GEOMETRIC PROBES FOR CONSTITUENCY STRUCTURE IN BERT EMBEDDINGS

THEODORE SANDSTROM

Date: March 9, 2023.

ABSTRACT. Since their introduction, transformer models such as BERT have revolutionized natural language processing with their ability to generate contextual word embeddings. It has been hypothesized that structural information about language is encoded in the geometry of these contextual embeddings. Recently, efforts have been made to embed BERT's embedding space into Riemannian manifolds such that syntactically related words are mapped to nearby points in the manifold. However, the ability of transformers like BERT to perform syntactically complex operations suggests a much deeper knowledge of syntax than a simple notion of when two words are syntactically related.

In this thesis, I assess the extent to which more specific and complex structural relations can be reliably recovered from BERT embeddings using geometric methods. Specifically, I investigate the extent to which (1) specific directions in Riemannian manifolds correspond with particular types of syntactic dependencies and (2) constituent subtrees are embedded in a consistent manner.

1. INTRODUCTION

Since their introduction to the field of natural language processing, transformer architectures have revolutionized the ways in which computers are able to interact with linguistic data. Particularly successful have been massive transformer models like BERT (Devlin et al., 2018), which quickly overtook state of the art models in a broad array of NLP tasks, or more recently the ChatGPT model developed by OpenAI which has become a pop culture phenomenon.

There is now significant evidence that transformer models of sufficient size are able to accurately perform tasks which require some degree of command over syntax. It has therefore become a major problem in computational linguistics to understand the mechanisms by which transformers encode and process syntactic data. One hypothesis first posed by Hewitt and Manning (2019) suggests that this mechanism is fundamentally geometric in nature. More specifically, they used a so-called structural probe to identify a subspace of BERT's embedding space in which words which are closely syntactically related are mapped to nearby points while more distantly related words are mapped farther apart from one another. Using this method, the authors were able to reconstruct dependency parses of sentences from BERT embeddings with a high degree of accuracy.

Building on this result, Auyespek et al. (2021) and Chen et al. (2021) have developed a modified structural probe which instead models the syntactic subspace of BERT using a hyperbolic space and

projects BERT embeddings into this hyperbolic space. It is a well documented phenomenon that models which embed data into hyperbolic spaces tend to exhibit an inductive bias toward hierarchical generalization patterns, in contrast with models embedding data into Euclidean spaces, which exhibit an inductive bias toward linear generalization patterns. This general pattern was reflected in the development of these hyperbolic structural probes, which outperformed their Euclidean counterparts even with lower-dimensional embedding spaces.

However, if BERT's mechanism of encoding syntax is fundamentally geometric, as these authors suggest, then we should expect the geometry of BERT's embeddings to reflect certain properties of the syntax of natural language. Specifically, it would be reasonable to expect BERT's embeddings to respect the compositionality of language: the notion that the meaning of a syntactic construction is built up from the meanings of its constituent pieces.

If BERT respects this property of language, we might expect the natural consequence that when a phrase with identical internal structure occurs in a variety of contexts, even though the absolute positions of the word embeddings might vary, the relative positions of the words in the phrase should be identical. More precisely, any two embeddings of the same phrase should be related by a rigid symmetry of the ambient space.

We might also reasonably expect that, since our probes are trained to extract the syntactic content of sentences, they should not be sensitive to purely semantic variations. Namely, if two phrases occur in the same context with identical syntactic structure but different semantic content, then their embeddings should be indistinguishable in the ambient space.

In this essay, we argue that if a structural probe fails to exhibit these properties, then it should be discarded as an explanation for BERT's encoding of syntax. We develop a mechanism for testing these properties, and use this mechanism to assess structural probes trained via various properties.

The essay is structured as follows. Section 2 provides background on the transformer architecture as well as a relatively thorough review of the fundamentals of Riemannian geometry. Section 3 reviews previous attempts to identify geometric encodings of syntactic data. Section 4 describes our methods for assessing the homogeneity of structural probes and for training homogeneous probes.

Section 5 presents the results of our inquiries. Section 6 concludes and proposes future directions for research.

2. Background

2.1. **Attention, Transformers, and Contextual Embeddings.**

2.1.1. *Long Range Dependencies.* In many NLP tasks, including next word prediction, translation, and question answering, model input consists of an entire sequence of input words forming a sentence or a longer document. Because traditional neural networks map a fixed size input to a fixed size output, one of the challenges in applying neural networks to NLP tasks is allowing for variable length input. Early approaches tried chopping input sequences into fixed-size subsequences called n-grams and training feed forward models on these inputs.

However, in the foundational work of Chomsky (1957), the author demonstrated that natural languages exhibit *unbounded dependencies*, in which syntactically related words might be separated by arbitrarily large distances. Consider the sentence in (1) in which the noun *whom* is the object of the verb *knew*, despite the fact that they are separated by 13 intervening words.

(1) Whom did you say that your friend thought that her father believed ... that you knew ___?

In fact, by adding more nested clauses, this separation of 13 words could be made arbitrarily large.

