. We make use of this attribute in the analyses of Section 4.5 by inspecting both the type and token representations of entities.

4.3 Compositional and Gradient Models

Optimization with respect to the Harmony measure $\mathcal{H}$ can be implemented wherever we can construct a triplet embedding $\boldsymbol{x}$. In our experiments, we apply Harmonic optimization of triplet representations to two compositional embedding models drawn from the knowledge base completion literature: DistMult and HolE. Both models specify a scoring function for triplet embeddings obtained via operations applied to embeddings of the three triplet components—two entity vectors and a relation vector—with no ad-

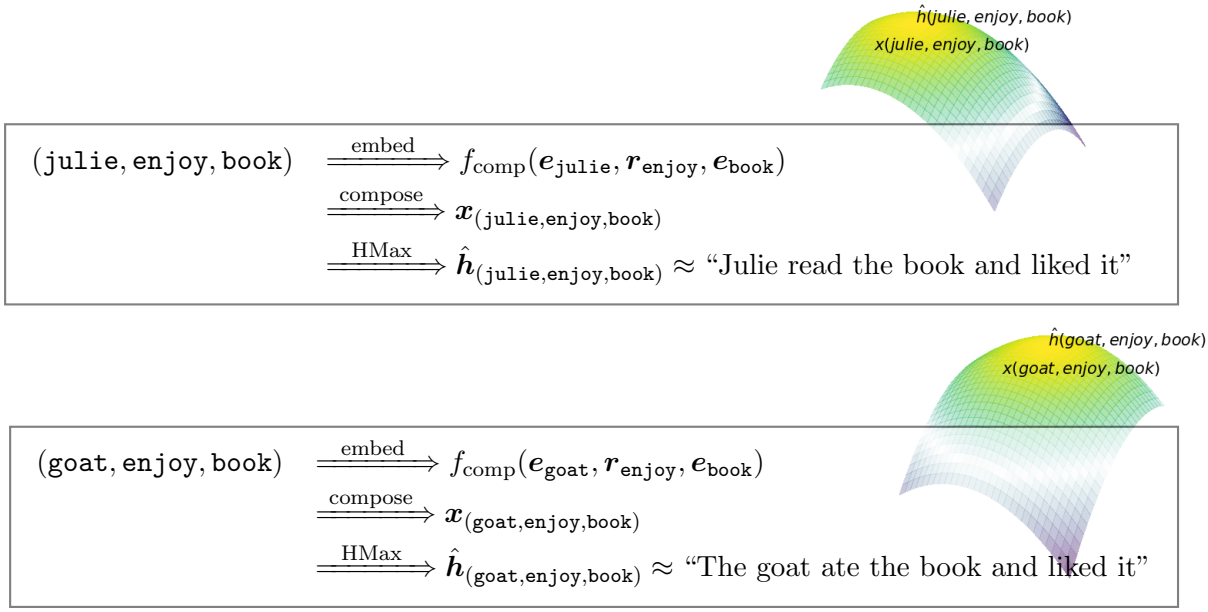


Figure 4.2: Coercion as optimization in a Gradient Graph.

ditional learned components apart from these representations of the triplet constituents. We take the terms occurring in these scoring functions to be components of the *representation* of the triplet, specifying what information about the triplet elements is important to evaluating the triplet's quality. Hence, we constructed Harmonic triplet embeddings according to the desideratum that every term occurring in the basic method's scoring function should also appear in the triplet representation $\boldsymbol{x}$ in the Harmonic model. For instance, the score of a DistMult triplet is a sum of three-way products of the corresponding elements of the embeddings e_ℓ, r, and e_r. Setting the products $[\mathrm{e}_\ell]_i[\mathrm{r}]_i[\boldsymbol{e}_r]_i$ to appear in our compositional triplet embeddings (as in Eqn 4.8) satisfies this desideratum.

DistMult [Yang et al., 2015] is a baseline model for scoring knowledge base triplets using the scoring function 4.7:

$$score_{\textsc{DistMult}}(\boldsymbol{e}_\ell, \boldsymbol{r}, \boldsymbol{e}_r) = \boldsymbol{e}_\ell{}^\top \ \mathrm{diag}(\boldsymbol{r}) \ \boldsymbol{e}_r \tag{4.7}$$

where $\boldsymbol{e}_\ell, \boldsymbol{r}, \boldsymbol{e}_r$ are d-dimensional embeddings and $\mathrm{diag}(\boldsymbol{r})$ is the $d \times d$-dimensional matrix obtained by arranging the elements of $\boldsymbol{r}$ along the diagonal. [Kadlec et al., 2017] have recently shown that DistMult can outperform many more elaborate scoring functions when hyperparameters are properly optimized, making it a strong baseline comparison for the method we propose. In addition, DistMult often occurs as a subcomponent in state-of-the-art KBC models—e.g. [Schlichtkrull et al., 2017, Toutanova et al., 2015]. From this starting-point, we construct Harmonic DistMult (HDistMult) by setting the triplet embedding $\boldsymbol{x}$ to the elementwise multiplication of the relation and the pair of entity vectors:

$$\boldsymbol{x}_{\mathrm{HDM}} = \boldsymbol{e}_\ell \odot \boldsymbol{r} \odot \boldsymbol{e}_r \tag{4.8}$$

where $\odot$ denotes elementwise multiplication.

Holographic Embeddings (HolE) were introduced by [Nickel et al., 2016] building on theoretical work by [Plate, 1995], as a means of constructing compressed tensor product representations of relational triplets. The method computes the score for a triplet $(\boldsymbol{e}_\ell, \boldsymbol{r}, \boldsymbol{e}_r)$ from the similarity between a relation vector and the circular correlation $\boldsymbol{e}_\ell \star \boldsymbol{e}_r$ of the entity vectors and a relation vector:

$$score_{\text{HolE}}(\boldsymbol{e}_\ell, \boldsymbol{r}, \boldsymbol{e}_r) = \boldsymbol{r}^\top (\boldsymbol{e}_\ell \star \boldsymbol{e}_r) \tag{4.9}$$

where the circular correlation of $\boldsymbol{e}_\ell$ and $\boldsymbol{e}_r$ is computed as 4.10.

$$\boldsymbol{e}_\ell \star \boldsymbol{e}_r = \mathcal{F}^{-1}\left(\overline{\mathcal{F}(\boldsymbol{e}_\ell)} \odot \mathcal{F}(\boldsymbol{e}_r)\right) \tag{4.10}$$

where $\mathcal{F}$ and $\mathcal{F}^{-1}$ denote the Fourier Transform and its inverse, and $\overline{\mathcal{F}(\boldsymbol{e}_\ell)}$ is the complex conjugate of $\mathcal{F}(\boldsymbol{e}_\ell)$, which reverses the sign of the imaginary terms of the Fourier-transformed vectors.[2] Circular correlation is asymmetric ($\boldsymbol{e}_\ell \star \boldsymbol{e}_r \neq \boldsymbol{e}_r \star e_\ell$)—allowing it to model asymmetric relations—and the result of the operation has the same dimensionality as the input vectors, while still carrying information about which pair of entities was bound together via correlation.

We construct Harmonic HolE (HHolE) triplet embeddings via elementwise mul-

[2]The Fourier transform redescribes a function of time into its frequency components. In the context of holographic embeddings, its utility comes from the *Convolution Theorem*, which states that convolution in the time domain corresponds to elementwise multiplication in the frequency domain. This is useful in actual computations. The circular correlation—which consists of convolution with a time-reversed signal—can also be computed as a sum over off-diagonals of the tensor product of vectors, with time complexity $\mathcal{O}(d^2)$. In contrast, the Fast Fourier Transform (FFT) has time complexity $\mathcal{O}(n \log n)$ [Nickel et al., 2016], though see Chapter 2's remarks on the computational efficiency of HRRs in the presence of anisotropy.

tiplication of relation vectors with the correlated pair of entity vectors:

$$\boldsymbol{x}_{\text{HHOLE}} = \boldsymbol{r} \odot (\boldsymbol{e}_\ell \star \boldsymbol{e}_r) \tag{4.11}$$

In both Harmonic models, the score for a candidate triplet $(\boldsymbol{e}_\ell, \boldsymbol{r}, \boldsymbol{e}_r)$ with embedding x is calculated by taking the Harmony of its optimal instantiation, $\hat{\boldsymbol{h}} = \mu(\boldsymbol{x})$, i.e.

$$\text{score}(\boldsymbol{x}) = \mathcal{H}(\mu(\boldsymbol{x}), \boldsymbol{x}) \tag{4.12}$$

In the experiments, we train our networks using the log-softmax objective with negative sampling. For each positive training example $(\boldsymbol{e}_\ell, \boldsymbol{r}, \boldsymbol{e}_r)$ with embedding $\boldsymbol{x}$, we construct N negative examples $(\tilde{\boldsymbol{e}}_\ell^n, \tilde{\boldsymbol{r}}^n, \tilde{\boldsymbol{e}}_r^n)$ obtained by deleting either the left or right entity of the true triplet and replacing it with a randomly sampled entity vector. Let $\tilde{\boldsymbol{x}}^n$ denote the embedding of the nth negatively sample triplet $(\tilde{\boldsymbol{e}}_\ell^n, \tilde{\boldsymbol{r}}^n, \tilde{\boldsymbol{e}}_r^n)$. The training objective is then to minimize 4.13:

$$\mathcal{L}_{\mathcal{H}}(\boldsymbol{e}_\ell, \boldsymbol{r}, \boldsymbol{e}_r) = -\log \frac{\exp\{\mathcal{H}(\mu(\boldsymbol{x}), \boldsymbol{x})\}}{\exp\{\mathcal{H}(\mu(\boldsymbol{x}), \boldsymbol{x})\} - \sum_{n=1}^{N} \exp\{\mathcal{H}(\mu(\tilde{\boldsymbol{x}}^n), \tilde{\boldsymbol{x}}^n)\}} \tag{4.13}$$

This has the effect of increasing the Harmony of positive examples relative to negative samples. The learning rule is thus Harmony-maximizing: the network parameters maximize the well-formedness of the positive examples relative to negative samples.

	FB15k						WN18					
		Rank		Hits@				Rank		Hits@		
Model	λ	**MR**	**MRR**	**1**	**3**	**10**	λ	**MR**	**MRR**	**1**	**3**	**10**
DistMult	-	-	.350	-	-	.577	-	-	.830	-	-	.942
Ensemble DM†	-	36	**.837**	**.797**	-	**.904**	-	457	.790	**.784**	-	.950
DistMult*	-	28	.710	.605	.792	.876	-	220	.825	.714	.938	.950
HDistMult	∞	**23**	.806	.751	**.845**	.898	∞	**164**	**.841**	.740	**.943**	**.955**
HDistMult	50.0	**23**	.742	.661	.799	.881	3.0	184	.831	.732	.931	.945
HolE	-	-	.524	.402	.613	.739	-	-	.938	.930	**.945**	.949
HolE*	-	39	.409	.289	.464	.647	-	205	.916	.893	.936	.946
HHolE	∞	32	.682	.575	.763	.850	∞	293	.919	.903	.934	.942
HHolE	1.0	**21**	**.796**	**.727**	**.848**	**.901**	2.0	**183**	**.939**	**.931**	**.945**	**.951**

Table 4.1: Results on FB15k and WN18. The results from the original DistMult and HolE models are drawn from [Yang et al., 2015] and [Nickel et al., 2016]. Our reimplementations* of DistMult and HolE differ in numerous details from those in the original papers (see Appendix B for technical details). Ensemble DistMult† refers to the hyperparameter-optimized Ensemble (product of experts) reimplementation of DistMult proposed by [Kadlec et al., 2017]. For each model, we report Mean Rank (MR) and Mean Reciprocal Rank (MRR), as well as Hits@N for $N \in \{1, 3, 10\}$. Hits@N denotes the fraction of test instances in which the true triplet completion had rank less than or equal to N. The best results within each category (DistMult and HolE) are marked in **bold**, and the best results overall are additionally **underlined**.

4.4 Experiments

We evaluated Gradient Graphs using the standard WN18 and FB15k datasets [Bordes et al., 2013]—which are subsets of the WordNet [Miller, 1995] and Freebase [Bollacker et al., 2008] databases—on the Entity Reconstruction task. In Entity Reconstruction, the network ranks completions of triplets $(\,\cdot\,, \boldsymbol{r}, \boldsymbol{e}_r)$ and $(\boldsymbol{e}_\ell, \boldsymbol{r}, \,\cdot\,)$ with deleted left and right entities. The model is successful if it ranks the true triplet above other candidate completions. We report results in the *filtered* evaluation setting [Bordes et al., 2013], in which a test triplet is only ranked against triplets that do not occur in the database. The rank of a test triplet is thus the rank of the *first* correct answer to the query. For both DistMult and HolE, we report the originally reported results alongside results for our reimplementations, comparing these models with our Harmonic variants HDist-

MULT and HHOLE with and without optimization of hidden layer representations. The Harmonic models with $\lambda = \infty$ have the Harmony function $\mathcal{H}(\boldsymbol{x}, \boldsymbol{x})$, i.e. where the hidden representation is just the compositional embedding itself and the Faithfulness penalty in (4.1) is 0.

Our models used 256- to 512-dimensional embeddings and manually tuned values of the hyperparameter λ. In all models, entity and relation embeddings were normalized to $\|\boldsymbol{v}\| = 1$. We do not regularize parameters, but instead set an upper bound $\lambda - \epsilon$ (ϵ a small constant) on the l_2 norm of the weight matrix $\mathbb{W}$, which helps constrain the spectral norm (maximum eigenvalue) of $\mathbb{W}$ to remain lower than λ. This may be seen as adopting a uniform prior on weight matrices lying within the n-ball with squared radius $\lambda - \epsilon$. Importantly, this procedure keeps the matrix $V = \mathbb{W} - \lambda I$ negative-definite—a necessary condition for the existence of a unique optimum for $\mathcal{H}(\boldsymbol{h}, \boldsymbol{x})$.

Results from the experiments are reported in Table 4.1. Overall, we found that models using our quadratic scoring function 4.1 to perform best across the board. This effect was particularly seen in more stringent evaluation criteria—Hits@1 and Hits@3—leading to, for instance—a 15% improvement in Hits@1 (accuracy) on FREEBASE between our DISTMULT reimplementation and quadratic HDISTMULT ($\lambda = \infty$). The bestDISTMULT models were those with high λ values; however, within-model comparison of HOLE shows dramatic improvements from including the optimization component—a 32% increase in FB15K accuracy between the results of [Nickel et al., 2016] and our HHOLE with a permissive λ-criterion of 1.0.

4.5 Discussion

In part, the appeal of our supracompositional representations stems from their ability to produce emeddings of *tokens* of semantic objects—that is, embeddings that take into account the context of a particular instance of a semantic type. Tokenized embeddings have proven useful in various settings. For instance, [Dasigi et al., 2017] construct token embeddings by superposing learned vectors for WORDNET senses in ratios determined by a probability distribution computed from the context. The resulting representation is a context-weighted sum of discrete senses drawn from a hand-crafted ontology. Closer to our approach, [Belanger and Kakade, 2015] model text as a linear dynamical system that generates texts through transitions of a continuous-state, discrete-time dynamical system across time. Estimates of the system's most probable internal state can then be extracted as an embedding of the tokens, which prove useful in language modeling and other downstream tasks.

In our framework, *types* correspond to static entity and relation embeddings that are the input to f_{comp}, and the triplet embeddings resulting from their combination. *Token* triplet embeddings are produced by optimization of the hidden layer of a GGRAPH. To understand the effect of optimizing the hidden layer of a GGRAPH both on its learned representations and on its performance in inference, we used the best-performing trained HHOLE model to produce token embeddings of database triplets in order to inspect their semantic neighborhoods.

For a given compositional triplet embedding $f_{comp}(\boldsymbol{e}_\ell, \boldsymbol{r}, \boldsymbol{e}_r) \equiv \boldsymbol{x}$, we first computed the optimized triplet representation $\mu(x) \equiv \hat{h}$ using Equation 4.3. Treating $\hat{\boldsymbol{h}}$ as the contextually optimal (token) embedding of the triplet $(\boldsymbol{e}_\ell, \boldsymbol{r}, \boldsymbol{e}_r)$, we then examined the semantic neighborhood by computing the 5 closest optimized embeddings in the context of

the same relation. Table 4.4 shows the semantic neighborhoods of compositional triplets $\boldsymbol{x}$ and optimized triplets $\hat{\boldsymbol{h}}$ for different possible completions of a number of queries. Rows 1 and 2 display completions of the query $(\,\cdot\,, \texttt{office\_position\_or\_title}, \texttt{US\_President})$, and Rows 2 and 4 consider the neighborhood of the entity embedding of *Bob Dylan* in the context of queries about his profession ($\mathtt{e}_\ell = \texttt{Bob\_Dylan}, \mathtt{r} = \texttt{has\_profession}$) while varying the profession $\mathtt{e}_r$ (Table 4.5. This illustrates how the representation of *Bob Dylan* varies across his different professional guises.

The tables illustrates the utility of token embeddings in inference. Token embeddings of *George W. Bush* and *Barack Obama* in the context of a query about their having held the office of U.S. president are in semantic neighborhoods with a greater density of true instances of U.S. Presidents than their type embeddings. The negative examples *John McCain* and *Hilary Rodham Clinton* have type embeddings that are close to actual presidents. This is sensible since, for instance, Hilary Rodham Clinton is married to Bill Clinton—one of her nearest neighbors. But both of the negative examples' token embeddings have neighborhoods that are mostly cleared of actual presidents—despite having type embedding neighborhoods that are relatively dense with presidents.

Turning to Table 4.5, we note that *Bob Dylan*'s type embedding is already in a neighborhood dense with singer-songwriters. It is appropriate, then, that this neighborhood undergoes no change apart from minor re-ranking when the triplet (`Bob_Dylan`, `has_profession`, `singer-songwriter`) is optimized. For more difficult cases, however, where *Dylan* is not a prototypical example, the semantic neighborhoods undergo dramatic rearrangement. For instance, optimizing the triplet (`Bob_Dylan`, `has_profession`, `disc_jockey`) correctly places *Dylan* in the neighborhood of other DJs, despite the implausibility of this association in the neighborhood of his type embedding, which contains no DJs. This places him in the token neighborhood of *Moby*, who is otherwise quite unlike

	Δdensity	t statistic	p
Pos	0.241	$t = 99.7$	$p \ll 10^{-10}$
Neg	-0.059	$t = -62.4$	$p \ll 10^{-10}$

Table 4.2: Change in neighborhood (top-5 closest neighbors) density of true triplets (Δdensity) for positive and negative triplets drawn from the triplet classification dataset introduced by [Socher et al., 2013], which is derived from the FB15k test set and consists of 59,071 positive triplets and the same number of negative triplets. This resulted in $N = 118,142$ queries for both positive and negative examples (two for each triplet, querying both the left and right entity). After computing each triplet's neighborhood, we counted the number of triplet neighbors that were in fact in the training, validation, or test sets of FB15k, yielding a measure of the concentration of true and false examples in the neighborhood of both type and token triplet embeddings.

Bob Dylan except in respect of their common career as DJs.

Combined with our finding that optimization yields the most dramatic improvements in the more stringent evaluation criteria (Hits@1 and Hits@3), this suggests that our optimization procedure is particularly helpful in arbitrating between difficult cases. This qualitative observation about the neighborhoods of compositional and supracompositional triplets can be quantified. Using the triplet classification dataset introduced by [Socher et al., 2013], which contains an equal number of positive and negative triplets, we find (Table 4.2) that positive triplets, on average, end up in neighborhoods with tightly packed positive examples than their compositional counterparts. On the other hands, negative triplets suffer a decrease in the number of positive triplets in the neighborhoods of their optimized embeddings.