For this reason, neural network models trained on n-gram inputs have a hard upper bound on performance on tasks that depend on long range dependencies; they can never identify dependencies between words separated by more than n words.[citation needed] The recurrent neural network (RNN) is a model which aims to address this problem by modifying the network architecture. In one basic form of RNN called a simple recurrent network (SRN), instead of consuming the whole input all at once and producing output in a single step, the SRN works *recursively*, consuming as input one sequence token along with the hidden state produced by the last recursive step. As output, the SRN produces a new hidden state as well as an output vector. The computation can be schematized as

$$h_t = g(Uh_{t-1} + Wx_t), \qquad y_t = f(VH_t).$$

In this way, the network is able to consume arbitrarily long input sequences one token at a time, hopefully capturing long range dependencies by encoding them in the hidden state.

Problems with this approach are threefold. First of all, at each recurrent step, the entire state of the model must fit inside a hidden state vector of constant dimension. For long input sequences with many long range dependencies, there can be too much information to fit into the hidden state vector, causing decreases in performance.

A second major problem is that stochastic gradient descent (SGD), the standard optimization algorithm used to train the weights in neural network models, often fails for RNNs taking long input sequences. The SGD algorithm operates by computing the derivatives of a loss function with respect to model parameters, and updating the model parameters accordingly to minimize loss. However, training an RNN using SGD necessitates "unrolling" the recursive structure of the network in such a way that each recurrent step effectively adds another layer to the model. As the number of layers increases, the derivative of loss with respect to model parameters approaches 0, and so training slows to a crawl in what is known as the *vanishing gradients problem*. While some RNN architectures like LSTMs and GRUs have been designed with the specific goal of lessening the impact of vanishing gradients, they remain a challenge when using RNNs to identify long range dependencies.

A third significant problem with RNNs is that the computations involved are inherently serial in nature. Recent advances in computing have largely focused on increasing parallelization, but because RNN models cannot compute h_t without first computing $h_1, \ldots, h_{t-1}$, it is difficult for such models to take advantage of parallel computing resources.

2.1.2. *Transformers and Contextual Embeddings.* Transformer models aim to resolve these shortcomings of RNNs using a mechanism called *attention*. Transformer blocks take as input a sequence $(x_1, \ldots, x_n)$ and produce an output sequence of the same length $(y_1, \ldots, y_n)$. In order to produce the output y_k from the input, *self-attention heads* compare the input word x_k to all of the other inputs to identify those words which are most *relevant* to processing x_k in the given context. The transformer then runs these relevant inputs through a feedforward layer to compute the y_k.

By using layers of transformer blocks instead of recurrent connections, transformer architectures avoid the problem of vanishing gradients, and long range dependencies do not have to be compressed into a single fixed-size vector. Moreover, the inherently parallel structure of the transformer architecture is well suited to the hardware acceleration available on modern computers. **To do** (1)

These properties of transformer models allowed them to swiftly overtake several other architecture types in benchmarks for many NLP tasks. The BERT model developed in Devlin et al. (2018) was one of the first large transformer models that saw great success, and since its development, a significant research program has grown around the problem of assessing what aspects of language structure BERT actually learns. Research has suggested that BERT does encode structural syntactic information (Bacon & Regier, 2019; Jawahar et al., 2019; Warstadt et al., 2019), however the exact mechanism of this encoding remains unclear. One proposal, investigated in Auyespek et al. (2021), Chen et al. (2021), Hewitt and Manning (2019), and Reif et al. (2019), suggests that such structural data could be encoded geometrically in the contextual word embeddings generated by BERT.

2.2. **Riemannian Geometry.** Broadly speaking, geometry refers to the field of math focused on studying shapes. Most people have some degree of familiarity with the plane geometry of Euclid and its applications in calculating lengths, angles, and areas of shapes drawn in a flat plane. However, many problems related to shapes do not fit neatly into this planar Euclidean paradigm.

For instance, it is a well known result of Euclidean geometry that, given any triangle in the plane, its three interior angles always sum to a total of exactly 180 degrees. However, with a little bit of thought, it is not hard to construct examples which seem to contradict this fact. Imagine if all of the oceans on the Earth were replaced with dry land so that one could walk all over the surface of the planet. Setting out from the North Pole, a person could walk due south along the prime meridian through Greenwich until they hit the equator, then turn 90 degrees right and walk due west until arriving in the Galapagos, and then turn 90 degrees right again and continue north until arriving at the North Pole. This journey traces out a triangle on the surface of earth with three right angles, adding to a total of 270 degrees, as sketched in Figure 1.

This apparent contradiction arises due to the curvature of the Earth. As it turns out, the geometry of curved surfaces can behave quite differently than that of the flat plane, and the field of *Riemannian*

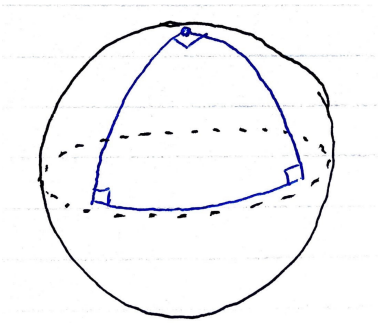

FIGURE 1. A triangle on the surface of the sphere with three right angles.

geometry gives us the tools that we need to describe and understand the geometry of curved surfaces and spaces.