To further quantify the role of semantic optimization in inference, we correlated the difference between the Harmony (score) of input triplets pre-and post-optimization with the change in its rank on the FB15k dataset. The change in Harmony is computed as $\Delta\mathcal{H} = \mathcal{H}(\mu(\boldsymbol{x}), \boldsymbol{x}) - \mathcal{H}(\boldsymbol{x}, \boldsymbol{x})$, i.e. the difference between the Harmony of the token em-

bedding and the Harmony of the type embedding. This comparison is model-internal—it does not compare models trained to do token inference with models trained for type inference. However, it serves as a useful index of the performance gains attributable to the optimization procedure. If optimizing a triplet representation indeed improves its relative position among all candidate triplets, we expect changes in Harmony to be negatively correlated with the change in rank of positive triplets. Consistent with this, we find that optimization leads to significant improvements in raw rank in our best trained HHolE model (Spearman's $\rho = -0.0157, p < 10^{-6}$, Figure 4.5). When considering the change in Mean Reciprocal Rank, a more standard evaluation metric, we find that $\Delta\mathcal{H}$ is positively associated with improvements in MRR ($\rho = .1370, p \ll 10^{-10}$),[3] particularly when triplets whose ranks do not change at all are omitted ($\rho = .3746, p \ll 10^{-10}$). In other words, when semantic optimization makes a difference, it does so for the better.

For HDistMult, $\Delta\mathcal{H}$ is significantly associated with increases in the rank of true triplets ($\rho = 0.1226, p \ll 10^{-10}$), a result consistent with our finding that this class of models disprefers low settings of λ. This illustrates the importance of choices of representational format for embeddings of semantic data. Our optimization procedure can only operate over information that is contained in its compositional input. Hence, choices about how to combine the learned features of entities and relations—i.e. about the manner of composition—are central to our framework.

4.5.1 Desiderata of a composition function

What factors affect the success of semantic optimization in combination with a particular composition scheme? We suspect that multiplicative interactions across embedding

[3] ΔMRR is computed as MRR($\mu(\boldsymbol{x})$) − MRR($\boldsymbol{x}$), i.e. the difference between the Mean Reciprocal Rank of the supracompositional and the compositional triplets.

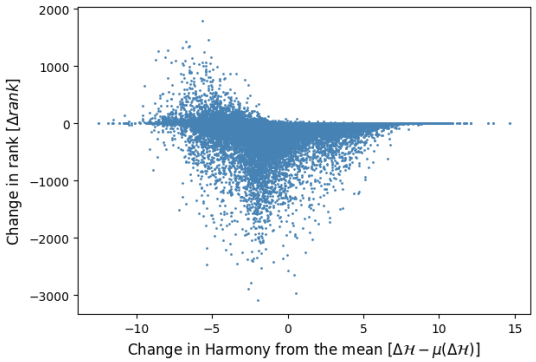


Figure 4.3: Effect of optimization on the rank of FB15K validation set triplets ($N = 100,000$; $50,000$ triplets with two queries per triplet) from the best-performing HHOLE model ($d = 512, \lambda = 1.0$). The horizontal axis is a triplet's change in Harmony pre-and post-optimiation ($\Delta\mathcal{H} \equiv \mathcal{H}(\mu(\boldsymbol{x}), \boldsymbol{x}) - \mathcal{H}(\boldsymbol{x}, \boldsymbol{x})$) minus the mean change in Harmony for all triplets ($\mu(\Delta\mathcal{H})$). This is plotted against $rank_q(\mu(\boldsymbol{x})) - rank_q(\boldsymbol{x})$, the triplet's change in rank due to optimization for query q. A negative correlation indicates reductions in rank (improvements) associated with increasing optimization of triplet representations.

components—which are present in HHOLE and absent in HDISTMULT—are essential for our optimization procedure to contribute helpfully to inference. Both DISTMULT and HOLE are special cases of contracted Tensor Product Representations (TPRs), obtained by summing over (HOLE) or discarding (DISTMULT) terms from the three-way tensor product $\boldsymbol{e}_\ell \otimes \boldsymbol{r} \otimes \boldsymbol{e}_r$.[4] In particular, DISTMULT retains only multiplicative interactions *within* components, omitting terms with non-matching indices. This fact appears to be crucial. In a follow-up experiment, we implemented a series of full TPR models trained on FB15K, using the composition operation (4):

$$\boldsymbol{x}_{\text{HTPR}} = \boldsymbol{e}_\ell \otimes \boldsymbol{r} \otimes \boldsymbol{e}_r \tag{4}$$

Such models are necessarily small in size due to the rapid growth of dimensionality for TPRs as a function of the dimensionality of entity and relation embeddings. Conse-

[4]See [Nickel et al., 2016] for discussion of holographic embeddings as compressed tensor products.

λ	**MR**	**MRR**	**H@1**	**H@3**	**H@10**
∞	150	.278	.192	.305	.447
1.0	**134**	**.295**	**.204**	**.326**	**.471**

Table 4.3: Performance of HTPR models with and without optimization (controlled by λ). For both models, entities were 5-dimensional and relations 20-dimensional. This trend held across other hyperparameter settings.

quently, their performance is also poor in comparison to our other implementations. However, the trend matched that which we observed within the HHolE class: models including the optimization procedure consistently outperformed those with $\lambda = \infty$ (see Table 4.3). From this, we conclude that other embedding-based KBC models incorporating cross-component multiplicative interactions are likely to see improvements from the addition of a semantic optimization step prior to scoring.

4.6 Conclusion

In this chapter, we proposed Gradient Graphs, a general method for augmenting compositional representations of Knowledge Graphs with a post-composition procedure that optimizes the well-formedness of triplet embeddings, highlighting the model's connection to Harmonic Grammar and Optimality Theory. The resulting model shows marked improvements over the compositional models it is implemented alongside, and also produces triplet token embeddings with properties that prove useful for inference about knowledge base entities. In future work, we intend to explore the utility of semantically-optimized token embeddings in other linguistic settings.

US Presidents

George W. Bush			**Barack Obama**		
n	x (compositional)	$\hat{h}$ (optimized)	n	x (compositional)	$\hat{h}$ (optimized)
1	**George H. W. Bush**	**George H. W. Bush**	1	Hillary Rodham Clinton	**George W. Bush**
2	**Bill Clinton**	**Bill Clinton**	2	Al Gore	**Bill Clinton**
3	**Jimmy Carter**	**Jimmy Carter**	3	**George W. Bush**	**John F. Kennedy**
4	**John F. Kennedy**	**Ronald Reagan**	4	**Bill Clinton**	**Ronald Reagan**
5	**Ronald Reagan**	**Barack Obama**	5	**John F. Kennedy**	**George H. W. Bush**

John McCain			Al Gore		
n	x (compositional)	$\hat{h}$ (optimized)	n	x (compositional)	$\hat{h}$ (optimized)
1	John Kerry	John Kerry	1	**Barack Obama**	Condoleezza Rice
2	Hillary Rodham Clinton	Colin Powell	2	**George W. Bush**	John C. Calhoun
3	Colin Powell	Nancy Pelosi	3	Colin Powell	Colin Powell
4	**Richard Nixon**	Joe Biden	4	Condoleezza Rice	Hillary Rodham Clinton
5	**Herbert Hoover**	Dick Cheney	5	**John F. Kennedy**	John Kerry

Table 4.4: Semantic neighborhoods of type (pre-) and token (post-optimization) triplets output by the best-performing HHOLE model ($d = 512, \lambda = 1.0$). Effect of optimization on the semantic neighborhoods of entity embeddings in the context of the query ($\cdot$, `office_title`, `US_President`). For each entity, we retrieved the 5 closest (Euclidean Distance) compositional triplet embeddings, as well as the five closest triplets, among all candidate triplets, when all these candidates are optimized. Triplet completions that in fact occur in FB15K are marked in **bold**. Human-readable entity names were retrieved from a mapping between FREEBASE machine IDs and names of Wikipedia articles built by [Ling and Weld, 2012]. See main text for discussion of the results.

Guises of Bob Dylan

	Singer-Songwriter			**Screenwriter**	
n	x (compositional)	$\hat{h}$ (optimized)	n	x (compositional)	$\hat{h}$ (optimized)
1	**Eric Clapton**	**Bonnie Raitt**	1	**John Lennon**	**John Lennon**
2	**Bonnie Raitt**	**Eric Clapton**	2	Jimi Hendrix	**Barbara Streisand**
3	**Van Morrison**	**Van Morrison**	3	**Barbara Streisand**	**Eric Idle**
4	**B.B. King**	**B.B. King**	4	Eric Clapton	**Nick Cave**
5	**Bob Seger**	**Bob Seger**	5	Eddie Vedder	**Alan Bergman**

	Disc Jockey			**Writer**	
n	x (compositional)	$\hat{h}$ (optimized)	n	x (compositional)	$\hat{h}$ (optimized)
1	Tom Petty	**Steven Van Zandt**	1	**John Lennon**	**Alanis Morissette**
2	Warren Zevon	**Erykah Badu**	2	**Alanis Morissette**	**John Lennon**
3	Willie Nelson	**Alice Cooper**	3	Paul McCartney	**Leonard Cohen**
4	John Mayer	John Mayer	4	Tina Turner	**Leonard Bernstein**
5	Steve Earle	**Moby**	5	Dolly Parton	**Prince**

Table 4.5: *Guises of Bob Dylan*: Effect of optimization on the semantic neighborhood of `Bob_Dylan` in the context of four queries about his `profession`: Bob Dylan as `singer-songwriter`, `screenwriter`, `disc_jockey`, and `writer`. `Bob_Dylan` is a positive instance of each of these professions in FB15K. The procedure for extracting neighborhoods is the same as in Table 4.4.

4.7 Appendix A: Model details

(5) **Claim**: $\mu(x) = -(\mathbb{W} - \lambda I)^{-1}(\frac{1}{2}b + \lambda x)$ is the unique global optimum for $\mathcal{H}(h, x)$ for any fixed x.

We define the Harmony of hidden state h with respect to triplet embedding x as in 4.1:

$$\begin{aligned}\mathcal{H}(h,x) &\equiv \frac{1}{2}\left[h^\top \mathbb{W} h + b^\top h - \lambda (h-x)^\top (h-x)\right] \\ &= \frac{1}{2}\left[h^\top \left(\mathbb{W} - \lambda I\right) h + \left(b + 2\lambda x\right)^\top h - \lambda x^\top x\right] \\ &\equiv \frac{1}{2}\left[h^\top V h + m(x)^\top h - \lambda x^\top x\right]\end{aligned}$$

Completing the square yields:

$$\begin{aligned}\mathcal{H}(h,x) =& \frac{1}{2}\left[\left(h - \frac{-1}{2}V^{-1}m(x)\right)^\top V \left(h - \frac{-1}{2}V^{-1}m(x)\right)\right] \\ &+ \frac{1}{2}\left[-\lambda x^\top x - \frac{1}{4}m(x)^\top V^{-1} m(x)\right] \\ \equiv& \frac{1}{2}\left[\left(h - \mu(x)\right)^\top V \left(h - \mu(x)\right)\right] + \ell(x)\end{aligned}$$

which is valid because $V = \mathbb{W} - \lambda I$ is symmetric. $\ell(x)$ does not depend on h, so it is sufficient to optimize $\frac{1}{2}\left[\left(h - \mu(x)\right)^\top V \left(h - \mu(x)\right)\right]$. Setting $\frac{\partial \mathcal{H}(h,x)}{\partial h} = 0$ yields $2V(h - \mu(x)) = 0$; $\therefore h = \mu(x)$. Since V is negative-definite, this point is a *maximum*.

The truth of the claim may be more quickly perceived by observing that $\mathcal{H}(h, x)$ defines a Gaussian distribution over the hidden state variable h with mean $\mu(x)$ and precision matrix $\Sigma^{-1} \equiv -V$. The optimality of $\mu(x)$ then follows from the unimodality of Gaussians.

The training objective 4.13 may be justified by the following considerations. We take the compositional triplet data to be generated by hidden states of the gradient graph network, and maximize the log probability of the training data using the maximum a posteriori point estimate of the hidden state h. The "complete data" are then $\mathcal{D} = \{\langle \hat{h}, x\rangle\} = \{\langle \mu(x), x\rangle\}$. For fixed x, $\mathcal{H}(h, x)$ models the conditional distribution $p(h|x)$, with

$$(6) \qquad p(h|x) = \frac{\exp\{\mathcal{H}(h,x)\}}{Z(x)}$$

where $Z(x) = \int_{h'} \exp\{\mathcal{H}(h', x)\}\, dh' = |2\pi V^{-1}|^{\frac{1}{2}} \exp\{\ell(x)\}$ is the partition function conditioned on x. Let $\chi_q = \{x'\}$ be the set of candidate triplet embeddings consistent with a given query q. Choosing the discrete distribution $p(x) = \frac{\exp\{\ell(x)\}}{\sum_{x' \in \chi_q} \exp\{\ell(x')\}}$ over triplet embeddings as the prior probability of the embedding x,[5] we have:

$$p(\mu(x), x|q) \propto \frac{\exp\{\mathcal{H}(\mu(x), x)\}}{|2\pi V^{-1}|^{\frac{1}{2}} \sum_{x' \in \chi_q} \exp\{\ell(x')\}}$$

For given parameters, the denominator is constant. So, renormalizing over the discrete triplets χ_q gives:

$$p(\mu(x), x|q) = \frac{\exp\{\mathcal{H}(\mu(x), x)\}}{\sum_{x' \in \chi_q} \exp\{\mathcal{H}(\mu(x), x)\}}$$

Approximating the discrete distribution over all of χ_q with a negative sample yields the objective 4.13.

[5]Note that, when $\mathcal{H}(\cdot, x)$ is evaluated at $\mu(x)$, the only nonzero term in $\mathcal{H}(\mu(x), x)$ is $\ell(x)$. Hence, it is sufficient to perform gradient descent on the prior: $\ell(x)$

4.8 Appendix B: Implementation details

In initial experiments, we searched through a number of candidate models. These included two Harmonic variants of the RESCAL model [Nickel et al., 2011], as well as models that constructed x as a simple concatenation of entity and relation vectors, as well as three-way tensor products of these vectors. These initial experiments led us to focus on DISTMULT and HOLE as the best-performing candidates. Our Harmonic models and reimplementations of the DISTMULT and HOLE baselines were written in TensorFlow [Abadi et al., 2016] and estimated using the Adam optimizer [Kingma and Ba, 2015]. With the exception of the HOLE reimplementation, we uniformly used the log-softmax loss 4.13, which performed best in initial experiments. In contrast, [Yang et al., 2015] use a margin-based ranking loss that is linear in the margin between the scores of positive and negative examples up to a threshold, and [Nickel et al., 2016] use the pairwise linear margin loss applied to the scores squashed by the logistic function. For HOLE, we used the *linear* margin loss, which provided by far the best performance in the experiments. For each model, we trained until performance on the validation set decrease, then chose the best-performing embedding size from among $d \in \{256, 512\}$. Batch size (512), negative sampling rate (500), and learning rate (0.001) were kept constant across models. We note in passing that regions of the hyperparameter space for DISTMULT explored by [Kadlec et al., 2017] were inaccessible to us for technical reasons. For the Harmonic models, we manually tuned the λ hyperparameter.

Chapter 5

Harmonic Memory Networks

Typical embedding-based strategies for knowledge base completion generally rely on learned embeddings of fact elements (entities and relations) into low-dimensional vector spaces. Since representations must be learned from training-set instances of each component, this creates problems when such databases are to be scaled, and therefore these methods have difficulty accommodating an open-world setting in which knowledge graphs evolve in time, since new facts inserted into the database after model training cannot be used for inference without model retraining. Furthermore, databases may be augmented in time not only with new facts about known entities, but also with new entities. In embedding-based models, new representation for such entities must be trained.

In this chapter, we present Harmonic Memories (HMem), a neural network which models entities by aggregating information about their neighborhoods in the knowledge graph using a superposition memory architecture, achieving generalization to new entities without retraining. The network combines two ideas. First, a representation of entities as memory states consisting of superposed vector associations between learned entity and relation embeddings. Second, completion of memory states using a learned transformation

based on Harmony-optimization methods [Smolensky and Legendre, 2006] (and see Chapter 3). We refer to vector associations as *bindings* in the sense of the "variable-binding problem" in the philosophy of cognitive science: in neural net models of cognition, how are representations of the elements of a structure bound together into structures? In this work, we investigate two solutions prominent in the cognitive science literature—tensor product binding [Smolensky, 1990] and circular convolution [Plate, 1994]—which were both effectively applied in knowledge base completion [Nickel et al., 2011, Nickel et al., 2016].

The approach is inspired by computational modeling of biological neural architectures for knowledge representation [Crawford et al., 2015], and is related to KBC methods based on convolution of graph neighborhoods [Schlichtkrull et al., 2017, Dettmers et al., 2017a, Nguyen et al., 2018], in which inference is performed over representations of aggregated entity neighborhoods. Recent work has extended this idea using Graph Attention Networks) [Nathani et al., 2018], which assign attention weights to entries in a graph neighborhood, these being later combined. For instance, [Veličković et al., 2018] use Graph Attention to generate weights for triplet representations obtained by transforming concatenated entity and relation vectors, combining the results by averaging. This is similar to our approach, with the key difference that formulating the model—as we do—in terms of binding allows for clear formal analysis of certain scaling results (§5.6). We therefore gain in interpretability.

HMem scales well in three respects. First, it allows a database with a fixed set of entities and relations to incorporate new facts into the model without parameter re-estimation. Empirically, performance improves in nearly every case when the neighborhoods are thus expanded. Second, it permits the addition of entities unseen in training, whose representations are undefined in a vector embedding framework. For our model,

inferences about these entities are possible when a subgraph including them becomes available. Third, our model effectively handles nodes with high in-degree. We show that, whereas existing embedding-based approaches show decreased performance with highly connected nodes, our model exhibits improved performance on nodes with many neighbors.

§5.1, §5.2 and §5.3 introduce the Harmonic Memory architecture, and §5.4 shows that our model achieves state-of-the-art results on benchmark KBC datasets. After evaluation on standard benchmarks, §5.5 introduces WNGen and FBGen, new datasets we have developed based on WordNet and Freebase that evaluate the network's ability to abstract from node identity and make inferences exclusively on the basis of information about nodes in its neighborhood, and §5.6 examines in detail how the model scales with the size of entity neighborhoods and the addition of new input facts. §5.8 concludes.

5.1 Representation of memory states

To review the task: a knowledge graph consists of triplets composed of a pair of entities and a relation, e.g. $(\boldsymbol{e}_{\text{cat}}, \boldsymbol{r}_{\text{has\_part}}, \boldsymbol{e}_{\text{paw}})$. Here, we say that cat is a left-neighbor of `paw` with respect to the relation `has_part`. In graph completion, we are given a query in the form of either $(\cdot, \boldsymbol{r}, \boldsymbol{e}_r)$ or $(\boldsymbol{e}_\ell, \boldsymbol{r}, \cdot)$, representing queries of the left and right entity respectively.