2.2.1. *Topological Manifolds.* Before attempting to formally define the objects and concepts of Riemannian geometry, it is worthwhile to return to the example of the spherical surface of the Earth and consider some of its notable properties. First of all, while we have observed that the global geometry of the sphere differs from that of the flat plane, if we zoom in and restrict our attention to a small neighborhood on the surface of the sphere, we find that the *local* geometry of the sphere is practically the same as that of the plane. As relative small creatures living on a very large sphere, this is reflected in our daily life; without making observations over long distances, it is impossible to distinguish our round Earth from a flat plane.

To generalize this notion, we will introduce a few concepts from the field of *topology*. This is a subfield of geometry which studies shapes without any notion of distance or angle. For those familiar with elementary Euclidean geometry, this might sound like a strange notion: what can one say about a shape without measuring any distances? For an example, consider three shapes: a cube, a sphere, and a flat plane. If we build a cube out of elastic rubber, it is not hard to imagine "inflating" the cube, allowing its faces to puff out until the cube becomes indistinguishable from a sphere. Importantly, we can perform this deformation *continuously*, without tearing the rubber. In contrast, there is no way to continuously stretch the sphere into the shape of a flat plane without ripping a hole in its surface. Therefore, the sphere and the cube are indistinguishable to a topologist who cannot measure distances or angles, while the sphere and the plane are topologically distinct. Two shapes which cannot be distinguished by a topologist are said to be *homeomorphic*.

Applying this terminology to the example of the Earth, we have observed that even though the global topology of the Earth's surface is that of a sphere, any small neighborhood is *locally* homeomorphic to a small neighborhood of the plane. In terms of this language, we arrive at a natural definition which generalizes this notable property of the sphere.

Definition 2.1. A *surface* S is a space in which each point $p \in S$ admits some small neighborhood U which, up to homeomorphism, is indistinguishable from a small neighborhood of the Euclidean plane.

While this definition is sufficient to describe many interesting geometric objects (spheres, tori, Möbius strips, Klein bottles, etc.), it is not quite powerful enough to describe the word embeddings generated by BERT. Because surfaces look locally like a plane, they are inherently 2-dimensional objects. BERT embeddings, on the other hand, are 1024-dimensional vector spaces, so any effort to collapse them down to just 2 dimensions is unlikely to preserve enough information to tell us anything interesting about syntactic structure. In order to describe high dimensional data like word embeddings, we use the following natural generalization.

Definition 2.2. An *n-dimensional manifold* (or *n-manifold*) M is a space in which each point $p \in M$ admits a small neighborhood U which, up to homeomorphism, is indistinguishable from a small neighborhood of the n-dimensional Euclidean space $\mathbb{R}^n$.

Example 1 (The 3-dimensional torus)**.** Imagine a cube-shaped room with teleportation portals built into the walls, floor, and ceiling. If you walk into any side of the room, you emerge from the opposite side (much like the controls of the classic game Asteroids). Locally, this space is indistinguishable from $\mathbb{R}^3$, so this is an example of a 3-dimensional manifold.

One significant challenges of working with abstract topological spaces is the difficulty of describing points in the space. For instance, it is not clear how to describe a specific point on the surface of a doughnut without pointing at a picture. One of the convenient properties of manifolds is that, because they are locally indistinguishable from Euclidean space, we can put local coordinates on our manifolds, allowing us to describe points in our manifolds using numerical coordinates.

Example 2 (Stereographic coordinates)**.** As noted earlier, the global topology of the sphere is distinct from that of the plane, so there cannot be any single coordinate chart covering the entire sphere. However, if we subtract the North Pole of the sphere (which we denote by N), then we can cover the rest of the sphere with so-called *stereographic coordinates* as sketched in Figure 2. We will assume that the sphere has radius 1, and place it in $\mathbb{R}^3$ centered at $(0,0,1)$ such that the

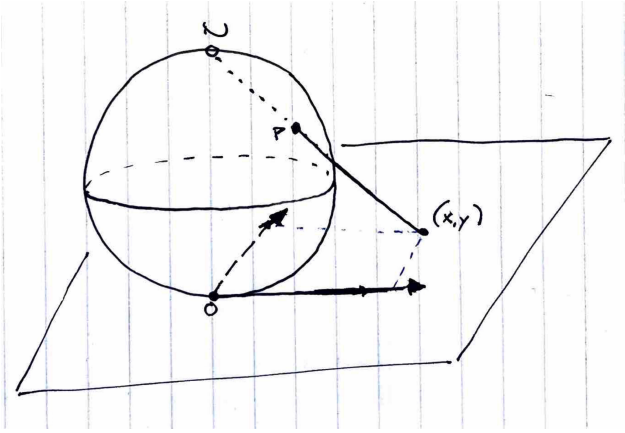

FIGURE 2. The stereographic projection of the sphere. N denotes the North Pole of the sphere, and the stereographic coordinates of p are given by (x,y).

xy-plane is tangent to the sphere at the origin. Then, for any point p on the surface of the sphere (away from N), there is a unique line passing through N and p. By reading off the xy-coordinates at which this line intersects the xy-plane, we obtain coordinates for the point p.