We denote sets of discrete symbols with calligraphic fonts (e.g. $\mathcal{V}$), corresponding vector spaces with italics (e.g. V), individual symbols from $\mathcal{V}$ with bold letters, corresponding vectors in V in bolded italics (e.g. $\mathtt{v}_i$ is a symbol and $\boldsymbol{v}_i$ its vector embedding) and memory states with $\mathtt{M}$. $\mathcal{E}$ and $\mathcal{R}$ denote sets of entities and relations respectively. We embed each entity and relation symbol $\mathtt{e}_i \in \mathcal{E}, \mathtt{r}_i \in \mathcal{R}$ in a d_e, d_r-dimensional space,

yielding vectors $\boldsymbol{e}_i \in E$ and $\boldsymbol{r}_i \in R$ which are used to construct graph memories $\mathtt{M}_i$ for each entity by aggregating triplet entries (see Section 5.2) into a representation of the entity's immediate neighborhood. For example, the memory state for $\mathtt{e}_{canine}$ would include bindings of `hypernym` to `dog`, `has_part` to `paw`, `hyponym` to `mammal`, etc.

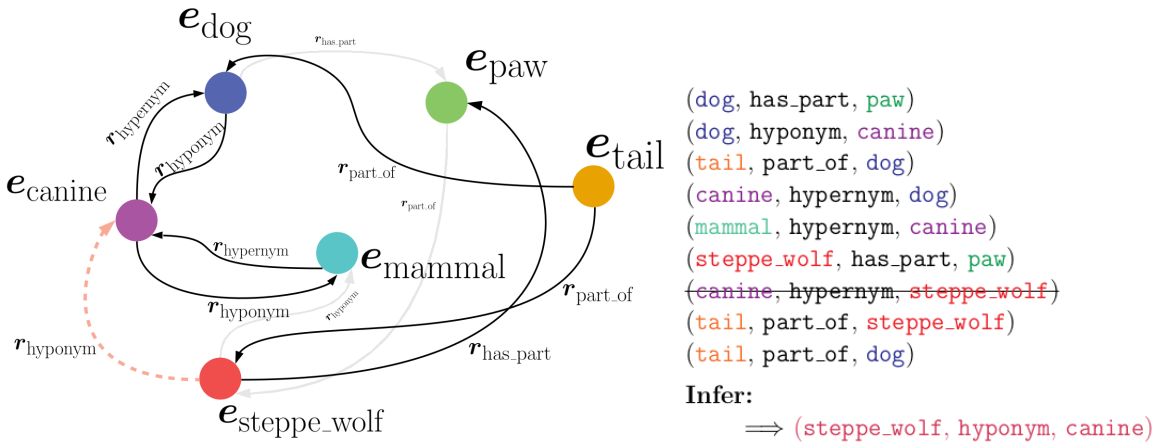


Figure 5.1: **Left**. Neighborhood subgraph for an entity $\mathtt{e}_{\text{canine}}$. In this example, the unweighted memory state $\mathtt{M}_{\text{canine}}$ in the query $(\cdot, \mathtt{r}_{\text{hyponym}}, \mathtt{e}_{\text{canine}})$ with true completion $\mathtt{e}_{\text{steppe\_wolf}}$ is $\mathbb{B}(\boldsymbol{r}^r_{\text{hyper}}, \boldsymbol{e}_{\text{dog}})) + \mathbb{B}(\boldsymbol{r}^\ell_{\text{hypo}}, \boldsymbol{e}_{\text{dog}}) + \mathbb{B}(\boldsymbol{r}^r_{\text{has\_part}}, \boldsymbol{e}_{\text{paw}}) + \mathbb{B}(\boldsymbol{r}^\ell_{\text{hypo}}, \boldsymbol{e}_{\text{mammal}}) + \mathbb{B}(\boldsymbol{r}^r_{\text{hyper}}, \boldsymbol{e}_{\text{mammal}}) + \mathbb{B}(\boldsymbol{r}^r_{\text{has\_part}}$. **Right**. The inference from this neighborhood depicted. If the premise (`canine`, `hypernym`, `steppe_wolf`) is included, then the inference goes through thanks to the inversion of `hypernym`—a formal property. When omitted, the inference is more inductive ("dogs are canines, pugs are canines, labradores are canines, and all of them have paws, tails, and teeth. So do steppe wolves, which are therefore likely canines"). Then, it is useful to enumerate examples of `canines`, with implicit links to their properties (`paws`, `tails`...) encoded in their embeddings in order to draw the inference. `canine`'s derivational relationship to the constellation `canis_minor` is less important in this connection. Drawing such distinctions is the role of the memory weighting module.

Each model in the HMEM class is parametrized by a binding map $\mathbb{B}$ that associates entity and relation entries, and a corresponding unbinding map $\mathbb{U}$. Unbinding approximately inverts binding operation by addressing the memory using a query relation vector r_q to retrieve entity vectors likely to be bound with that relation in the memory. The binding and unbinding maps are chosen such that, for an entity whose neighborhood is

the singleton set $\{(\boldsymbol{e}_i, \boldsymbol{r}_j)\}$:

$$\mathbb{U}\left(\boldsymbol{r}_j, \mathbb{B}\left(\{(\boldsymbol{e}_i, \boldsymbol{r}_j)\}\right)\right) \approx \boldsymbol{e}_i$$

i.e., e_i can be approximately retrieved from the binding of e_i to r_j by addressing the memory $\mathtt{M} = \mathbb{B}\left(\{(\boldsymbol{r}_j, \boldsymbol{e}_i)\}\right)$.

Associations between entity and relation vectors in our model are pairwise. To account for the directionality of relations, we train two embeddings for each relation—one associated with an entity's left-neighbors and another for its right-neighbors. Each relation thus has a left-embedding $\boldsymbol{r}^{\ell}$ and a right-embedding $\boldsymbol{r}^{r}$, with, for instance, entries $(\boldsymbol{r}^{r}_{\text{has\_part}}, \boldsymbol{e}_{\text{claw}})$ in the memory state for `canine` and an entry $(\boldsymbol{r}^{\ell}_{\text{has\_part}}, \boldsymbol{e}_{\text{canine}})$ in the memory state for `paw` (see Fig. 5.1).

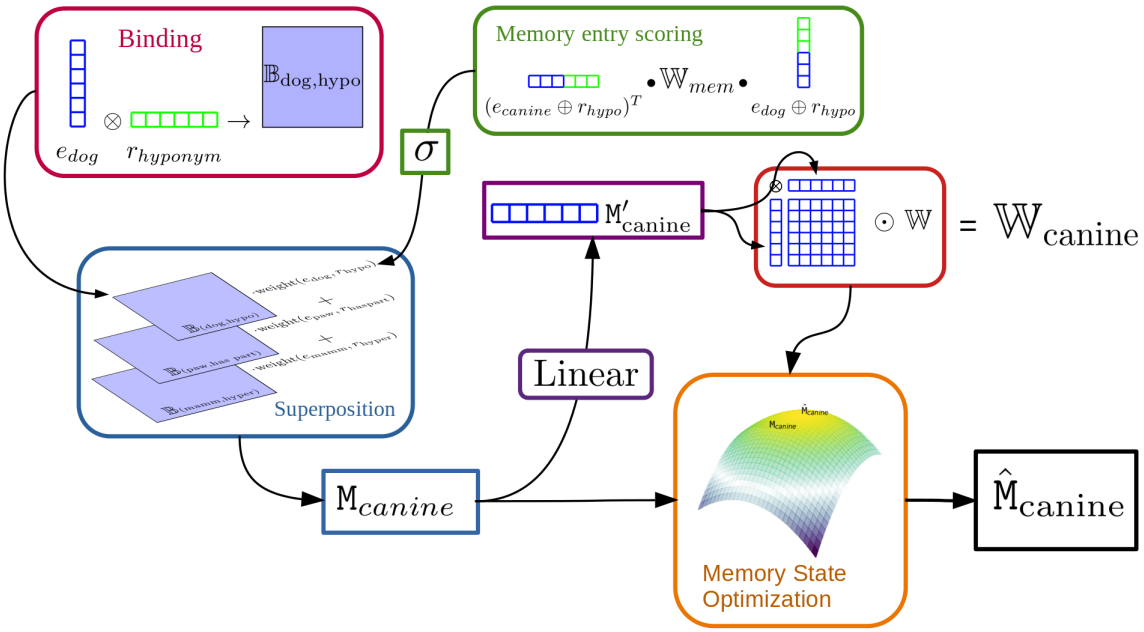


Figure 5.2: Harmonic Memory architecture with TPR binding.

5.2 Binding

Harmlessly overloading the symbol $\mathbb{B}$ to apply to both tuples and sets of tuples, we consider binding operations that aggregate neighborhoods via summation of the individual entries, i.e.

$$\mathbb{B}\left(\{(\boldsymbol{r}_i, \boldsymbol{e}_i)\}\right) = \sum_i \mathbb{B}\left(\boldsymbol{r}_i, \boldsymbol{e}_i\right)$$

where $\mathbb{B}\left(r_i, e_i\right)$ is the binding of the ith memory entry. As binding operations, we evaluate the tensor product (5.1) [Smolensky, 1990] and circular convolution (5.2).

$$\mathbb{B}^{\text{TPR}}(\boldsymbol{r}, \boldsymbol{e}) = \boldsymbol{r} \otimes \boldsymbol{e} \tag{5.1}$$

$$\mathbb{B}^{\text{CConv}}(\boldsymbol{r}, \boldsymbol{e}) = \boldsymbol{r} \circledast \boldsymbol{e} \tag{5.2}$$

A tensor product is unbound by left-dotting the memory state with a relation vector r:

$$\mathbb{U}^{\text{TPR}}\left(\boldsymbol{r}, \mathtt{M}\right) = \boldsymbol{r} \cdot \mathtt{M} \tag{5.3}$$

When the relation vectors are normalized, $\mathbb{U}^{\text{TPR}}$ exactly recovers e from the singleton memory state $\mathtt{M} = r \otimes e$. For CConv, the unbinding operation is

$$\mathbb{U}^{\text{CConv}}\left(\boldsymbol{r}, \mathtt{M}\right) = \boldsymbol{r} \star \mathtt{M} \tag{5.4}$$

$\star$ denotes circular correlation, which is computed efficiently using the Fourier transform $r \star \mathtt{M} = \mathcal{F}^{-1}\left(\overline{\mathcal{F}(\boldsymbol{r})} \odot \mathcal{F}(\boldsymbol{e}\right)$, $\overline{\boldsymbol{x}}$ denoting the complex conjugate of $\boldsymbol{x}$.

As previewed in Chapter 2, circular convolution encoding with correlation decoding

was introduced to connectionist modeling by [Plate, 1994], and applied to KBC by [Nickel et al., 2016] and our work [Lalisse and Smolensky, 2019], exploiting the fact that the correlation is an approximate inverse of convolution ($\boldsymbol{x} \star (\boldsymbol{x} \circledast \boldsymbol{y}) \approx \boldsymbol{y}$), under additional stipulations discussed in Appendix 1. There, we describe a transformation on embeddings in the CConv model that guarantee this property, and improved performance on the CConv models.

Memory weighting. For entities with large neighborhoods, it is intractable to compute bindings for all neighbors.[1] So, prior to superposition, we filter candidate bindings and commit them to memory in graded form. Entity and relation pairs are scored with respect to $\boldsymbol{e}_i$ and the query relation $\boldsymbol{r}_q$ according to Eqn. (5.5), where $\oplus$ denotes vector concatenation. W_{weight} and $\boldsymbol{b}_{\text{weight}}$ are learned weight matrices and bias vectors indexed to $\boldsymbol{r}_q$.

$$\text{weight}(\boldsymbol{e}_c, \boldsymbol{r}_c | \boldsymbol{e}_i, \boldsymbol{r}_q) = \sigma(\, (\boldsymbol{e}_i \oplus \boldsymbol{r}_q)^\top W_{\text{score}} \, (\boldsymbol{e}_c \oplus \boldsymbol{r}_c) + \boldsymbol{r}_{score}^{q\,\top} \, (\boldsymbol{e}_c \oplus \boldsymbol{r}_c) \,) \tag{5.5}$$

After scoring, the top $k = 200$ candidate neighbors are bound and entered into memory, weighted by their scores:[2]

$$\mathtt{M}_i = \sum_c \text{weight}(\boldsymbol{e}_c, \boldsymbol{r}_c | \boldsymbol{e}_i, \boldsymbol{r}_q) \mathbb{B} \, (\boldsymbol{r}_c, \boldsymbol{e}_c) \tag{5.6}$$

Remark. Our binding-based approach is inspired by [Crawford et al., 2015], who developed a biologically realistic neural network for representing WordNet. In their

[1] The largest entity neighborhood in WordNet contains 961 links (mean=7), and 9739 (mean=65) for Freebase.

[2] While the memory state $\mathtt{M}_i$ for a given query also depends on the query relation $\boldsymbol{r}_q$, this additional subscript is omitted for convenience.

model, a memory state vector for each entity is formed by summing pairwise associations (convolution) of entities and relations, one association per graph link. Links can be recovered by unbinding stored associations from entity memory states to recover a node's immediate neighbors. Since Crawford et. al. are mainly preoccupied with neural realism rather than learning or generalization, the embeddings for each graph element are randomly sampled rather than trained, and the model is evaluated on an embedding of the full WORDNET database in a simple artificial task (graph traversal). Thus, their work only investigates the model's ability to robustly retrieve the vectorized knowledge graph. Since our target task requires generalization from a partial graph, we introduce additional operations that complete the representation of each entity.

5.3 Memory completion

As with the Gradient Graphs model (4), the assembled memory state is forwarded to a memory completion operation based on optimization of the Harmony Equation (5.7)—which is parametrized by a learned symmetric weight matrix $\mathbb{W}$ and bias vector $\boldsymbol{b}$—with respect to the vector $\boldsymbol{m}$.

$$\begin{aligned}\mathcal{H}_{\mathbb{W},\boldsymbol{b}}(\mathtt{M}_i, \boldsymbol{m}) =& \frac{1}{2}\left(\boldsymbol{m}^\top \mathbb{W}\boldsymbol{m} + \boldsymbol{b}^\top\right) \\ & - \frac{\lambda}{2}\left(\mathtt{M}_i - \boldsymbol{m}\right)^\top \left(\mathtt{M}_i - \boldsymbol{m}\right)\end{aligned} \tag{5.7}$$

This is solved by

$$\begin{aligned}\hat{\mathtt{M}}_i &= \text{argmax}_{\boldsymbol{m}} \mathcal{H}_{\mathbb{W},\boldsymbol{b}}(\mathtt{M}_i, \boldsymbol{m}) \\ &= (\mathbb{W}_i - \lambda I)^{-1}(2\lambda \mathtt{M}_i + \boldsymbol{b})\end{aligned} \tag{5.8}$$

when λ—a hyperparameter—is greater than the spectral norm of $\mathbb{W}$, guaranteeing the existence of a unique optimum for $\mathcal{H}_{\mathbb{W},b}$. This formulation is motivated by associative memory models like the formally similar Hopfield networks [Hopfield, 1982], which complete corrupted input patterns by minimizing the Energy of the resulting network configuration. λ controls the magnitude of the penalty for the squared distance between the output and the input memory $\mathtt{M}_i$—encoding the constraint that the output cannot deviate too far from the initial input (see Chapter 4)—with a value of $\lambda = \infty$ implying that $\mathcal{H}_{\mathbb{W},\boldsymbol{b}}$ is maximized at $\mathtt{M}_i$ (the output memory is the initial superposition of pairwise bindings).

We allow the parameters of the Harmony function to change with the query being posed by specifying a weight matrix $\mathbb{W}_i$ computed for any given $\mathtt{M}_i$. The local weight matrix $\mathbb{W}_i$ is computed from a global weight matrix $\mathbb{W}_{\text{global}}$ and a filter vector $\boldsymbol{M}'_i$, which is a function of the input memory:

$$\mathbb{W}_i = \boldsymbol{M}'_i \boldsymbol{M}'^{\top}_i \odot \mathbb{W}_{\text{global}} \tag{5.9}$$

where

$$\boldsymbol{M}'_i = W_{map}\, \mathtt{M}_i + \boldsymbol{b}_{\text{map}} \tag{5.10}$$

W_{map} and $\boldsymbol{b}_{\text{map}}$ are a learned matrix and bias vector mapping each memory state to a filter vector, whose self-outer product multiplies the global weight matrix elementwise. The resulting weight matrices vary smoothly with the value of the input memory state, leading to distinct hypersurfaces in the $|\mathtt{M}|$-dimensional space of memory states (e.g. Figure 5.3.) The specific weight-modulation function that is used is a hyperparameter of the overall

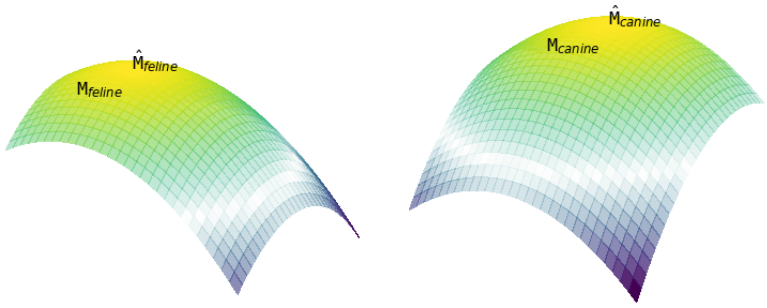


Figure 5.3: Harmony functions on a two-dimensional memory state space (visualized as surfaces) and optima defined by weight matrices $\mathbb{W}_{\text{feline}}, \mathbb{W}_{\text{canine}}$ calculated from distinct memory states $\mathtt{M}_{\text{feline}}, \mathtt{M}_{\text{canine}}$.

framework. We selected this function because of its simplicity (bilinear in the memory $\mathtt{M}_i$) and the fact that it preserves symmetry of the output weight matrix (since $\boldsymbol{M}'_i \boldsymbol{M}'^{\top}_i$ is symmetric).

In inference, we first optimize $\mathtt{M}_i$ to generate $\text{opt}(\mathtt{M}_i) = \hat{\mathtt{M}}_i$. The optimized memory state is then probed using the unbinding map $\mathbb{U}(\boldsymbol{r}_q, \text{opt}(\mathtt{M}_i))$, returning a vector $\boldsymbol{e}_o$ representing the output of a probe of the memory $\mathtt{M}_i$ of $\boldsymbol{e}_i$ for entities that are related to $\boldsymbol{e}_i$ via $\boldsymbol{r}_q$. The output of unbinding is then compared with the vectors e_c for all candidate completions $\boldsymbol{e}_c$ using the negative squared Euclidean distance.

$$\text{score}(\boldsymbol{e}_c) = -\left\|\boldsymbol{e}_o - \boldsymbol{e}_c\right\|^2$$

Training. During training, the link for the current training instance is withheld from the neighborhood for the corresponding entity, with the input memory state constructed from the remaining neighbors. This incomplete memory state is then optimized with respect to Eqn. (5.7), and the result probed for a predicted completion. We use the cross entropy training loss derived from the squared Euclidean distance of the output from the true completion, relative to a negative sample $\mathcal{N} = \{\boldsymbol{e}_n\}$ of alternative completions. For

instance, the loss for a right-probe of $(\boldsymbol{e}_i, \boldsymbol{r}, \cdot)$ with true entity $\boldsymbol{e}_j$ is

$$\mathcal{L}(\boldsymbol{e}_j|\boldsymbol{e}_i, \boldsymbol{r}) = -\log \frac{\exp\{-\left\|\boldsymbol{e}_o - \boldsymbol{e}_j\right\|^2\}}{\sum_{\boldsymbol{e}_n \in \mathcal{N}} \exp\{-\left\|\boldsymbol{e}_o - \boldsymbol{e}_n\right\|^2\}}$$

5.4 Results

Models were evaluated using the benchmark datasets WN18 (a subset of WordNet), FB15k (subset of Freebase), and the challenge dataset WN18RR, which removes reciprocal relation pairs from the training and test set of WN18, which can be solved by adopting a simple rule-based system [Dettmers et al., 2017a]. We varied the binding method {CConv, TPR}, the value of optimization constant λ {∞, 1, 2}, and entity and relation embedding sizes. To illustrate the effect of each model component, we report results for both binding methods and best results from finite and nonfinite values of λ. If $\lambda = \infty$, the optimization step is the identity map, in which case the inference objective is to express the target binding as a linear combination of input bindings via memory weighting.

We report the standard evaluation metrics for the Link Prediction task, in which the model is queried on both the left and right sides, ranking candidates for each query. For instance, in the left-query $(\cdot, \boldsymbol{r}, \boldsymbol{e}_j)$ with true completion $(\boldsymbol{e}_T, \boldsymbol{r}, \boldsymbol{e}_j)$, each candidate entity $\boldsymbol{e}_c$ is scored as $\text{score}_\ell(\boldsymbol{e}_c|\boldsymbol{e}_j, \boldsymbol{r})$. The results are ranked and tabulated with the Mean Rank (MR), Mean Reciprocal Rank (MRR), and Hits@N metrics. In each case, all attested links (those found in the training, validation and test sets) were first filtered from the list of candidates. In Appendix 2, we also report the results from ablating conditioning of the weight matrix $\mathbb{W}_i$ on $\mathtt{M}_i$, instead using the global weight matrix $\mathbb{W}_{\text{global}}$.

	WordNet					Freebase				
Model	MR	MRR	H@1	H@3	H@10	MR	MRR	H@1	H@3	H@10
DistMult [Yang et al., 2015]†	457	.790	-	-	.950	**36**	.837	-	-	**.904**
ComplEx [Troullion et al., 2016]	-	.941	.936	.945	.947	-	.692	.599	.759	.840
R-GCN+ [Schlichtkrull et al., 2017]	-	.819	.697	.929	**.964**	-	.696	.601	.760	.842
ConvE [Dettmers et al., 2017a]	374	.943	.935	.946	.956	51	.657	.558	.723	.831
SimplE [Kazemi and Poole, 2018]	-	.942	.939	.944	.947	-	.727	.660	.773	.838
HypER [Balazevic et al., 2019a]	**431**	**.951**	**.947**	**.955**	.958	44	**.790**	**.734**	**.829**	.885
TorusE [Ebisu and Ichise, 2018]	-	.947	.943	.950	.954	-	.733	.674	.771	.832
HMem-CConv	262	.927	.913	.939	.946	**24**	.664	.548	.749	.867
HMem-CConv+	227	.933	.919	**.945**	**.952**	**24**	.664	.547	.749	.866
HMem-CConv$_\infty$	308	.884	.851	.912	.934	39	.488	.363	.554	.734
HMem-CConv$_\infty$+	**183**	.899	.866	.930	.951	39	.481	.357	.546	.725
HMem-CConv$_{\text{im}}$	344	**.936**	**.929**	.942	.947	25	**.728**	**.637**	**.795**	**.881**
HMem-TPR	253	.934	.923	.944	.948	30	.590	.478	.660	788
HMem-TPR+	**174**	**.944**	**.932**	**.955**	**.960**	29	.592	.479	.662	.791
HMem-TPR$_\infty$	395	.874	.823	.922	.939	38	.612	.517	.669	.782
HMem-TPR$_\infty$+	323	.879	.24	.930	.950	37	.616	.521	.674	.786
HMem-TPR$_{\text{im}}$	245	.936	.924	.947	.952	**24**	**.790**	**.731**	**.831**	**.886**

Table 5.1: **Results on WordNet and Freebase benchmarks**. Hyperparameters were tuned on the validation set for each model class M, with results recorded for both infinite and the best finite value of λ. Each model was evaluated on the test set using the same graph as in training, and also (M+) when extending the inference graph with all triplets from the validation set, without performing additional gradient descent on the validation triplets. Implicit binding models (M_{im}) forego explicit binding in favor of memory states that are directly learned for each entity. MR: Mean rank. MRR: Mean reciprocal rank—mean(1/rank). Hits@N: Proportion of test trials in which the rank of the test entity was less than or equal to N. †: results from [Kadlec et al., 2017], who optimized hyperparameter settings for DistMult.

Model	MR	MRR	H@1	H@3	H@10
ComplEx†	5261	.44	.41	.46	.51
ConvE	5277	.46	.39	.43	.48
ConvKB	2554	.248	-	-	.525
HypER	5798	.465	.436	.477	.522
CCONV+	**4609**	**.408**	.373	**.427**	**.471**
CCONV+	7553	.387	.347	.414	.453
CCONV_{im}	4775	.401	**.381**	.408	.437
TPR+	**2223**	**.432**	.384	**.458**	**.514**
TPR+	3662	.397	.350	.432	.469
TPR_{im}	3595	.424	**.393**	.440	.479

Table 5.2: **Results on WN18RR**. We compare with all models from Table 5.1 where authors reported results on WN18RR. †: results from [Dettmers et al., 2017a]. The **first**- and **second**-best results are bolded/underline and bolded respectively.

A virtue of our model is that it can be freely augmented with additional graph triplets after training; hence, we also report results when including validation triplets in the graph used for inference (Model+). This introduces no bias in model selection, which is performed just on the training data. We also explicitly compare neighborhood aggregation to an embedding-based approach, the *implicit binding* models (Model_{im}), in which memory states for each entity are learned directly as embeddings rather than being assembled from the entity neighborhood (*explicit binding*). These remain binding models since they are treated identically to the explicit binding memories with respect to unbinding. For instance, the TPR_{im} model predicts links by unbinding a predicted entity from the optimized memory state $\text{opt}(\mathtt{M}_i)$, where $\mathtt{M}_i$ is now a $m_E \times m_E$ tensor that is learned for each entity.

The Harmonic Memory models are state of the art for WN18 and FB15K (Table 5.1), achieving especially noteworthy improvements in the Mean Rank metric. TPR binding outperforms circular convolution in every setting with quite low-dimensional embeddings

(at most 80d for entities and 25d for relations). It is also competitive with recent models on WN18RR. As well, extending the graph with additional triplets after training yields improvements in all but one case (HMem-TPR on Freebase). The inclusion of the memory-completion module substantially improves performance on WordNet relative to M_∞ on the more stringent evaluation metrics—leading for instance to an 8-point improvement in Hits@1 for HMem-TPR. The implicit binding models substantially outperform explicit binding on Freebase, a fact that is only true on aggregate, with important distinctions arising when entities with different neighborhood sizes are considered. We explain this further in Section 5.6.

5.5 Generalizing to new entities

To evaluate HMem's ability to generalize exclusively on the basis of aggregated neighborhoods, we introduce a new KBE task in which models make inferences about entities not seen in the training set. Consider the following scenario: a model is trained to complete a given knowledge base, but the knowledge base can be augmented in time not just with new facts about the current set of entities, but also with new entities. It would be desirable to perform inference over these new entities, without re-training the model, once partial information about these entities becomes available. Embedding-based models generally require that a representation for each entity be learned in the course of training. Hence, entities not encountered in the training set cannot be modelled. This creates a scalability problem: the knowledge base cannot be augmented with new entities without additional rounds of gradient descent. In contrast, our networks model entities by aggregating their links with other entities in the training graph, allowing entities not seen in training (and hence without a learned embedding) to be represented as memory

states once information about these entities' neighborhoods becomes available. Of course, because our model separates the entity embeddings used to construct memories and the "embeddings" (memories) derived from memory construction itself, this means we can only construct representations for the unseen entities using the ones that *were* seen in training. This flaw, however, is shared with any model that uses trained embeddings as an input, as opposed to "embeddings" whose features are derived from rule-based feature extraction or static word embeddings.

We built two datasets for knowledge base embedding with generalization (KBEGen) using WordNet (WN18) and Freebase (FB15k). First, a random selection of entities in each database (1500 for WN18, 1000 for FB15k) were randomly held-out, and all triplets not containing these entities were assigned to the training set. The number of entities held out was manually chosen to yield approximately the same training data size as the original datasets (WN18 = 141k, FB15k = 483k). Of the remaining triplets, we removed any for which both entities were part of the held-out set, and further split the remaining data into an *observed subgraph* (2/3), a validation set (1/6), and a testing set (1/6). The observed subgraph was used to construct neighborhoods for each of the held-out entities, which were not trained with any further rounds of gradient descent and did not have trained entity embeddings.

Evaluation We fit the model using the training set, with the validation set used for model selection (early-stopping and hyperparameter selection, varying embedding dimension and λ). In evaluation, the observed subgraph was used to construct a memory state for each entity in the held-out set using summation of entity-relation bindings in the observed subgraph for the held-out entity. For each test triplet, the memory state was probed using the query relation to rank the held-in entities as candidate neighbors for the modelled entity.

	heldout	train	valid	test	obs
WNGen	1.5K	141K	1.7K	1.7K	6.8K
FBGen	1K	496K	15K	15K	62K

Table 5.3: Size of the KBC generalization dataset partitions WNGen and FBGen. *heldout*: # of held-out entities; *obs*(erved): # of triplets containing held-out entities that were retained for constructing the inference graph. The number of entities held out for each dataset was manually chosen to yield approximately the same training data size as the original datasets (WN18 = 141,442, FB15k = 483,142).

Results. Performance on generating memory states for unseen entities (Table 5.4) is far from ceiling but well above chance, with a more than 50% accuracy (Hits@1) for the best-performing model on Wordnet.[3] Notably, performance on Wordnet improves dramatically from the addition of the validation subgraph during inference, leading to a nearly 10-point increase in accuracy for the best-performing model (CConv+). Improvements are smaller but reliable for FBGen.

5.6 Scaling properties

As illustrated in Fig. 5.4, superposition memories are prone to increased decoding errors as the number of stored vectors increases. This is due to two factors: overlap between relation vectors even when these are linearly independent, and many-to-one nature of relation-entity bindings. Our model balances two competing priorities: (1) including as much information as is relevant for inference; (2) reducing the number of stored entity-relation bindings, which tend to interfere with each other during retrieval.

The memory weighting module shoulders the burden of priority number (2), and is effective for this purpose (5.5). We compared the embedding-based models with weighted explicit binding by considering performance as a function of the size of an entity's neigh-

[3]Performance with random initialization on Wordnet is less than 1%.

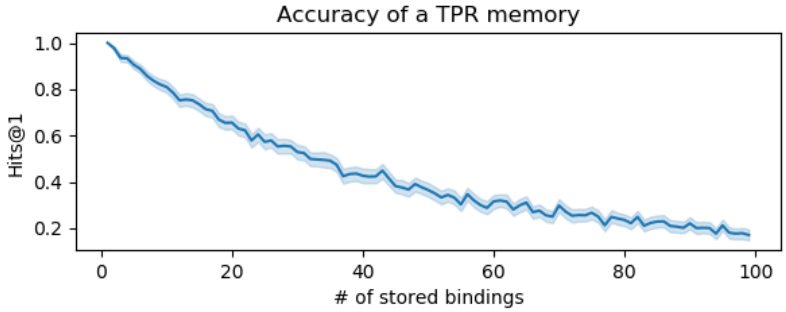


Figure 5.4: **TPR memory**. Performance of a simulated TPR memory in retrieval of stored bindings of 100 100d "entity" vectors bound to twenty 20d "relation" vectors. For each simulation, we summed n (x-axis) bindings of randomly sampled relations to randomly sampled entities. Decoding from the TPR degrades as the number of entity-relation bindings increases (though see [Haley and Smolensky, 2020] for a more optimistic assessment of the decoding error from overloaded TPRs).

borhood (node degree). Explicit binding outperforms direct embedding in WORDNET, though both methods are unaffected by neighborhood size. Large differences appear in FREEBASE, which has a much higher average node degree. Performance in the implicit model is highest in smaller neighborhoods and declines with node degree. With explicit memory construction, however, model performance is higher on nodes with large neighborhoods, peaking at an MRR of .9 for nodes with more than 500 neighbors.

This appears counterintuitive given that higher-degree nodes have more training instances, which might yield higher-quality embeddings in the implicit models. We can explain this result. Consider the simplified scenario of a TPR memory trained to minimize the retrieval error for fixed entity/relation vectors with respect to the embedding $\mathtt{M}_{\text{cat}}$, we have:

$$\begin{aligned}\mathcal{L} &= \mathbb{E}\left[\|\boldsymbol{e}_j - \boldsymbol{r}_i\|^2\right] \\ &= \sum_{i,j} p(\boldsymbol{r}_i, \boldsymbol{e}_j | \boldsymbol{e}_{\text{cat}}) \, \|\boldsymbol{e}_j - \boldsymbol{r}_i \cdot \mathtt{M}_{\text{cat}}\|^2\end{aligned}$$

$\mathtt{M}_{\text{cat}}$ is optimized[4] by

$$\mathtt{M}_{\text{cat}} = \sum_{i,j} p(\boldsymbol{r}_i, \boldsymbol{e}_j | \boldsymbol{e}_{\text{cat}})\ \boldsymbol{r}_i \otimes \boldsymbol{e}_j$$

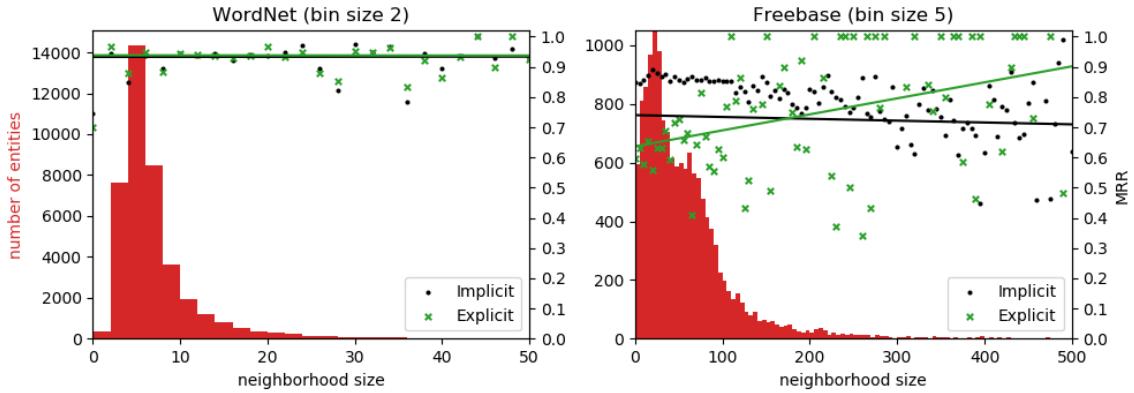
i.e. an expectation-weighted superposition of pairwise vector outer products. In the KBC setting, the distribution $p(\boldsymbol{r}_i, \boldsymbol{e}_j | \boldsymbol{e}_{\text{cat}})$ is uniform over all nonzero entity-relation pairs in the training set, meaning that entities with more neighbors have more nonzero terms in the solution (cf. fig. 5.4). The learned embedding is thus susceptible to increasing decoding error with increasing neighborhood size.

This result extends quite generally to a large class of models —such as Rescal [Nickel et al., 2011] and HolE [Nickel et al., 2016]—that we can formulate as binding models. For instance, the bilinear scoring function Rescal evaluates triplets by dotting left and right entity vectors with a relation-specific bilinear form $W_r \in \mathbb{R}^{d_e \times d_e}$: $\text{score}(\boldsymbol{e}_i, \boldsymbol{r}, \boldsymbol{e}_j) = \boldsymbol{e}_i{}^\top W_r \boldsymbol{e}_j$. For given entity embeddings and where the norm of W_r is capped at some maximum value, the optimal relation embedding[5] is a multiple of $\hat{W}_r = \frac{1}{n}\sum_i^n \boldsymbol{e}_{\ell,i}\boldsymbol{e}_{r,i}^\top = \frac{1}{n}\sum_i^n \boldsymbol{e}_{\ell,i} \otimes \boldsymbol{e}_{r,i}$ for all of the n attested $\langle \boldsymbol{e}_{\ell,i}, \boldsymbol{e}_{r,i} \rangle$ edges involving r. This is a superposition of entity pairs bound by the tensor product where, to evaluate candidate links for the query $(\boldsymbol{e}_i, \boldsymbol{r}, \cdot)$, we first retrieve prototypical $\boldsymbol{r}$-neighbor for $\boldsymbol{e}_i$—$\boldsymbol{e}_i \cdot W_r$, which is the weighted sum of all of the entities $\boldsymbol{e}_j$ that $\boldsymbol{e}_i$ occurred with—and then compare each candidate with this prototype. The greater the number of attested neighbors $\boldsymbol{e}_j$, the higher the anticipated retrieval error.

The improvements from comparing explicit versus implicit binding can be attributed to the weighting module's judicious choice of information to include in a particular query.

[4]Up to a scaling factor, provided the components of the relation vectors r_i are uncorrelated (i.e. the second moment $\mathbb{E}[\boldsymbol{r}_i \boldsymbol{r}_i{}^\top] \propto I$). This can be guaranteed by transformation (compare Appendix 1).

[5]Because the expected score is $\mathbb{E}\left[\text{vec}(\boldsymbol{e}_i \otimes \boldsymbol{e}_j)\text{vec}(W_r)^\top\right] = \mathbb{E}\left[\text{vec}(\boldsymbol{e}_i \otimes \boldsymbol{e}_j)\right]\text{vec}(W_r)^\top$ which is maximized when $\text{vec}(W_r)$ is collinear with $\text{vec}(\boldsymbol{e}_i \otimes \boldsymbol{e}_j)$.



Model	100	200	300	400	500	600
Implicit	.862	.816	.793	.702	.741	.617
Explicit	.632	.746	.772	.856	.835	.900

Figure 5.5: Effect of entity neighborhood size on task MRR for the best implicit and explicit binding models in WordNet and Freebase, with neighborhood sizes binned at increments of 2 and 5 respectively. The line of best fit for neighborhood size against MRR is also plotted. **Table**: MRR for implicit and explicit binding models trained on FB15k averaging over neighborhoods of different sizes (0 to 99 neighbors, 100-199 neighbors, etc.).

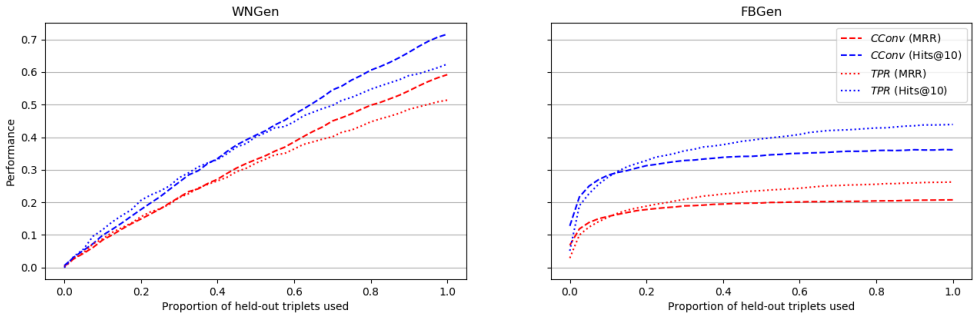


Figure 5.6: Performance on WNGen (left) and FBGen (right) test sets as a function of the proportion of observed graph triplets included in the inference graph. Each fraction of the full dataset (observed+valid) was used to construct the inference graph, the model then being evaluated on the test set. The model was not trained on any triplets from the observed subgraph or validation set.