While coordinates provide us one tool for describing and visualizing manifolds, another useful approach is to embed manifolds inside of higher dimensional spaces. Understanding the sphere was made easier because we were able to embed the 2-dimensional surface of the sphere inside of our ordinary 3-dimensional space. In fact, it can be proven that every n-dimensional manifold can be embedded into the higher dimensional Euclidean space $\mathbb{R}^{2n+1}$. This can sometimes provide a useful perspective for thinking about manifolds in higher dimensions.

2.2.2. *Riemannian Metrics.* With these topological foundations in place, we can now return to the goal of fitting traditional geometric concepts such distances and angles into this new setting. In order to define a reasonable notion of distance on a manifold, it is worthwhile to remind ourselves of how to measure distances in ordinary Euclidean spaces. If $\gamma : [0,1] \to \mathbb{R}^n$ is a smooth curve, then

we can compute its velocity $\dot{\gamma}(t)$ by simply taking the derivative of γ at time t. Then the speed of γ at time t is simply the Euclidean norm of this velocity, defined in terms of the dot product by $\|\dot{\gamma}(t)\| = \sqrt{\dot{\gamma}(t) \cdot \dot{\gamma}(t)}$. Finally, we compute the *arclength* along γ by the integral

$$\int_0^1 \|\dot{\gamma}(t)\| dt. \tag{1}$$

Also recall that, to measure the angle θ between vectors $v, w \in \mathbb{R}^n$, we can use the identity

$$\frac{v \cdot w}{\|v\|\|w\|} = \cos\theta.$$

The key observation here is that, in order to work with distances and angles, we need (i) language to describe velocities and directions, and (ii) some notion generalizing the dot product of vectors. The following definitions will fill exactly these roles and will finally give us the tools we need to think about classical geometric concepts on manifolds.

Definition 2.3. Given a manifold M and a point $p \in M$, the *tangent space* to M at p is the vector space of all velocities with which a curve in M can pass through p.

This definition is best understood by example. Consider the 2-dimensional surface of the sphere embedded in $\mathbb{R}^3$. **To do** (2)

Next, we need to generalize the dot product. In the language of linear algebra, an *inner product* on a vector space V is a symmetric, positive definite, bilinear form on V. This means that, to every pair of vectors $v, w \in V$, the inner product is a real number denoted by $\langle v, w \rangle$ which shares the following properties with the dot product:

(i) (*Symmetry*) For any pair of vectors $v, w \in V$, $\langle v, w \rangle = \langle w, v \rangle$.

(ii) (*Linearity*) For any vectors $u, v, w \in V$ and $a \in \mathbb{R}$, $\langle u + v, w \rangle = \langle u, w \rangle + \langle v, w \rangle$ and $\langle av, w \rangle = a\langle v, w \rangle$.

(iii) (*Positive definiteness*) For any vector $v \in V$, $\langle v, v \rangle \geq 0$ with $\langle v, v \rangle = 0$ only for $v = 0$.

When a vector space is endowed with an inner product, it is possible to measure the magnitude of vectors and the angle between vectors by

$$\|v\| = \sqrt{\langle v, v\rangle} \qquad\qquad \cos\theta = \frac{\langle v, w\rangle}{\|v\|\|w\|}.$$

Definition 2.4. A *Riemannian metric* g on a manifold M is a smoothly varying choice of inner product for each tangent space T_pM.

This is the last piece of data that is required to translate all of the classical concepts of plane geometry into the non-Euclidean setting. For instance, if $\gamma\colon [0,1] \to M$ is a smooth curve, then we can define the arclength of γ to be

$$\int_0^1 \sqrt{g_{\gamma(t)}(\dot\gamma(t), \dot\gamma(t))}dt,$$

just as in the Euclidean case (c.f., Equation (1)).