	Model	MR	MRR	H@1	H@3	H@10
WNGen	CConv	2286	.487	.426	.527	.594
	CConv+	**1359**	**.592**	**.518**	**.647**	**.716**
	TPR_∞	2127	.435	.373	.476	.540
	TPR_∞+	1507	.514	.448	.565	.624
FBGen	$CConv_\infty$	378	.205	.130	.225	.358
	$CConv_\infty$+	373	.207	.131	.251	.361
	TPR_∞	401	.252	**.173**	**.299**	**.439**
	TPR_∞+	**397**	**.263**	**.173**	**.299**	**.439**

Table 5.4: Results on the KBEGen task.

Notably, our results differ from those of [Schlichtkrull et al., 2017], who found performance decreases with increasing node degree in a graph convolution-based model, indicating that our approach is promising to pursue in graphs with high mean node degree.

HMem with explicit binding also scales well with the open-world setting of evolving knowledge graphs. We note again that performance almost always increases on WordNet and Freebase when the inference graph is augmented with validation triplets on which the model was not trained. Fig. 5.6 makes this point dramatically in the context of KBEGen. For entities held out from training, we gradually increased the proportion of the available inference graph—the entirety of which was held out in training—used to predict links from the test set. For both WNGen and FBGen, performance increases when new triplets are added, almost linearly in the case of WNGen. The concavity of the performance from added graph triplets in FBGen is likely not due to diminishing returns from the addition of information about particular entities, but rather to the fact that Freebase entities mostly have small neighborhoods, meaning that gains are felt mainly in the long tail.

Out-of distribution inference. It is instructive to regard this setting (WNGen and

FBGen) from the point of view of out-of distribution (OOD) inference [Liu et al., 2020]. It is required that the model generalize the learned inventory of memory states (and associated vector features) to unattested memory states about which the model has received partial memory states (absent key statistics), making the XGen tasks an instance of few-shot learning [Wang and Yao, 2019]. Due to the compositional nature of HMem entity-memory representations, the HMem model is structurally capable of deriving predictions for held-out entities. Its success on that task will depend on how closely the training samples resemble the held-out samples. Specifically, HMem models the joint distribution $p(\boldsymbol{e}_j, \boldsymbol{e}_i, \boldsymbol{e}_q, \boldsymbol{r}_q)$ as the modeled probability that $\mathtt{e}_j \in \hat{\mathtt{M}}_i(\boldsymbol{e}_i, \boldsymbol{e}_q, \boldsymbol{r}_q)$ (q for query), where the notion of set membership employed here refers to the probability that a TPR/HRR binding is one of the superposed elements in the memory state vector $\hat{\mathtt{M}}_i$ (which is a function of observed subgraph and also the query entity and relation). The features of the query entity and relation $\boldsymbol{e}_q, \boldsymbol{r}_q$ are learned, as well as those of the target entity $\boldsymbol{e}_j$ and those of all candidate links $(\boldsymbol{e}_c, \boldsymbol{r}_c)$ in the observed subgraph. Information about the conditional relationships between the modeled entity and its memory contents are omitted from the training set.

Are the held-out memory states out-of-distribution? The relatively lower performance on the XGen tasks in comparison with the standard Wordnet and Freebase evaluations suggests that yes, or at least partly so, but with the following caveat. The way we constructed the WNGen and FBGen datasets means that, one key statistic—the size of entity neighborhoods entering into inference—differed systematically between the training and test sets. In WNGen (FBGen), the 39,443 (13,951) held-in entities comprising the training set occur in 7.18 (71.12) triplets on average. In the test phase, inference is done on the basis of the observed subgraph, with each triplet having one and only one held-out entity, meaning that in WNGen (FBGen), memory states in the test phase

have 4.53 (62) neighbors on average, or 5.67 (77) if the validation triplets are added in. At the level of this coarse statistic, then, there is a mismatch between the training and testing samples in WNGen, though not in FBGen. The performance on FBGen is essentially asymptotic when all observed/valid triplets are included for inference (Figure 5.6), suggesting genuinely out-of-distribution structure. For WNGen, the gap between the memory state size between the training and test phase may account for the performance gap in this versus the standard setting. So, when in a context of applied knowledge base completion systems, deploying HMem networks in the open-world setting of extensible graphs depends on the degree to which the novel data serving as the subgraph for inference match those used in training, especially as relates to node connectivity. On this point, additional evaluation is required.

5.7 Revisiting the binding models from Chapter 2

In Chapter 2, we evaluated an array of bilinear binding models in controlled simulations, finding that (1) convolution-correlation decoding is equivalent to generic bilinear binding when the assumptions of that method—isotropy of the input representations—are satisfied, and (2) that violation of those assumptions leads to severe degradation of those methods. We now return to the question of bilinear binding models' performance, this time in an applied setting of knowledge base completion in WordNet.

Using WordNet data as the input introduces a primary new factor—correlations between fillers and roles—that was not present in the simulations, e.g. the probability of `being`—a top-level category—occurring as the left entity of the `hypernym` relation, as opposed to `napoleon`—a proper name—occurring in the same position. Such correlations are extremely difficult to correct for via the decorrelation transformations discussed

in Section 2.3.3. Whereas it is computationally tractable to apply transformations to the individual elements (entity and relation vectors) of a TPR input to bilinear binding—including HRRs—applying a transformation that includes all sources of correlation requires, via the most direct route, computing the inverse of a $d_E d_R \times d_E d_R$ matrix, with time complexity $\mathcal{O}({d_E}^3 {d_R}^3)$. On the other hand, it may be that, if encoding matrices are allowed to be directly learned from the data, much of the information about the relevant correlations, including an approximation to the decorrelating transformation, may be packed into the learned matrices.

We investigated this empirically by training several HMem models while varying the input binding functions. The optimization component of the model is the same as in the rest of this chapter. However, we introduced bilinear binding functions parametrized by $d_E \times d_R \times d_B$ tensors deployed as $E \otimes R \longrightarrow d_B$ bilinear maps. We also compared "Naive" versions of each representational method with those that implement the decorrelating transformation recapitulated in Appendix 1 of this chapter in connection with the HMem model (5.9). Varying the binding function $\mathbb{B}$, we trained each model on WordNet using output dimension 32, keeping the representations small so that they could be compared with a full TPR model, whose output representations have $32 \times 32 = 1024$ elements in the costly optimization step. The Naive versions input the entity and relation vectors as they were learned, while the non-Naive versions decorrelated entities and relations separately. Additionally, we evaluated a model in which the covariance matrix is estimated as a learned parameter rather than being computed directly from the embeddings. This allows us to implement the full computation specified in the non-naive version of the optimal decoding equation from Theorem 2.3.3 without actually computing the training sample covariance. The models are:

1. **Tensor Product**: $\mathbb{B}\left(\left\{\boldsymbol{e}_{i/\boldsymbol{r}_i}\right\}\right) = \sum_i \boldsymbol{r}_i \otimes \boldsymbol{e}_i \equiv \mathtt{M}_{\text{TPR}}$ where $\boldsymbol{r}_i$ and $\boldsymbol{e}_i$ are normalized. To unbind, dot the optimized TPR $\hat{\mathtt{M}}_{\text{TPR}}$ with the target relation vector $\boldsymbol{r}_t$.

2. **Naive HRR**: $\mathbb{B}\left(\left\{\boldsymbol{e}_{i/\boldsymbol{r}_i}\right\}\right) = \sum_i \boldsymbol{r}_i \circledast \boldsymbol{e}_i \equiv \mathtt{M}_{\text{HRR-N}}$. To unbind, do: $\boldsymbol{r}_t \star \hat{\mathtt{M}}_{\text{HRR-N}}$

3. **HRR**: Compute covariance matrices Ω_E, Ω_R for the entity and relation vectors, separately. Transform each entity and relation vector with the square root of the corresponding precision matrix and rescale by $1/\sqrt{d}$, then apply the HRR operations. Thus: $\mathbb{B}\left(\left\{\boldsymbol{e}_{i/\boldsymbol{r}_i}\right\}\right) = \sum_i \left({\Omega_R}^{-\frac{1}{2}}\ \boldsymbol{r}_i/\sqrt{d}\right) \circledast \left({\Omega_E}^{-\frac{1}{2}}\ \boldsymbol{e}_i/\sqrt{d}\right) \equiv \mathtt{M}_{\text{HRR}}$. Unbind with $\left({\Omega_R}^{-\frac{1}{2}}\ \boldsymbol{r}_t/\sqrt{d}\right) \star \hat{\mathtt{M}}_{\text{HRR}}$.

4. **Naive Bilinear Binding (Naive BB)**: Binding is parametrized by a bilinear map $W \in E \otimes R \otimes E$—reshaped to a $d_E \times d_R d_E$ matrix—where d_R and d_E are in general permitted to differ from one another (in contrast to HRRs). The forward map is $\mathbb{B}\left(\left\{\boldsymbol{e}_{i/\boldsymbol{r}_i}\right\}\right) = \sum_i W\ \text{vec}(\boldsymbol{r}_i \otimes \boldsymbol{e}_i) \equiv \mathtt{M}_{\text{BB-N}}$, which is a bilinear map on $R \otimes E$. The backward map is computed as W^-, the Moore-Penrose inverse of W. The unbinding map is the dot product of the target role with the decompressed tensor product: $\boldsymbol{r}_t \cdot \text{unvec}\left(W^- \hat{\mathtt{M}}\right)$.

5. **Bilinear Binding (BB)**: The target of this model is to perform a transformation that optimizes the decoding error of Theorem 2.3.3 in the presence of anisotropy. The forward map to calculate $\mathtt{M}_{\text{BB}}$ is the same as (4) above: a learned encoding matrix applied to the TPR. For unbinding, we aim to calculate the modified MPI:

$$W^- = \Omega W^\top \left(W \Omega W^\top\right)^{-1}$$

where Ω is the covariance $\mathbb{E}\left[\mathtt{M}_{\text{BB}} {\mathtt{M}_{\text{BB}}}^\top\right]$. It is intractable to compute Ω directly, because doing so requires gathering the memory representations for every entity

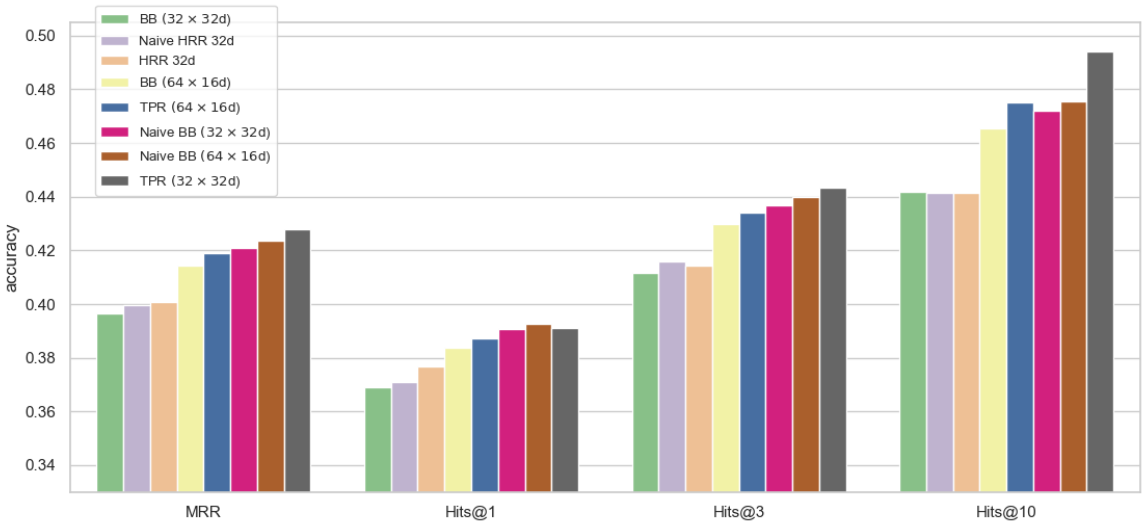


Figure 5.7: Results on WordNet using an array of bilinear binding methods.

using the entire training dataset, which must be re-assembled at each training step as the entity and relation vectors are updated. To cope, we specify Ω as a learned parameter and solve for W^- as usual, with Ω as an additional term in the MPI solution.

Observe that the Naive version of the TPR model is identical to the non-naive version, since the modified MPI in the case where the forward map M is the $(d_E d_R) \times (d_E d_R)$ identity matrix I yields $M^- = \Omega M^\top (M \Omega M^\top)^{-1} = \Omega \Omega^{-1} = I$.

Results are displayed in Figure 5.7. An advantage of the Bilinear Binding method, as distinct from HRRs, is that it allows the vector dimension of entities and relations to differ. In this setting, we find that, among all the models with TPR compression, Naive Bilinear Binding works best for every metric, with slightly better performance when using this degree of freedom ($d_E = 64, d_R = 16$) than the bilinear methods matching the input-output size of HRRs ($d_E = 32 = d_R$). Note that both the BB methods have same-sized input to the binding step ($d_{E \otimes R} = 1024$) and the same output size (32). The bilinear

models with learned covariance matrices Ω performed worse than the corresponding Naive models, perhaps a result of overparametrization.

The naive and non-naive versions of the HRR model performed about equivalently. In Section 2.3.5, we found that isotropy was required for HRRs to be serviceable. Why, then, would the decorrelating transformation applied to HRRs not lead to greater performance? Recall that the transformation applied in this case only captures part of the correlation structure of the full TPRs—that associated with the distribution of roles and fillers taken separately, ignoring correlations between them. This may explain why that transformation is unhelpful in this setting. The relatively good performance of the Naive HRRs would therefore be attributable to the structure of the embeddings adapting itself to the binding mechanism rather than the other way around, so that that the resulting representations approximate isotropy. In contrast, the fact that we obtain better performance with the Naive versions of the Bilinear Binding models validates the hypothesis that the broad range of correlations between elements of the tensor products may be represented, if imperfectly, in the learned encoding maps. The large 32×32 TPRs performed best, but with relatively low gain given the quadratic increase in representation size.

5.8 Conclusion

This chapter has presented a neural model for knowledge base completion that is powerful enough to achieve state of the art results on large databases, and flexible enough to evolve with knowledge base content without retraining. The approach complements existing neighborhood-aggregation techniques (e.g. graph convolution), with the advantage of interpretable mechanisms: vector binding and memory completion. The results indicate that the model operates well at scale and in an open-world setting.

5.9 Appendix 1: Conditions on CCorr embeddings (decorrelation transformation)

As discussed in [Plate, 1994], a sufficient condition for circular correlation to approximately invert circular convolution is that the components of the vectors occurring in the memory are independently and identically distributed, with an expected norm of 1,[6] in which case the result the binding-unbinding sequence $\boldsymbol{x} \star (\boldsymbol{x} \circledast \boldsymbol{y}) = (1 + \eta)\boldsymbol{y} + \varepsilon$ where η and ε are zero-mean and approximately Gaussian noise terms [Plate, 1994, pg 66].

During learning, the entity and relations embeddings evidently depart from these strict conditions—as is desirable, since many of their latent features can and do covary. But to preserve the integrity of the decoding process, we enforce Plate's distributional constraints by applying a decorrelating transformation to the embeddings.[7] At each step of training or inference, the entity and relation embeddings are concatenated, and the resulting array is centered by calculating the mean embedding μ and subtracting it from each embedding. Let $\boldsymbol{E}$ denote the $(|\mathcal{E}| + 2|\mathcal{R}|) \times d$ matrix of centered relation and entity embeddings. We calculate the empirical covariance matrix of the embeddings Σ_{emp} and regularize it to produce an estimate $\hat{\Sigma}$.

$$\Sigma_{\text{emp}} = \frac{1}{|\mathcal{E}| + 2|\mathcal{R}|}\boldsymbol{E}^\top \boldsymbol{E}$$

$$\hat{\Sigma} = (1 - \alpha)\Sigma_{\text{emp}} + \alpha I$$

α was set to .2. The regularized estimate of the covariance is then used to calculate the precision matrix $\hat{\Sigma}^{-1}$ for the centered embeddings. This defines a decorrelating—or

[6] This can be guaranteed by setting the componentwise variance of the input vectors to $\frac{1}{d}$.

[7] Early experiments confirmed that applying decorrelation improved performance with the CConv model.

"whitening" [Diedrichsen and Kriegeskorte, 2017]—transformation for any embedding v, obtained by centering the vector and then post-multiplying it with the square root of the precision matrix, divided by $\sqrt{d}$ to ensure a variance of $\frac{1}{d}$ in every direction:

$$\hat{\boldsymbol{v}} = \frac{1}{\sqrt{d}}(\boldsymbol{v} - \boldsymbol{\mu})\hat{\Sigma}^{-\frac{1}{2}}$$

where $\boldsymbol{\mu}$ is the empirical average of all entity and relation vectors. The resulting distribution of transformed embeddings is approximately spherical with variance $\frac{1}{d}$ and an expected norm of 1. When this transformation is applied to all vectors involved in binding and unbinding, Plate's conditions are met.

5.10 Appendix 2: Ablation

To evaluate the role of each model component, we ablated (1) the memory-completion operation, in which case the network's goal is to obtain held-out links as weighted sums of known links, and (2) conditioning the weight matrix in Eqn. 5.8 on the location of the input memory $\mathtt{M}_i$. Ablation (1) is implicit in setting the hyperparameter $\lambda = \infty$. We performed ablation (2) by keeping the weight matrix constant across all choices of $\mathtt{M}_i$, completing the memory using Eqn. 5.8 where $\mathbb{W}_i$ is set to $\mathbb{W}_{\text{global}}$.

Results The results of ablation (1) on the primary models are discussed in the main text. Ablation of the conditional weight matrix substantially affected performance with the convolution models, so that the best-performing models are those where $\mathbb{W}_i$ is recomputed for each $\mathtt{M}_i$ (Table 5.5). Ablated TPR models performed slightly better on FREEBASE, while on WORDNET showed better performance on the Mean Rank metric, but were substantially outperformed by unablated TPR models on the key evaluation

metrics, MRR and Hits@1, generally understood as the final arbiters of model performance. Interestingly, in contrast to the main results, where adding the validation graph to the TPR model improved performance on FREEBASE, doing so with the global weight matrix is marginally harmful.

	WordNet					Freebase				
Model	MR	MRR	H@1	H@3	H@10	MR	MRR	H@1	H@3	H@10
HMEM-CCONV	639	.774	.719	.818	.863	336	.456	.372	.517	.597
HMEM-CCONV+	475	.787	.730	.880	.952	361	.442	.362	.500	.575
HMEM-CCONV$_\infty$	815	.842	.793	.884	.916	336	.456	.372	.517	.597
HMEM-CCONV$_\infty$+	690	.854	.804	.900	.930	361	.442	.362	.500	.575
HMEM-TPR	209	.854	.796	.908	.932	303	.449	.361	.507	.601
HMEM-TPR+	124	.863	.804	.920	.944	321	.440	.355	.496	.586
HMEM-TPR$_\infty$	154	.866	.784	.923	.950	**36**	**.618**	**.523**	**.677**	**.786**
HMEM-TPR$_\infty$+	**110**	**.868**	**.796**	**.935**	**.963**	37	.616	.521	.674	**.786**

Table 5.5: Results of ablation of memory-conditioned weight matrix (see Appendix 2 text).