Example 3 (Round spherical metric)**.** If M is the two-dimensional surface of the unit sphere, then if we throw away the prime meridian together with the North and South poles, we can put coordinates on the rest of the sphere by (θ, ϕ) where $0 < \theta < \pi$ is the azimuthal angle of a point (the angle from the point to the north pole) and $0 < \phi < 2\pi$ is the polar angle (the east/west angle away from the "prime meridian"), as sketched in Figure 3. Because the sphere can be embedded in $\mathbb{R}^3$ as the

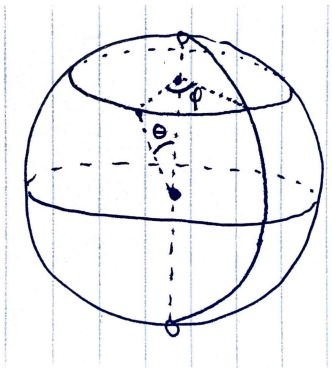

FIGURE 3. Spherical coordinates on the round sphere. The azimuthal angle is denoted by ϕ and the polar angle by θ.

unit sphere, it inherits a metric which is given in these coordinates by

$$g = \begin{bmatrix} 1 & 0 \\ 0 & \sin^2\theta \end{bmatrix}.$$

Therefore, if $\gamma : (0,\pi) \to M$ is the meridian curve given by $\gamma(t) = (t,\pi)$, then we have $\dot{\gamma}(t) = (1,0)$, and so the arclength of γ is given by

$$\int_0^\pi \sqrt{\begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \sin^2\pi \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix}}\, dt = \int_0^\pi dt = \pi,$$

which agrees with the fact that the circumference of any great circle on M is 2π.

2.2.3. *Curvature.* The final ingredient in our study of BERT's word embeddings is a discussion of curvature. Recall our earlier observation that, while triangles in the plane have internal angles summing to 180 degrees, while those on the surface of a sphere have angles summing to more than 180 degrees. This behavior is due to the fact that the sphere is curved, and by generalizing the notion of curvature to Riemannian manifolds in higher dimensions, we can learn a lot about their geometry.

Given a connected 1-dimensional manifold, often called a curve, embedded in a higher dimensional space $\mathbb{R}^n$, we can parameterize it in terms of a single real-valued variable by $r : \mathbb{R} \to \mathbb{R}^n$. Then the curvature at a point p is simply the second derivative of the curve with respect to arclength. That is, curvature describes how the direction of the curve changes as you walk along it.

In the case of 2-dimensional surfaces, there are a few different ways that one can think about curvature. At each point on a surface S, one can always identify two orthogonal geodesic curves, the so called *principle curves*, of maximal and/or minimal curvature. We obtain the intrinsic *Gaussian curvature* at p by multiplying these two *principle curvatures* together. Notably, however, we negate the Gaussian curvature if the two principle curves curve in opposite directions. For an example, consider the surfaces in **??**. On the sphere on the left, the two principal curves both curve away from us, in the direction of the sphere, so when we multiply their curvatures, the result will be a *positive* Gaussian curvature. On the cylinder, one principle curve bends away from us while the other does not bend at all; thus the Gaussian curvature is 0. Finally, on the saddle shaped surface,

one principle curve bends up and the other bends down. Since these bend in opposite directions, we say that the saddle surface has negative Gaussian curvature.

Another way of interpreting Gaussian curvature is that it measures the deviation of the area of a small circle from what we would expect in flat space. It can be proven that Gaussian curvature satisfies the equation

$$\kappa = \lim_{r \to 0^+} 12 \frac{\pi r^2 - A(r)}{\pi r^4}.$$

Thus, in spaces where κ is positive, circles have *greater* area than in flat space, and in spaces where κ is negative, circles have *less* area than in flat space. This interpretation will prove useful shortly when we discuss the use of hyperbolic spaces in neural networks.

In higher dimensions, the story of curvature becomes more complicated. There is an intricate theory allowing one to define the so-called *Riemann curvature tensor* on any Riemannian manifold. However, compared to the low dimensional cases, this tensor is more difficult to work with or interpret, so instead, we will discuss the notion of *sectional curvature*. At any point p in an n-dimensional manifold, we can choose a 2-dimensional subspace of the tangent space $\Pi \subset T_pM$. Then, we can consider the space of all geodesic curves passing through p with velocity contained in Π. Locally, these curves sweep out a 2-dimensional submanifold contained in M, an we define the sectional curvature associated with Π, denoted by $\sec(\Pi)$, to be the Gaussian curvature of this 2-dimensional surface.

2.3. **Hyperbolic Spaces.** Of particular interest in our investigation of BERT's embeddings will be Riemannian manifolds of constant, negative sectional curvature, called *hyperbolic spaces*. By constant sectional curvature, we mean that for any point $p \in M$ and any plane $\Pi \subset T_pM$, the sectional curvature $\sec(\Pi)$ is equal to some constant.

These negatively curved spaces can be difficult to visualize because, unlike familiar shapes like the sphere, no hyperbolic space admits an embedding into Euclidean 3-space. Thus, in order to visualize hyperbolic spaces, we resort to various so-called model spaces. We will be most interested in the *Poincaré ball* model of hyperbolic space. Given $c > 0$, we define the space $\mathbb{D}_c^n = \{x \in \mathbb{R}^n \mid c\|x\|^2 < 1\}$, metrized using the Riemannian metric $g_x^{\mathbb{D}} = (\lambda_x^c)^2$ where

$\lambda_x = 2/(1-c\|x\|^2)$ is called the *conformal factor*. It can be shown that the resulting space has constant sectional curvature $-c$.**To do** (3)

Recalling that, in spaces with negative curvature, circles grow *faster* than in flat space, let us consider the possible benefits of embedding syntactic data in hyperbolic spaces rather than Euclidean spaces. One of the basic assumptions of modern linguistic theory is that the structure of language is fundamentally hierarchical in nature. This is nowhere more evident than in syntax, where data is typically presented in tree-like

Note that, as the depth of a tree grows, the number of nodes in the tree grows exponentially. Contrast this with the fact that, in Euclidean space, as the radius of a ball grows, its volume only grows with polynomial speed. Because exponential growth is faster than any polynomial group, sufficiently deep trees will always fail to admit isometric embeddings into Euclidean spaces. However, in hyperbolic spaces with negative curvature, the volume of a ball grows exponentially with respect to radius, so arbitrarily deep trees can be isometrically embedded in hyperbolic space. Thus, hyperbolic spaces are often said to impart an inductive bias towards learning hierarchical representations of data, a fact which we will rely heavily upon in the following work.

3. Related Work

3.1. **Probing Language Models.**

The so called "interpretability problem" is not unique to natural language processing. Across the field of machine learning, the task of interpreting the internal representations computed in neural network models has come to be viewed as one of the fundamental challenges of the field. Probing methods have emerged as one powerful set of tools for understanding how data is encoded by neural networks, and have been applied in natural language processing to assess the extent to which neural networks' representations of language encode syntactic and morphological information (Belinkov et al., 2017; Peters et al., 2018).

Definition 3.1 (Probe)**.** Let $\mathscr{M}$ be an abstract language which maps sequences of words to sequences of vectors, so that if $s = (w_1, \ldots, w_n)$ is a sequence of words belonging to the vocabulary W, then M assigns to s a sequence $(x_1, \ldots, x_n)$, where the x_i are vectors belonging to a hidden space $\mathbb{R}^d$.

Let q be a linguistic property which can be computed for each word in a sentence. Let S denote the set of linguistically well-formed sequences $(w_1, \ldots, w_n)$, and let S_n denote the subset of sentences of length n. Then, broadly, we can think of q as a map $S_n \to U^n$, where U is some codomain measuring the relevant linguistic property.

Then a *word-level probe* for the property q on the model $\mathscr{M}$ is a map $\mathbb{R}^d \to U$ which assigns a value in U to each embedding vector. Assuming that $\mathscr{L} : U \times U \to \mathbb{R}^+$ is a loss function on U, then given a dataset $\overline{S} \subset S$, we assess the accuracy of the probe p using the loss function

$$\sum_{s \in \overline{S}} \sum_{i=1}^{|s|} \mathscr{L}(q(s)_i, p(\mathscr{M}(s)_i)).$$

For example, consider the situation in which $\mathscr{M}$ is a transformer model. Following [citation needed], we will examine the embeddings produced at a single layer of $\mathscr{M}$, so that we can treat the transformer model as a simple function of the sort described in the above definition. Suppose that we would like to understand the extent to which $\mathscr{M}$ encodes information about the part-of-speech of words. If C is a finite set of part-of-speech classes (e.g., $C = \{\mathrm{N}, \mathrm{V}, \mathrm{Adj}, \ldots\}$), then we can define U to be the set of probability distributions over C. Then $q : S \to U^{\mathbb{N}}$ maps each word in a given input sentence to the probability distribution which assigns 100% probability to the correct part-of-speech class.

A word-level part-of-speech probe for $\mathscr{M}$ would thus be a map $\mathbb{R}^d \to U$ which, given an embedding vector, outputs a probability distribution of part-of-speech classes. Using the usual negative log likelihood loss function on U, we can assess the quality of the probe across a data set.

In a simple case, the probe might consist of a single linear map $\mathbb{R}^d \to \mathbb{R}^{|C|}$ followed by a softmax function. In this case, the probe has $d|C|$ parameters (the weights of the linear map), and the loss function is differentiable with respect to these weights, so using standard gradient-descent methods, we can train the probe to optimize its performance on a training set.

Because the only input to the probe is the output of the language model (whose weights are not modified during the probe's training procedure), if the probe learns to correctly classify the part-of-speech classes of the words in an input sequence, we conclude that the relevant syntactic and/or morphological data must have been preserved by the vector embeddings. Critically, if the probe is sufficiently simple, we might have license to conclude that, not only was the relevant data

preserved by the probe, but in fact it was extracted or somehow made salient. The reasoning here is that, especially for multi-sense words, if we take a single linear layer with softmax activation and feed it inputs consisting of one-hot word encodings, there is a hard upper bound on how well it can perform part-of-speech classification. Therefore, if our probe exceeds that limit, then that increase in ability can only be due to the language model which we are probing.

This line of reasoning raises the key question of how simple a probe must be in order to conclude that its performance is due to the processing of the language model. For example, especially if we relax the constraint that the probe must act pointwise on a single word at a time (a necessary modification, as will become clear shortly), a sufficiently complex probe could perform quite well on some tasks even if the model $\mathscr{M}$ does minimal/no processing of its inputs. In such a case, it would certainly be misguided to conclude that $\mathscr{M}$ has extracted or highlighted the relevant linguistic data: all we can say is that it has preserved the relevant data (or has preserved enough information for the probe to reconstruct the relevant data).