Chapter 6

Spatial Attention Networks

In this chapter, we develop a model—Spatial Attention Networks (SAN)—in which we adopt a geometrical view of Tensor Product Representations as 3-dimensional volumes with a spatial structure. We develop a spatial attention mechanism that selects subgroups of representation units (TPR components) on the basis of which inference will occur. At an analogical level, this approach may be likened to the actual spatial organization of the human brain, in which information regarding particular feature types (e.g. visual vs. semantic) and particular domains (e.g. humans vs. tools) are selectively encoded in contiguous volumetric units. Indeed, this is the basic premise of the "searchlight" procedure in cognitive neuroscience [Kriegeskorte et al., 2006], and the bulk of research is consistent with it.

At an analytical level, directing network attention to subsets of components in a tensor representation of the graph may be regarded as an instantiation of a set of proposals based on modeling knowledge bases using order-3 tensor factorization. Let $\mathcal{E}$ and $\mathcal{R}$ be the sets of entities and relations, and further let $G \in \mathbb{R}^{|\mathcal{E}|} \otimes \mathbb{R}^{|\mathcal{R}|} \otimes \mathbb{R}^{|\mathcal{E}|}$ be the binary tensor with $G_{ijk} = 1$ if $(\mathtt{e}_i, \mathtt{r}_j, \mathtt{e}_k)$ is in the graph $\mathcal{G}$, and 0 otherwise. The goal is to find

$W \in \mathbb{R}^{d_E} \otimes \mathbb{R}^{d_R} \otimes \mathbb{R}^{d_E}$, a low-rank approximation of G, by factoring G as

$$G \approx f\left(E \cdot_1 \left(R \cdot_2 \left(E \cdot_3 W\right)\right)\right) \tag{1}$$

where E and R are the entity and relation embedding matrices, f is some linking function, and $\cdot_m$ is the inner product of the trailing axis of the embedding matrices along the m^{th} mode of W. The resulting tensor should have high values in the i, j, k cell if $(\mathtt{e}_i, \mathtt{r}_j, \mathtt{e}_k)$ is in the knowledge base, and low values if it is not. As observed in [Lacroix et al., 2018b] and [Balazevic et al., 2019b], a number of existing knowledge base representation techniques fit within this general schema, parameterized by the selection of particular components of the approximation matrix W. For instance, the DistMult model [Yang et al., 2015] (elementwise multiplication of entity and relation vectors) sets the elements W_{ijk} to 1 if $i = j = k$, and 0 otherwise, i.e. taking the superdiagonal of the tensor $\boldsymbol{e}_1 \otimes \boldsymbol{r} \otimes \boldsymbol{e}_2$. The Rescal model [Nickel et al., 2011]—where each relation is embedded as a bilinear form R_r with triplets scored as ${\boldsymbol{e}_1}^\top R_r \boldsymbol{e}_2$—can be rewritten with relation vectors $\boldsymbol{r}$ with ${d_E}^2$ elements indexed as tuples (i, j) in the closed integer intervals $[1, d_E] \times [1, d_E]$. W is then the fixed tensor with $W_{i,(k,\ell),j} = 1$ if $i = k$ and $\ell = j$, and 0 otherwise. Other cases, like ComplEx [Troullion et al., 2016], can also be thus subsumed (see [Lacroix et al., 2018b, Balazevic et al., 2019b]). In the most general case, W is learned rather than preset, allowing the network to learn which third-order interactions in the three-way product $\boldsymbol{e}_1 \otimes \boldsymbol{r} \otimes \boldsymbol{e}_2$ to pay attention to, and with what weight.

Taking the components of the factored knowledge graph (i.e. the embedding matrices E and R as well as the graph kernel W) as the input, our approach builds in a dynamic selection of components of the learned weight matrix W, using the vectors for a given query to produce a distribution—the attention—over elements of compressed graph represen-

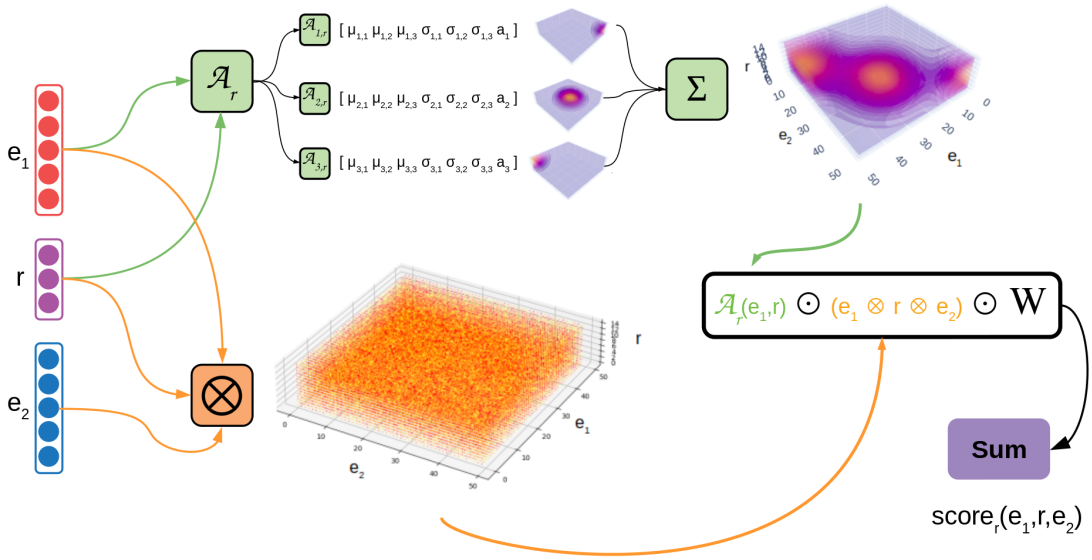


Figure 6.1: Spatial Attention Network architecture. Entity and relation vectors go through two streams of computation. The query is $(\mathtt{e}_1, \mathtt{r}, ?)$. The first stream (green) computes a 3-dimensional attention mask. $\boldsymbol{e}_1$ and $\boldsymbol{r}$ are combined to yield parameters for the H-headed attention distribution, each head a Gaussian. These are summed together to produce an attention mask $\mathcal{A}_\ell(\boldsymbol{e}_1, \boldsymbol{r})$. The second stream (orange) produces a third-order tensor with the same dimensions as the attention mask—the tensor product $\boldsymbol{e}_1 \otimes \boldsymbol{r} \otimes \boldsymbol{e}_2$. The results of both streams are multiplied elementwise with a third-order tensor W, with all elements of the resulting tensor summed to yield a score for the triplet. For simplicity, we depict a single vector for each entity and relation, entering into both the tensor product module and the attention module. Some of the models have separate sets of vectors for each module (see Results 6.3).

tation which are then used in inference. Specifically, we constrain the attention weights to a form which preserves the spatial distribution of tensor elements conceptualized as a volume (see Fig. 6.1).

Embeddings to volumes. We start by embedding each entity and each relation in d_E and d_R-dimensional spaces, respectively. For a given triplet $(\mathtt{e}_\ell, \mathtt{r}, \mathtt{e}_r)$, there are vectors e_ℓ, r, e_r, which are combined using the tensor product $(\otimes)$ to create a $d_E \times d_R \times d_E$-

dimensional tensor embedding for the triplet:

$$\text{embed}(\mathtt{e}_\ell, \mathtt{r}, \mathtt{e}_r) = \boldsymbol{e}_\ell \otimes \boldsymbol{r} \otimes \boldsymbol{e}_r$$

The resulting tensor can be visualized as a 3-dimensional space (a volume) whose coordinates are three-tuples of the indices of the vectors $\boldsymbol{e}_\ell, \boldsymbol{r}, \boldsymbol{e}_r$.

Attention and attention-free models. Ultimately, candidate triplets are evaluated with respect to a third-order weight tensor $W \in E \otimes R \otimes E$. In the simplest model, without attention, this is done by dotting the triplet embedding with $\mathbb{W}$ to yield a score for each candidate. In models with attention, we perform the additional step of multiplying the weight matrix elementwise with an attention distribution computed from the entity and relation vectors for the query. As in Chapters 4 and 5, queries come in the form of an entity-relation pair with either the left or right entity to be selected from a list via ranking. For instance, for the query $(\mathtt{soul}, \mathtt{part\_of}, ?)$, a right-query attention function $\mathcal{A}_r$ computes an attention volume $\mathcal{A}_r(\text{soul}, \text{part\_of})$ from the vectors for `soul` and `part_of`. In the two cases, the scores are therefore computed as:

$$\text{score}(\mathtt{e}_1, \mathtt{r}, \mathtt{e}_r) = \langle \boldsymbol{e}_1 \otimes \boldsymbol{r} \otimes \boldsymbol{e}_2, \mathbb{W} \rangle \qquad \text{(Base model)} \tag{2}$$

$$\text{score}_r(\mathtt{e}_1, \mathtt{r}, \mathtt{e}_2) = \langle \boldsymbol{e}_1 \otimes \boldsymbol{r} \otimes \boldsymbol{e}_2, \mathcal{A}_r(\boldsymbol{e}_1, \boldsymbol{r}) \odot \mathbb{W} \rangle \qquad \text{(Attention model)} \tag{3}$$

where $\langle a, b \rangle$ is the inner product and $\odot$ is elementwise multiplication. The scores for the Attention model are indexed to the query direction because the corresponding attention functions only take the relation and query entity vector as inputs—$\boldsymbol{e}_1$ for $\mathcal{A}_r$ in query $(\mathtt{e}_1, \mathtt{r}, ?)$, and $\boldsymbol{e}_2$ for $\mathcal{A}_\ell$ in query $(?, \mathtt{r}, \mathtt{e}_2)$.[1]

[1]Query direction-specific attention heads are necessary for computational reasons. The training regime requires negative sampling of possible alternative completions of the triplet. It is well known that tuning

Multi-headed Gaussian attention. We adopt a Gaussian functional form for the attention distributions. For each of H attention heads (indexed by h), we train a subnetwork to generate parameters for a Gaussian distribution centered at a point in the volume. Each attention head generates a parametric normal distribution over the 3-dimensional coordinates of the weights. For a given (right-directed) attention head $\mathcal{A}_{h,r}$ (with left-directed attention heads indexed as $\mathcal{A}_{h,r}$, the output is 3 mean parameters (a vector $\mu_h = [\mu_{1,h}, \mu_{2,h}, \mu_{3,h}$ computed from $[m_{1,h}, m_{2,h}, m_{3,h}]$) and 3 variance parameters (a diagonal matrix Σ_h with diagonal elements computed from $[s_{1,h}, s_{2,h}, s_{3,h}]$), as well as an attention weight a_h. All of the distributional parameters—including the attention weight, are computed using the "Nonlinear-Bilinear" function (4).

$$[m_1, m_2, m_3, s_1, s_2, s_3, a] = W_{\text{out}} \left(\tanh(W_{e,\ell}\boldsymbol{e}_1 + \boldsymbol{b}_{e,\ell})^\top \; V \; \tanh(W_{r,\ell}\boldsymbol{r} + \boldsymbol{b}_{r,\ell})\right) + \boldsymbol{b}_{\text{out}} \tag{4}$$

where V is a $d_E \times 7 \times d_R$ bilinear weight matrix $\boldsymbol{b}_{\text{out}}$ a bias vector, both shared for left and right queries.

Other combination functions examined. We investigated a number of different combination functions for computing the attention. In each case, W_{out} is a matrix (or tensor) mapping into the 7-dimensional space of distribution parameters (means, variances, and attention weight). Indices to the query direction (left or right) are removed for simplicity, but in general some or all of the parameters are specific to a direction, except where otherwise indicated:

---

the batch size and negative sampling rate parameters for knowledge base completion models has sizable effects on the model performance [Kadlec et al., 2017]. The H attention masks computed based on just the query entity and relation collectively have size $h{d_e}^2 d_r n_b$. If the attention is additionally conditioned on the second entity, this becomes $h{d_e}^2 d_r n_b n_n$. For single-headed attention with competitive batch sizes (1024) and negative sampling rates (512 per training triplet) with $d_e = 75, d_r = 25$ as in the experiments, this is 73 billion real numbers, which is intractable for our computational resources.

(5) a. **Elementwise multiplication** $W_{\text{out}}(W_\ell \boldsymbol{e}_q \odot W_\ell \boldsymbol{r}_q) + \boldsymbol{b}_{\text{o}}$

Linearly mapping the entity and relation vectors into a common space (using W_e and W_r), multiplying this result elementwise, and linearly mapping this into the 7-parameter space (W_{out}))

b. **Simple nonlinear** $W_{\text{out}}\ \sigma(W_{in}(\boldsymbol{e}_q \oplus \boldsymbol{r}_q))$

Taking the concatenation of entity and relation vectors, and feeding these through a 2-layer nonlinear network. We tried versions where W_{out} is specific to direction and where it is shared across directions—with directionality encoded by the input map W_{in}.

c. **Simple bilinear** $\boldsymbol{e}_q \cdot W_{\text{out}} \cdot \boldsymbol{r}_q + \boldsymbol{b}^\top(\boldsymbol{e}_q \oplus \boldsymbol{r}_q)$

Application of a bilinear map to the pair of input vectors $\boldsymbol{e}_q, \boldsymbol{r}_q$, where W_{out} is a $d_e \times 7 \times d_r$-dimensional tensor.

d. **Linear attention** $W_{\text{out}}(\boldsymbol{e}_q \oplus \boldsymbol{r}_q) + \boldsymbol{b}^\top(\boldsymbol{e}_q \oplus \boldsymbol{r}_q)$

A linear map applied to the entity and relation vectors.

The array of models investigated were largely variations on the basic schemata indicated above.

The parameters $W_{e,\ell}, \boldsymbol{b}_{e,\ell}, W_{r,\ell}$, and $\boldsymbol{b}_{r,\ell}$ are specific to right-hand queries—i.e. querying $(\mathsf{e}_1, r, ?)$ using the left entity vector $\boldsymbol{e}_1$—and the rest are shared for both directions. The output is then nonlinearly transformed, bounding (6) the mean within the $d_E \times d_R \times d_E$ box and (8) the variance between a minimum and maximum variance $\min_{\text{var-}i}, \max_{\text{var-}i}$ (for the ith coordinate), which are set as hyperparameters:

$$\boldsymbol{\mu} = [\mu_1, \mu_2, \mu_3] = [\tanh(m_1)\tfrac{d_e}{2}c + g_1, \tanh(m_2)\tfrac{d_r}{2}c + g_2, \tanh(m_3)\tfrac{d_e}{2}c + g_3] \tag{6}$$

$$\boldsymbol{\sigma} = [\sigma_1, \sigma_2, \sigma_3] \tag{7}$$

(8) $\sigma_i = \min_{\sigma_i} + \text{sigmoid}(\sigma_i)(\max_{\text{var-}i} - \min_{\text{var-}i})$

(9) $\alpha = \text{sigmoid}(a)$

where c is a manually set dilation constant (generally set to 1) allowing the mean vector to be slightly larger than the maximum box coordinates, and $\boldsymbol{g} = [g_1, g_2, g_3]$ is the grid center.[2] The minimum and maximum values for σ_i are set to multiples of the corresponding dimensionalities. The covariance matrix Σ is $\text{diag}(\boldsymbol{\sigma})$.

These parameters define a Gaussian density, which we do not renormalize with the partition function (i.e. so that the value of the attention at the mean is 1). The Gaussian for this head is then multiplied by α, which weights each attention head, turning it on or off:

$$\mathcal{A}_{h,r}(\boldsymbol{e}_1, \boldsymbol{r})_{i,j,k} = \alpha \left(\exp \left\{ -([i,j,k] - \boldsymbol{\mu}_h)^\top \Sigma_h^{-1} ([i,j,k] - \boldsymbol{\mu}_h) \right\} \right)$$

where each $[i, j, k]$ is a grid point in $\mathbb{R}^{d_e} \times \mathbb{R}^{d_r} \times \mathbb{R}^{d_e}$. The output of attention head is an order-3 tensor, of the same size as the triplet representation $\boldsymbol{e}_1 \otimes \boldsymbol{r} \otimes \boldsymbol{e}_2$. Finally, the results from each attention head are summed to produce a distribution as a weighted sum of the distributions from each head:

$$\mathcal{A}_r(\boldsymbol{e}_1, \boldsymbol{r}) = \sum_h \mathcal{A}_{h,r}(\boldsymbol{e}_1, \boldsymbol{r})$$

To obtain a score for each triplet, the attention mask is then input into the scoring equation (3), yielding a dot product of the attention-masked weights with the tensor product representation of the triplet.

[2] e.g. if $d_e = 75, d_r = 25$, the grid center is $[37, 12, 37]$ (indices range from 0 to 74 for the entity, 0 to 24 for the relation)

6.1 Computational efficiency considerations

In principle, it is possible to select Gaussian densities with non-diagonal covariance matrices Σ_h by generating the 6 independent elements of the covariance matrix (3 diagonal and 3 off-diagonal cells). However, the representations thus constructed are very large. Let $t = |E|^2|R|$ denote the size of the tensor product representation T, and T_3 the spatial coordinate-tensor with $3t$ elements. The cost of the bilinear operation is $3t \times 9$ products (the multiplication of Σ_h^{-1} with each coordinate $[i, j, k]$ in T_3), with the $3t$ elements of the output multiplied elementwise by T ($3t$ products). This step thus requires $30t$ products for each head. The output is the attention mask, which is then multiplied elementwise by T. For instance, if (as in our experiments) the entities are 75-dimensional and with 25-dimensional relations, the triplet representations have $t = 140,625$ components. The spatial coordinate tensor T_3 has 421,875 components, and computing the density requires $31t = 4,359,375$ products, multiplied by the number of heads. Computations on this scale exceeded available resources.

The computations can be simplified by employing diagonal covariances, in which case the masks can be calculated for each constituent (entities and relations) as 1d Gaussian masks for each head parameterized by the mean and the 1d variances. As a result, there are now $|E|$ products for the two entities from applying the mask, another $|R|$ from applying the mask to the relation, with the masked triplet representation given by the tensor product of the three. This yields a complexity of $2|E| + |R| + |E|^2|R|$, which in the use-case above corresponds to 140,800 products—a 30-fold speed-up.

In early experiments, we explored non-diagonal covariance structures and found that they performed about equivalently to the diagonally-parametrized densities. Given those models' added computational cost, those reported here uniformly employed diagonal vari-

ances.

6.2 Datasets

To evaluate SAN, we used the standard benchmark datasets as well additional databases constructed with the intention of probing the qualitative aspects of the model's distribution of attention. The intuition behind the SAN approach is that the use of attention might drive the model to allocate information relevant to a particular inference-domain to different regions of the input vectors, to the effect that, for instance, information about an entity's physical concrete aspects (e.g. shape, size—physical attributes) and its abstract aspects (e.g. taxonomy) might be spatially separated. Noting that, in all of the standard datasets (WordNet, Freebase, etc.), there is a wide variety of entities and relations (e.g. WordNet includes humans, countries, animals—including species and higher-order taxonomic objects—abstract categories, and much more), we constructed subsets of Wikidata[3], a large-scale knowledge base that aims to convert Wikipedia into the standardized $(\mathtt{e}_1, \mathtt{r}, \mathtt{e}_2)$ format of knowledge bases. In addition to evaluating the Spatial Attention concept on larger datasets, using a dataset that is restricted to a particular domain of entities allows us to qualitatively probe the spatial organization of information arrived at by the model after training—for instance, whether information about time period is spatially dissociated from information about geographical location, profession, etc.