While this assumption has been challenged by Pimentel et al. (2020), we will follow Hewitt and Liang (2019) in the assumption that simpler probes give a better metric for assessing the extent to which a language model's embeddings encode linguistic data.[**To do** (4)].

3.2. **A Geometric View of Neural Networks.** Another recent approach to interpreting neural networks, explored in detail in Hauser and Ray (2017), is to treat them as fundamentally geometric objects. In the simplest multi-layer perceptron (MLP) architectures, each layer consists of a linear transformation together with a nonlinear activation function (typical activation functions include sigmoid, tanh, or ReLU). The activation function defines a continuous deformation of the hidden embedding space, while the learned weights of the linear transformation determine the directions and scales over which this deformation is applied. In a binary classification task, for example, a multilayer perceptron might learn a continuous deformation of the embedding space such that the two classes become linearly separable.

Hewitt and Manning (2019) proposes a so-called *structural probe* for identifying syntactic data in BERT's contextual word embeddings which, as we will see, is intimately related to this geometric view of neural networks. This probe differs from the word-level probes described above in that,

rather than learning a word-level task $q : S_n \to U^n$, it learns a sentence-level task $q : S_n \to U_n$, assigning a single element of U_n to each sentence. In the particular case of Hewitt and Manning's structural probe, U_n is the space of $n \times n$ matrices, and q assigns to each sentence s the distance matrix arising from a dependency parse of s. Here, the subscript on U_n serves only to remind us that the codomain depends on the length of the input sentence.

More precisely, for any well-formed sentence s, it is assumed that there is a unique dependency parse tree which captures certain aspects of the syntactic structure of s. The vertices of this tree are exactly the words of the sentence. Then for a sentence $s = (w_1, \ldots, w_n)$, $q(s)$ is the symmetric distance matrix d whose d_{ij} entry is the number of edges along the unique path from w_i to w_j in the dependency tree of s.

Critically, the probe p is forced to be as simple as possible in the sense that it consists only of a rank k symmetric positive semi-definite quadratic form B on the embedding space. That is, $p : (\mathbb{R}^d)^n \to U$ assigns to each sequence $(x_1, \ldots, x_n)$ the $n \times n$ matrix whose (i, j)-entry is given by $B(x_i - x_j)$.

An alternative, but mathematically equivalent description of the probe can be formulated as follows. Instead of a rank k bilinear form, we simply say that the probe p consists of a $k \times d$ matrix A, and the (i, j)-entry of $p(x_1, \ldots, x_n)$ is simply the squared Euclidean distance between Ax_i and Ax_j.

Under this second perspective, the fundamentally geometric character of Hewitt and Manning's structural probe becomes clear. The dependency structure of the sentence $s \in S$ endows it with the structure of the metric space, and when the probe is trained using an L^1 loss function on $U_n = \mathrm{Mat}_{n \times n}$, the probe *almost* induces an isometric embedding of s into $\mathbb{R}^k$. We say that this embedding is almost isometric because the tree-distance in s is approximated bot by the Euclidean distance in $\mathbb{R}^k$ but rather by the squared Euclidean distance. We will call such an embedding a *squared-distance embedding*.

The reason for using the squared Euclidean distance is in fact quite simple: in general, not every tree admits an isometric embedding in $\mathbb{R}^k$. Consider, for example, the tree in **??**. In order to see that this tree does not admit an isometric embedding into Euclidean space of any dimension, we

use a simple proof by contradiction. In order to achieve such an embedding, the points x_1 and x_2 would necessarily be colinear, with x_0 their midpoint. However, the same can be said about x_1 and x_3. This forces x_2 and x_3 to coincide, contradicting the assumption that $d(x_2, x_3) = 2$.

On the other hand, it is not hard to show that every tree admits a squared-distance embedding into some Euclidean space of sufficiently high dimension. The case of trees with a single vertex is trivial, and we proceed by induction on the number of vertices. If T is a tree with n vertices, then choose one leaf vertex x and let y be the parent of x. By the inductive hypothesis, there exists a squared-distance embedding of $T \setminus \{x\}$ into $\mathbb{R}^k$ for some k. Viewing $\mathbb{R}^k$ as a subspace of $\mathbb{R}^{k+1}$, we can insert x into $\mathbb{R}^{k+1}$ a distance 1 from y in a direction orthogonal to the hyperplane containing $T \setminus \{x\}$. By the pythagorean theorem, for each vertex $z \in T \setminus \{x\}$, we have

$$\|z, x\|_{\text{Euc}}^2 = \|z, y\|_{\text{Euc}}^2 + \|y, x\|_{\text{Euc}}^2 = d(z, y) + 1,$$

where d denotes tree distance. The last equality in the above arises from the inductive hypothesis that $T \setminus \{x\}$ is mapped into $\mathbb{R}^k$ by a squared-distance embedding. By the assumption that x is a leaf and y is the parent of x, we conclude that $\|z, x\|_{\text{Euc}}^2 = d(z, x)$ for all $z \in T$, and so the entire tree is squared-distance embedded in $\mathbb{R}^{k+1}$.