The two new datasets—Humans and Companies—were built using [Boschin and Bonald, 2019]'s collections of domain-specific subsets of Wikidata, consisting of all triplets rooted at the "topic" nodes `human` and `business`. Taking Humans as the

[3] `https://www.wikidata.org/`

Dataset	Min node occurrence	Min attr occurrence	n-nodes	n-attributes	Attr. in test?	Train	Validation	Test
HUMANS	25	10	63,166	124,282	X	1,383,214	10,000	8,930
COMPANIES	10	5	7,855	8,721	✓	56,468	5,000	4,964

Table 6.1: Statistics for the WIKIDATA subsets HUMANS and COMPANIES.

example, each WIKIDATA entity `e` that occurs in the triplet (`e`, `instance_of`, HUMAN) was labeled a "node" and included in the HUMANS subgraph. All within-domain relations between humans were retrieved, e.g. `sibling`, `child`, `doctoral_advisor`, `godparent`, etc. In addition, all "attributes" of within-domain entities were harvested, meaning all WIKIDATA entities that were within one hop of a within-domain entity with respect to some relation. Thus, since (`heraclitus`, `occupation`, `philosopher`) is a WIKIDATA triplet, `philosopher` was included in the dataset as an attribute.

To derive domain-specific subsets suitable for the knowledge base completion task, we removed the knowledge bases first for infrequent entities, and then for attributes that were infrequent for those entities. Due to the size of the HUMANS dataset, we made inference more tractable by only including node links (not links to the out-of-domain attributes) as triplets in the test phase, holding out 10% of the 189,304 within-node triplets as test/validation sets. The dataset statistics are presented in Table 6.1.

We also evaluated the model on the NELL-995 dataset introduced to KBC in [Xiong et al., 2017]. NELL is a knowledge graph consisting of high-confidence facts gleaned from the automated Never-Ending-Language Learner initialized in 2010 [Carlson et al., 2010], which induces KB-format facts from text data. The NELL database has a natural entity-type system, with, for instance, `mick_jagger` occurring with the prefixes `actor` in triplets about his marriage with `jerry_hall`, and `musician` in facts about the `rolling_stones`. This led to the following architectural tweak: we split each entity symbol into *type* and *entity* tokens, learning separate embeddings for both enti-

HUMANS

Entity 1	Relation	Entity 2
sidney_moncrief	position_on_team	shooting_guard
quentin_roosevelt	father	theodore_roosevelt
nathanael_greene	cause_of_death	hyperthermia
virgin_mary	child	jesus_christ
ferenc_farkas	occupation	music_pedagogue
sir_richard_ingoldsby	honorific_prefix	sir
louis_ix_of_france	manner_of_death	natural_causes

COMPANIES

Entity 1	Relation	Entity 2
svenska_dagbladet	place_of_publication	stockholm
hotel_bristol	instance_of	hotel
harman_international_industries	subsidiary	martin_professional
the_niu_ridge	payment_types_accepted	maestro
theodore_roosevelt_island	located_in	washington_d.c.
gazprom_neftekhim_salavat	industry	petroleum_industry
dreamville_records	country	united_states_of_america

Table 6.2: Example triplets from the HUMANS and COMPANIES datasets.

ties and their types. We inserted compositionality into the representations by combining entity and type vectors when constructing queries, both during training and inference. Candidate vectors were generated using a simple composition function f applied to the pair of vectors, so that the embedding for musician_mick_jagger was obtained as $f(\texttt{musician}, \texttt{mick\_jagger})$. We report the results for simple addition—$f(\texttt{musician}, \texttt{mick\_jagger}) = \boldsymbol{v}_{\texttt{musician}} + \boldsymbol{v}_{\texttt{mick\_jagger}}$, which worked better than elementwise multiplication. Both combination functions performed better than using undecomposed entity symbols.

Model	MR	MRR	Hits@1	Hits@3	Hits@10
M3GM† [Pinter and Eisenstein, 2018]	2193	.498	.454	-	.590
GAAT [Wang et al., 2019]	**1270**	.467	.424	.525	.604
Inverse Model [Dettmers et al., 2017b]	13526	.348	.348	.348	.348
TPR Base	3858	.364	.344	.371	.398
SAN 2h	3463	.376	.353	.386	.416
Inverse Model+rev	13526	.348	.348	.348	.348
TPR Base+rev	**1180**	**.599**	**.572**	**.613**	**.645**
SAN 4h+rev	1656	**.605**	**.580**	**.619**	**.644**

Table 6.3: Performance of the SAN model on WN18RR. TPR Base denotes the baseline third-order model with scoring function (2). SAN 2h is the Spatial Attention Network with 2 attention heads. The models in the bottom panel ($\mathcal{M}$+rev) are trained on then WN18RR dataset —which removes inverse relation pairs from WN18—augmented with inverse triplets for each relation—i.e. we introduce a relation $\mathtt{r}^{-1}$ for each $\mathtt{r}$ and, for each training triplet $(\mathtt{e}_1, \mathtt{r}, \mathtt{e}_2)$, add the triplet $(\mathtt{e}_2, \mathtt{r}^{-1}, \mathtt{e}_1)$ to the training set—a regimen that is standard in this literature (e.g. [Xiong et al., 2017, Lacroix et al., 2018a]). The Inverse Model refers to a model introduced by the creators [Dettmers et al., 2017b] of WN18RR. The Inverse Model achieves state of the art performance on WN18, but not WN18RR. The results are identical in the standard and +rev regimens, which provides a sanity check for that approach. †: Best published.

6.3 Results

We obtained the best-published results on WN18RR dataset (Table 6.3. Our best model outperforms the state of the art by more than 10 points on the key MRR metric. These performance improvements hold for both the standard TPR model, as well as for the Spatial Attention Networks. Additionally, including the spatial attention mechanism yields increases on the most stringent metrics (MRR and Hits@1).

We also observed noteworthy performance improvements on the COMPANIES dataset from introducing Spatial Attention, which led to improvements on all metrics, including more than 2 points on MRR. Performance on the remaining datasets (Table 6.4) showed little difference between the baseline TPR model and those with spatial attention, with the baseline model performing better in most cases.

6.4 Discussion

Beyond performance, we were primarily interested in the way that the introduction of spatial attention would lead to a reorganization of information in entity, relation, and triplet representations relative to models without this mechanism. A great variety of knowledge base completion models are usefully characterized as subclasses of a TPR model, where the binary graph tensor is factored into an approximation. The characteristic distinguishing such models is which subset of elements of the tensor are selected for inference, this selection being static (DISTMULT, for example, statically selecting the diagonal elements of the tensor).

To examine the amount of spatial localization of information about different entity properties, we mapped the model's accuracy when selecting entity features using a sliding

All queries

	Model	MR	MRR	Hits@1	Hits@3	Hits@10
NELL-995	TPR Base	**752**	**.408**	**.334**	**.449**	**.536**
	SAN3h	833	.401	.326	.442	.534
	SAN3h-s	1812	.392	.315	.434	.532
FB15k-237	TPR Base	**124**	**.314**	**.228**	**.341**	.486
	SAN3h	**124**	.309	.221	.335	**.487**
	SAN4h-s	130	.307	.221	.334	.482
Humans	TPR Base	**154**	**.556**	.306	**.782**	.891
	SAN3h	177	.550	**.310**	.764	.888
	SAN4h-s	174	.541	.281	.777	**.895**

Left queries

	Model	MR	MRR	Hits@1	Hits@3	Hits@10
NELL-995	TPR Base	**1148**	.206	.159	.222	.290
	SAN3h	1171	**.208**	**.161**	.218	**.298**
	SAN3h-s	3142	.204	.155	**.226**	.294
FB15k-237	TPR Base	160	**.249**	**.171**	**.267**	.404
	SAN3h	**154**	.239	.155	.256	**.410**
	SAN4h-s	163	.240	.158	.260	.405
Humans	TPR Base	**152**	**.559**	**.307**	**.787**	.890
	SAN3h	177	.551	.306	.768	.890
	SAN4h-s	168	.545	.284	.781	**.894**

Right queries

	Model	MR	MRR	Hits@1	Hits@3	Hits@10
NELL-995	TPR Base	**357**	**.610**	**.510**	**.677**	**.782**
	SAN3h	495	.594	.490	.667	.769
	SAN3h-s	482	.579	.476	.641	.771
FB15k-237	TPR Base	**89**	**.380**	.285	**.414**	**.567**
	SAN3h	95	**.380**	**.288**	**.414**	.564
	SAN4h-s	98	.375	.284	.408	.560
Humans	TPR Base	**157**	**.552**	.305	**.777**	.892
	SAN3h	177	.548	**.314**	.759	.886
	SAN4h-s	180	.537	.279	.773	**.895**

Table 6.4: Results of the TPR Base and SAN graph models on NELL-995, FB15k-237, and the new Humans dataset. Models with separate entity and relation vectors for computing attention are suffixed as $\mathcal{M}$-s. The best performance in each category is bolded.

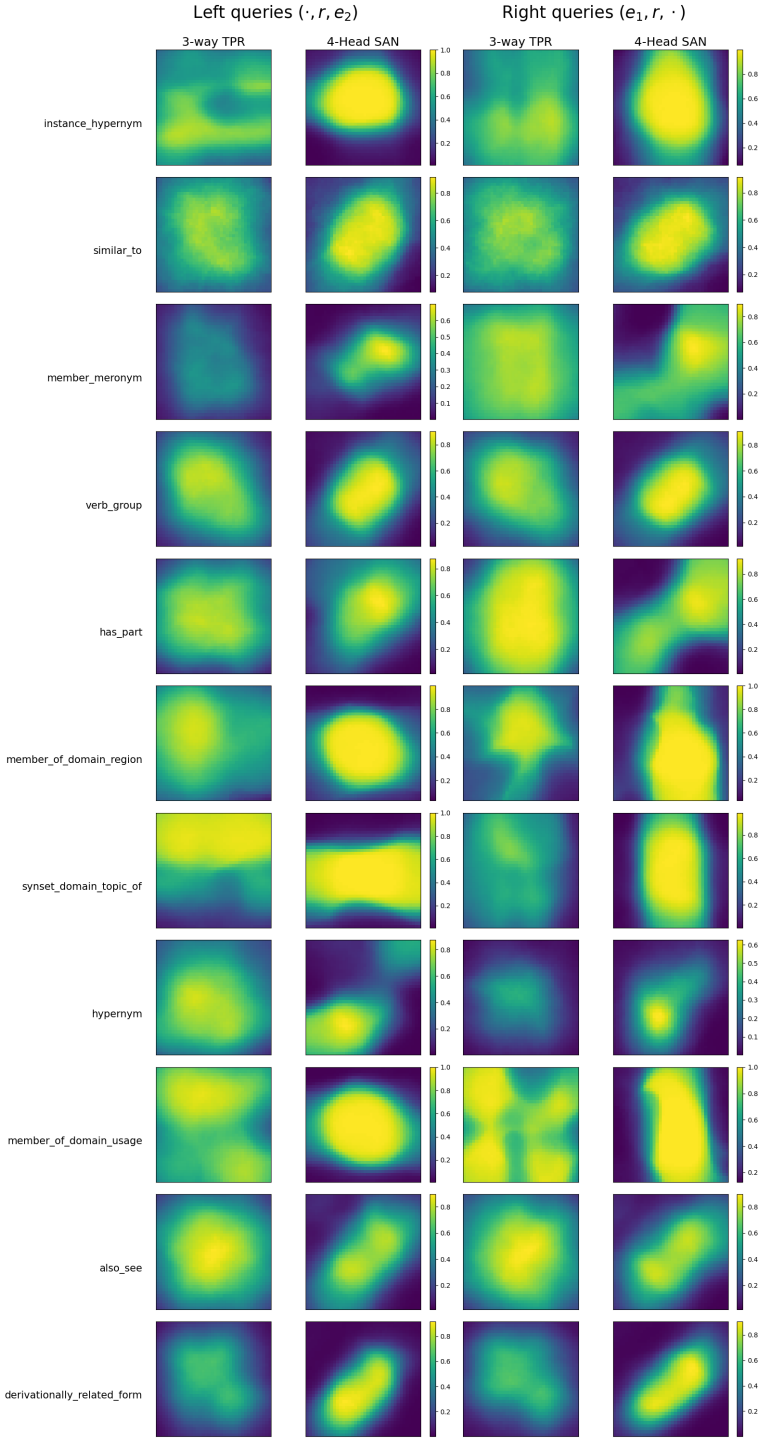


Figure 6.2: Accuracy (MRR) for the baseline TPR model and the 4-headed Spatial Attention Network for Left-headed and Right-headed queries when attention is centered at a pair of mean coordinates (m_1, m_2). We used a Gaussian "searchlight" centered at each pair of the 75 indices of $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$. For (m_1, m_2), we multiplied $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$ elementwise with Gaussian kernels centered at (m_1, m_2) and with the maximum variance allowed by the model. The variance in this case is 148, which is the variance allowed by the model that generated the attention maps in Figure 6.3. The diagonal $m_1 = m_2$ is the line from the top left to the bottom right. Accuracy is measured as the mean reciprocal rank (MRR), with colorbar values shared across the left-two and right-two columns to facilitate comparison of total accuracies across the pair of models for same-direction queries. See section 6.4 for discussion.

Model	MR	MRR	Hits@1	Hits@3	Hits@10
TPR Base	1164	.267	.197	.295	.405
SAN 3h	**983**	**.292**	**.220**	**.326**	**.421**

Table 6.5: Results on the Companies dataset.

Gaussian window applied to the WN18RR entity vectors. Figure 6.2 visualizes this accuracy on the plane defined by the two entity vectors, split by relations. Each point on the heatmap is the result of computing the accuracy (MRR) for a sample of triplets when multiplying each entity vectors with the Gaussian filters centered at the corresponding pair of components, with no filter applied to the relation. Accuracy ranges are indicated by the colorbars. Left- and right-directed queries are split within each panel.

We find that in comparison with the baseline TPR model (left panel), the trained SAN model displays clear spatial structure as well as a tight concentration of information at the regions peak accuracy. Comparing the accuracy ranges for the two pairs of heatmaps shows that, systematically, small regions of the plane can be used to obtain high accuracy on the SAN-trained model, whereas this is not the case for the simple TPR —confirming that information pertinent to different kinds of queries is spatially localized in SAN, and moroseo distributed in the case of the baseline model. Right-queries of `member_meronym` have a widespread accuracy profile and with a maximum accuracy of about .4 in the TPR baseline, whereas a small number of components are sufficient to yield well above .6 for the SAN models (the color bars to the right of each figure indicate the min and max values for each heatmap). The maximum value of queries of `derivationally_related_form` is around .65 for the TPR, and .85 for SAN. For both model types, the very edges of the plane tend to have low accuracy, which is simply a function of the fact that fewer components are activated at those edges.

As well, the resulting localization patterns reflect properties we would expect from the logical characteristics the corresponding relations. Symmetric relations like `similar_to`, `verb_group`, `also_see`, or `derivationally_related_form` are essentially transposes of one another across left- and right-queries, reflecting the fact that, for such relations, the model should assign the same score to $\boldsymbol{e}_1 \otimes \boldsymbol{r} \otimes \boldsymbol{e}_2$ as to the transpose $(\boldsymbol{e}_1 \otimes \boldsymbol{r} \otimes \boldsymbol{e}_2)^\top = \boldsymbol{e}_2 \otimes \boldsymbol{r} \otimes \boldsymbol{e}_1$. Transposition-invariance is thus a learned feature of the spatial maps. For highly asymmetric relations like `has_part` (`cat`, `paw`), or `member_meronym` (`wolf`, `pack`) do not show such symmetry.

Figure 6.3 displays the attention maps output by the attention module for four relations (rows), varying the entity (columns) and the query-direction (sub-rows). Attention distributions vary substantially as a function of all three factors (entity, relation, and query-direction)—indicating that the dynamic attention module retrieves information from the volume with greater precision than is suggested by the aggregate accuracy maps. Visually, these show the kinds of coarse generalizations one would anticipate. Entities at the same taxonomic levels (e.g. `cat`/`dog` and `feline`/`canine`) have attention distributions, across relations, that are more similar to each other than to their neighbors at different levels of the taxonomy. Conceptually similar entities with the same taxonomic rank (depth in the hierarchy) such as `president_fillmore` and `napoleon` have essentially the same attention maps across the board and yet have essentially disjoint graph neighborhoods, indicating that the relatively coarse category-distinctions reflected in the attention-allocation mechanism are in fact highlighting regions where informative features are likely to reside, with those regions containing the information itself.

There are two ways the model could be succeeding: by (1) direction to relevant information, or by (2) redirection to the answer. Suppose you go to the library and ask the librarian, "Where does the Pope sleep?" The librarian might (1) hand you a book about

the Pope, or (2) hand you a book called "The Vatican". In the first case, the librarian did not know the answer, but directed you to a source of information. In the second case, the librarian already knew the answer, and gave you a "location" that *is* the answer. In our model, (2) would be an effective strategy if, for instance, there as a single component in the representation denoting "The Vatican", the searchlight's job is to highlight that specific component. If the other argument of the query is not "The Vatican", then the score will be low. The question here is essentially, "Is the attention mechanism doing the job of the librarian (locating information to be used by the downstream module), or is the attention mechanism doing inference itself? If it is doing the inference itself, then we would expect the spatial maps to differ substantially even for near neighbors. Instead, we observe that near neighbors receive very similar attention maps. As illustrated in 6.3, the relevant conceptual dimensions that define similarity do not simply involve nearness in the sense of distance in the graph. The similarity maps for `cat` and `dog` are more similar to one another than the maps for `cat` and `feline`, despite the fact that `cats` are proper subsets of `felines`. Similarity in this case takes into account a conceptual dimension of taxonomic depth, along which `cat` and `feline` differ. The result is a model that has the anticipated structure: the attention mechanism provides a lookup mechanism for features encoded in the vector representations for the entities, rather than serving as the core inference engine itself.

6.5 Summary

In this chapter, we presented a novel graph completion model that takes into account the three-dimensional structure of graph element representations arising from the use of 3-way tensor products. The model allocates spatial attention to regions of a volume defined

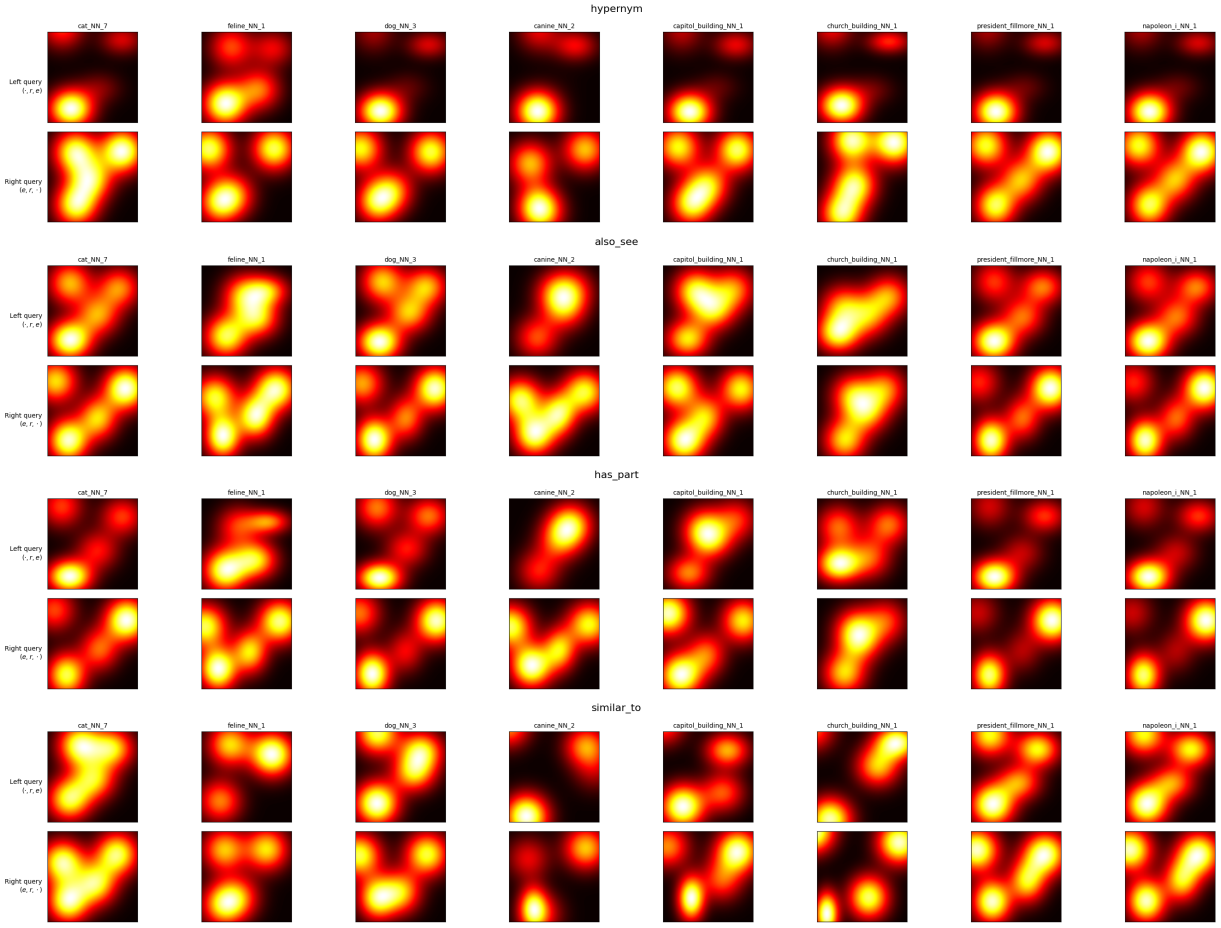


Figure 6.3: Attention distributions for a sampling of entities and four relations: `also_see` (top panel), `has_part` (middle panel), and `similar_to` (bottom panel) in WN18RR. The top row in each panel shows attention distributions for left queries—e.g. $(?, \texttt{has\_part}, \texttt{cat})$—and the bottom row for right queries—e.g. $(\texttt{cat}, \texttt{has\_part}, ?)$. Weighted attention distributions were computed for each head and then summed across head. We then took the central slice along the relation axis, yielding the planes visualized above. See section 6.4 for discussion.

by the components of the input vectors. We situate this model in a strand of research that encompasses a broad range of models found in the literature—in particular, all those whose scoring functions are strictly functions of products between embeddings for each entity and a relation. Such models—the trilinear forms—basically involve a selection of components of the 3-way products which enter into the final sum. Usually, these selections are made a priori, as in the case of DistMult, Rescal, HolE, etc. In contrast, our model selects products from the full tensor product in a graded fashion by weighting each component using a spatial attention model, and does so dynamically on the basis of the input representations themselves—one particular way of implementing dynamic selection of a trilinear form for graph completion. Interestingly, we find that the attention maps typically do not contain the core components of classic models like DistMult (the elements of the main diagonal in the 3-way tensor product), yielding instead asymmetric attention maps distributed across the entire representation. In addition to providing performance boosts in several databases, we suggest that this framework may serve as a way of investigating the properties of different trilinear models, and inducing such models directly from data.

Chapter 7

Conclusion

This dissertation has developed, implemented, and studied the properties of an array of novel machine learning models. The goal of the range of models we have examined is twofold: to take seriously the proposal, which is at the core of cognitive science as a field, that the form of the language of thought and its implementation as a symbolic system are tightly interwoven. The most prominent feature in our collective theories of the language faculty, incomplete as they are, is their description as the computations of machines operating over discrete sets, functions on those sets, and—in work that trods in the direction the quantitative description of the language system as it is deployed in real time—probabilities assigned to the elements of those sets.

From the other side, we have the recognition that the descriptive systems we adopt in the course of analyzing those systems bear limited relation to what is known about the machinery that carries those computations out. And whereas a variety of cognitive operations, in relation to their neural underpinnings can be understood rather well and to such an extent that precise mappings can be established between the neural circuitry and its resultant behavior, that knowledge is limited to what can be gathered in the course

of invasive experiments that have been unavailable in the study of higher cognition—by which I mean simply that which is unique to humans.

Here we have chosen, in a sense, to approach this problem from the top down: to select, among the bloom of models yielded by several decades' investigation into AI, those that are most amenable to analysis on the terms set forward by cognitive science. What seems an initially strict limitation is in fact quite expansive. The classical methods that implement the principle of compositionality in its most direct form, those that submit easily to formal analysis, are those based on Tensor Product Representations [Smolensky, 1990]. As it turns out, a large class of methods that share the basic properties realized by the tensor product—specifically, those that involve multiplicative interactions between components of vectors, i.e. the bilinear forms—all fit within a common framework that includes other classical proposals, such as that of Plate [Plate, 1994]—which aim to deal with the large space requirements of TPR-like systems—as well as more modern implemented systems that directly compute tensor products and then compress them with a linear map [McCoy et al., 2019]. All such models share a common core, and may be reasoned about in much the same way: as Encoders of structured representations (TPRs) into compressed versions of those representations, paired with Decoders that are optimal given those constraints.

TPRs as a class of models, then, provide a basis for investigating structured representations in a more applied setting. To that end, we have deployed three new models on the task of knowledge base completion. The task domain of KBC requires the representation of a large numbers of symbolic relationships, which are naturally approached using compositional methods used to construct the elements of a knowledge base—3-tuples constructed from entities and relations. We have shown how a broadly compositional approach to assembling a representation can be combined with methods drawn from con-

nectionist Energy functions—in linguistics, Optimality Theory and Harmonic Grammar [Smolensky and Legendre, 2006]—allow for both compositional representations with clear constituency relations between the elements, as well as adjustments made to those representations in the course of computation. These models, we show, can be analyzed in exactly this way: as compositional building-blocks paired with neural operations that adjust the representation so as to account for its context in the structure. In Chapters 4 and 5, we develop methods for analyzing those adjustments, and directly link them to the model's performance on benchmarks.

Finally, pursuing an existing line of research unifying a variety of KBC models in the common framework of tensor products and elements selected from them a priori, we developed a model in which there is a dynamic selection of components of the tensor product representations that capture three-way interactions between entities and relations. That model was based on an analogy with the spatial contiguity of co-functional voxels in the brain—a premise of the searchlight method in cognitive neuroscience. It and the rest of the ensemble of methods developed here represent state of the art performance on the KBC task, and in one case set a new state of the art (Chapter 6).

We hope that this work will bring new interest to the analysis of compositionality in neural systems for knowledge base representation.

Chapter 8

Bibliography

[Abadi et al., 2016] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., and Zheng, X. (2016). Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, pages 265–283, Berkeley, CA, USA. USENIX Association.

[Altmann et al., 2014] Altmann, C., Uesaki, M., Ono, K., Matsuhashi, M., Mima, T., and Fukuyama, H. (2014). Categorical speech perception during active discrimination of consonants and vowels. *Neuropsychologia*, 64:13–23.

[Asher, 2011] Asher, N. (2011). *Lexical Meaning in Context: A web of words*. Cambridge University Press.

[Balazevic et al., 2019a] Balazevic, I., Allen, C., and Hospedales, T. (2019a). Hypernetwork knowledge graph embeddings. In *International Converence on Artificial Neural*

Networks.

[Balazevic et al., 2019b] Balazevic, I., Allen, C., and Hospedales, T. (2019b). TuckER: Tensor factorization for knowledge graph completion. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5185–5194, Hong Kong, China. Association for Computational Linguistics.

[Barnes et al., 2008] Barnes, D., Hofacer, R., Zaman, A., Rennaker, R., and Wilson, D. (2008). Olfactory perceptual stability and discrimination. *Nature Neuroscience*, 11.

[Belanger and Kakade, 2015] Belanger, D. and Kakade, S. (2015). A linear dynamical system model for text. In *ICML 32*, volume 37.

[Bollacker et al., 2008] Bollacker, K., Evans, C., Paritosh, P., Sturge, T., and Taylor, J. (2008). Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 1247–1250, New York, NY, USA. ACM.

[Bordes et al., 2014] Bordes, A., Glorot, X., Weston, J., and Bengio, Y. (2014). A semantic matching energy function for learning with multi-relational data. *Machine Learning*, 94(2):233–259.

[Bordes et al., 2013] Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., and Yakhnenko, O. (2013). Translating embeddings for modeling multi-relational data. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 26*, pages 2787–2795. Curran Associates, Inc.

[Bordes et al., 2011] Bordes, A., Weston, J., Collobert, R., and Bengio, Y. (2011). Learning structured embeddings of knowledge bases. In *AAAI*.

[Boschin and Bonald, 2019] Boschin, A. and Bonald, T. (2019). Wikidatasets: Standardized sub-graphs from wikidata.

[Carlson et al., 2010] Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka Jr., E., and Mitchell, T. (2010). Towards an architecture for never-ending language learning.

[Castro et al.,] Castro, J., Ramanathan, A., and Chennubhotla, C. Categorical dimensions of human odor descriptor space revealed by non-negative matrix factorization. *PLoS ONE*, 8.

[Crawford et al., 2015] Crawford, E., Gingerich, M., and Eliasmith, C. (2015). Biologically plausible, human-scale knowledge representation. *Cognitive Science*, 13.

[Dasigi et al., 2017] Dasigi, P., Ammar, W., Dyer, C., and Hovy, E. (2017). Ontology-aware token embeddings for prepositional phrase attachment. In *ACL55*, pages 2089–2098.

[Dettmers et al., 2017a] Dettmers, T., Minervini, P., Stenetorp, P., and Riedel, S. (2017a). Convolutional 2d knowledge graph embeddings. *CoRR*, abs/1707.01476.

[Dettmers et al., 2017b] Dettmers, T., Minervini, P., Stenetorp, P., and Riedel, S. (2017b). Convolutional 2d knowledge graph embeddings.

[Diedrichsen and Kriegeskorte, 2017] Diedrichsen, J. and Kriegeskorte, N. (2017). Representational models: A common framework for understanding encoding, pattern component, and representational similarity analysis. *PLOS Computational Biology*, 13.

[Ebisu and Ichise, 2018] Ebisu, T. and Ichise, R. (2018). TorusE: Knowledge graph embedding on a lie group. In *Proceedings of AAAI 32*.

[Fodor and Pylyshyn, 1988] Fodor, J. and Pylyshyn, Z. (1988). Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28:3–71.

[Goldstone and Hendrickson, 2010] Goldstone, R. and Hendrickson, A. (2010). Categorical perception. *WIREs Cognitive Science*.

[Haley and Smolensky, 2020] Haley, C. and Smolensky, P. (2020). Invertible tree embeddings using a cryptographic role embedding scheme. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3671–3683.

[Hanley and Roberson, 2011] Hanley, J. and Roberson, D. (2011). Categorical perception effects reflect differences in typicality on within-category trials. *Psychonomic Bulletin Review*, pages 355–363.

[Hopfield, 1982] Hopfield, J. (1982). Neural networks and physical systems with emergent collective computational abilities. *PNAS*, 79:2554–2558.

[Ji et al., 2016] Ji, G., Liu, K., He, S., and Zhao, J. (2016). Knowledge graph completion with adaptive sparse transfer matrix. In *AAAI-16*, pages 985–991.

[Kadlec et al., 2017] Kadlec, R., Bajgar, O., and Kleindienst, J. (2017). Knowledge base completion: Baselines strike back. In *2nd Workshop on Representation Learning for NLP*, pages 69–74.

[Kazemi and Poole, 2018] Kazemi, S. M. and Poole, D. (2018). SimplE embedding for link prediction in knowledge graphs. In *Proceedings of NIPS*.

[Kingma and Ba, 2015] Kingma, D. and Ba, J. (2015). Adam: A method for stochastic optimization. In *ICLR*.

[Kriegeskorte et al., 2006] Kriegeskorte, N., Goebel, R., and Bandettini, P. (2006). Information-based functional brain mapping. *PNAS*, 103:3863–3868.

[Lacroix et al., 2018a] Lacroix, T., Usunier, N., and Obozinski, G. (2018a). Canonical tensor decomposition for knowledge base completion.

[Lacroix et al., 2018b] Lacroix, T., Usunier, N., and Obozinski, G. (2018b). Canonical tensor decomposition for knowledge base completion.

[Lalisse and Smolensky, 2019] Lalisse, M. and Smolensky, P. (2019). Augmenting compositional models for knowledge base completion using gradient representations. In *Proceedings of the Society for Computation in Linguistics*.

[Levin and Rappaport Hovav, 2005] Levin, B. and Rappaport Hovav, M. (2005). *Argument Realization*. Cambridge University Press.

[Liebesman and Magidor, 2019] Liebesman, D. and Magidor, O. (2019). Copredication, counting, and criteria of individuation: A response to gotham. *Journal of Semantics*, 36:549–561.

[Lin et al., 2015] Lin, Y., Liu, Z., Sun, M., Liu, Y., and Zhu, X. (2015). Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, pages 2181–2187.

[Ling and Weld, 2012] Ling, X. and Weld, D. (2012). Fine-grained entity recognition. In *AAAI 26*.

[Liu et al., 2020] Liu, W., Wang, X., Owens, J. D., and Li, Y. (2020). Energy-based out-of-distribution detection. *CoRR*, abs/2010.03759.

[McCoy et al., 2019] McCoy, R. T., Linzen, T., Dunbar, E., and Smolensky, P. (2019). Rnns implicitly implement tensor product representations.

[Miller, 1995] Miller, G. A. (1995). Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41.

[Nathani et al., 2018] Nathani, D., Chauhan, J., Sharma, C., and Kaul, M. (2018). Learning attention-based embeddings for relational prediction in knowledge graphs. In *ACL*.

[Nguyen et al., 2018] Nguyen, D., Nguyen, T., Nguyen, D., and Phung, D. (2018). A novel embedding model for knowledge base completion based on convolutional neural network. In *Proceedings of NAACL-HLT 2018*, pages 327–333.

[Nickel et al., 2016] Nickel, M., Rosasco, L., and Poggio, T. (2016). Holographic embeddings of knowledge graphs. In *AAAI*.

[Nickel et al., 2011] Nickel, M., Tresp, V., and Kriegel, H.-P. (2011). A three-way model for collective learning on multi-relational data. In *Proceedings of the 28 th International Conference on Machine Learning*.

[Niessing and Friedrich, 2010] Niessing, J. and Friedrich, R. (2010). Olfactory pattern classification by discrete neuronal network states. *Nature*, 465:47–52.

[Pinter and Eisenstein, 2018] Pinter, Y. and Eisenstein, J. (2018). Predicting semantic relations using global graph properties. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1741–1751, Brussels, Belgium. Association for Computational Linguistics.

[Plate, 1994] Plate, T. (1994). *Distributed Representations and Nested Compositional Structure*. PhD thesis, University of Toronto.

[Plate, 1995] Plate, T. (1995). Holographic reduced representations. *IEEE Transactions on Neural Networks*, 6:623–641.

[Prince and Smolensky, 2004] Prince, A. and Smolensky, P. (2004). *Optimality Theory: Constraint Interaction in Generative Grammar*. Wiley-Blackwell.

[Rosch, 1978] Rosch, E. (1978). Principles of categorization. In Rosch, E. and Lloyd, B., editors, *Cognition and Categorization*. Lawrence Erlbaum Associates.

[Schlichtkrull et al., 2017] Schlichtkrull, M., Kipf, T. N., Bloem, P., Berg, R. v. d., Titov, I., and Welling, M. (2017). Modeling relational data with graph convolutional networks. *arXiv preprint arXiv:1703.06103*.

[Smolensky, 1990] Smolensky, P. (1990). Tensor product variable binding and the representation of symbolic structures in connectionist networks. *Artificial Intelligence*, 46:159–216.

[Smolensky and Goldrick, 2016] Smolensky, P. and Goldrick, M. (2016). Gradient symbolic representation in grammar: The case of French liaison. Ms., Johns Hopkins University, Baltimore, MD.

[Smolensky et al., 2014] Smolensky, P., Goldrick, M., and Mathis, D. (2014). Optimization and quantization in gradient symbol systems: A framework for integrating the continuous and the discrete in cognition. *Cognitive Science*, pages 1102–1138.

[Smolensky and Legendre, 2006] Smolensky, P. and Legendre, G. (2006). *The Harmonic Mind: From Neural Computation to Optimality-Theoretic Grammar*, volume 1: Cognitive Architecture. The MIT Press.

[Socher et al., 2013] Socher, R., Chen, D., Manning, C. D., and Ng, A. (2013). Reasoning with neural tensor networks for knowledge base completion. In *NIPS*.

[Toutanova et al., 2015] Toutanova, K., Chen, D., Pantel, P., Poon, H., Choudhury, P., and Gamon, M. (2015). Representing text for joint embedding of text and knowledge bases. In *EMNLP*. ACL – Association for Computational Linguistics.

[Troullion et al., 2016] Troullion, T., Welbl, J., Riedel, S., Éric Gaussier, and Bouchard, G. (2016). Complex embeddings for simple link prediction. In *ICML 33*.

[Tupper et al., 2018] Tupper, P., Smolensky, P., and Cho, P. W. (2018). Discrete symbolic optimization and boltzmann sampling by continuous neural dynamics: Gradient symbolic computation.

[Vashishth et al., 2020] Vashishth, S., Sanyal, S., Nitin, V., Agrawal, N., and Talukdar, P. (2020). Interacte: Improving convolution-based knowledge graph embeddings by increasing feature interactions.

[Veličković et al., 2018] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. (2018). Graph attention networks. In *ICLR*.

[von Humboldt, 1999] von Humboldt, W. (1999). *On Language: On the diversity of human language construction and its influence on the mental development of the human species*. Cambridge University Press.

[Wang et al., 2019] Wang, R., Li, B., Hu, S., Du, W., and Zhang, M. (2019). *IEEE Access*, 8:5212–5224.

[Wang and Yao, 2019] Wang, Y. and Yao, Q. (2019). Few-shot learning: A survey. *CoRR*, abs/1904.05046.

[Whan Cho et al., 2017] Whan Cho, P., Goldrick, M., and Smolensky, P. (2017). Incremental parsing in a continuous dynamical system: sentence processing in Gradient Symbolic Computation. *Linguistics Vanguard*, 3.

[Xiong et al., 2017] Xiong, W., Hoang, T., and Wang, W. Y. (2017). DeepPath: A reinforcement learning method for knowledge graph reasoning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 564–573, Copenhagen, Denmark. Association for Computational Linguistics.

[Yang et al., 2015] Yang, B., Yih, W., He, X., Gao, J., and Deng, L. (2015). Embedding entities and relations for learning and inference in knowledge bases. In *Proceedings of ICLR*.

[Yoon et al., 2016] Yoon, H., Song, H., Park, S., and Park, S. (2016). A translation-based knowledge graph embedding preserving logical property of relations. In *Proceedings of NAACL-HLT*, pages 907–916.