In view of these two results, training a structural probe to perform squared-distance embedding of dependency parse trees into Euclidean space seems a somewhat more natural task. If the probe learns to perform the task well, then it provides compelling evidence that (a) syntactic data can be extracted from BERT's contextual word embeddings and (b) this data is encoded in the geometry of the embeddings, in the sense that some k-dimensional subspace of the embedding space admits a Euclidean metric such that squared Euclidean distance recovers the tree distance between words in the dependency parse tree.

3.3. **Hyperbolic Probes.** As discussed in Section 2.3, when it comes to embedding trees in Riemannian manifolds, some of the shortcomings of Euclidean space can be overcome in hyperbolic spaces. Recent evidence suggests that in machine learning tasks focused on hierarchical data like trees, hyperbolic representations can significantly outperform Euclidean embeddings (De Sa et al., 2018; Ganea et al., 2018a; Nickel & Kiela, 2017). In fact, Chen et al. (2021) demonstrated

a modification of the structural probe of Hewitt and Manning (2019) which results in a squared-distance embedding of dependency trees into the Poincaré ball that outperforms Euclidean probes, and Auyespek et al. (2021) used a similar technique to achieve, not a squared-distance embedding, but an honest isometric embedding into the Poincaré ball.

The squared distance probe developed in Chen et al. (2021) operates in three phases: first, like the Euclidean probe of Hewitt and Manning (2019), they train a linear map $P : \mathbb{R}^d \to \mathbb{R}^k$, where k is the rank of the probe (alternatively, the dimension of the syntactic subspace). Next, following Ganea et al. (2018b) and Mathieu et al. (2019), they identify the codomain $\mathbb{R}^k$ with $T_0\mathbb{D}_c^k$, the tangent space to the Poincaré ball at the origin, and exponentiate the resulting tangent vectors to obtain points in $\mathbb{D}_c^k$. Finally, they train a $k \times k$ matrix Q and apply Möbius matrix-vector multiplication (as defined in Section 2.3) to transform the points in the Poincaré ball. Thus, if $h \in \mathbb{R}^d$ is a contextual word embedding produced by the language model, then the probe embeds this point at $Q \otimes \exp_0(Ph) \in \mathbb{D}_c^k$.

As the authors note, the definition of Möbius multiplication is such that $Q \otimes \exp_0(Ph) = \exp_0(QPh)$, and so this final step does not in fact increase the complexity of the probe in the sense that the resulting probe is equivalent to one with only dk probe parameters. Rather, they claim that the extra step helps to stabilize the optimization process. Thus, there is a reasonable claim to be made that this squared-distance Poincaré probe is among the simplest possible maps from $\mathbb{R}^d$ to $\mathbb{D}_c^k$, in the sense that any such map must include some linearity, and this map consists only of a linear map and a canonical, geometrically motivated nonlinear function.

Auyespek et al. (2021) also use a three step method to map contextual word embeddings into the Poincaré ball. Instead of using an exponential map to project $\mathbb{R}^k$ onto $\mathbb{D}_c^k$, they use in two different trials the gnomonic map and the hyperboloid map, defined respectively by

$$g(x) = \frac{x}{\sqrt{1 + \|x\|^2}} \qquad h(x) = \frac{x}{1 + \sqrt{1 + \|x\|^2}},$$

although the motivation for choosing these maps is not explained. One consequence is that, because the Möbius matrix-vector multiplication interacts with the gnomonic and hyperboloid maps differently than with the exponential map, it cannot be simplified away, and in fact acts as a second nonlinearity in the probe. This means that the probe is not necessarily equivalent to a

probe with only kd parameters but rather rather a probe with $k(d+k)$ parameters, and it potentially allowing for more complex deformations of the geometry of the underlying contextual embeddings. While this added complexity might help to improve the performance of the probe, it potentially has the undesirable consequence of allowing the probe to extract features which are not salient in the geometry of BERT's contextual embeddings.

3.4. **Probing for Constituency.** Beyond these attempts to recover the geometry of dependency parses from BERT's contextual word embeddings, attempts have also been made to probe BERT for the types of constituency structures associated with standard syntax trees.

Tenney et al. (2019) demonstrated a probe trained to assign a non-terminal label to a known constituent in a sentence. Instead of probing the contextual embeddings arising from a single layer of BERT, the authors compare two different methods for computing contextual word embeddings which take into account information from multiple layers. The first of these, which they call the concatenation method, simply concatenates a context-independent subword token with the top layer activation of the model. The second method, called the mixing method, takes a weighted average of the context-independent embeddings and the activations from all layers, with weights learned during the training process. Once these contextual embeddings have been computed, the probe takes as input a span of several such embeddings corresponding to a known constituent in a sentence, and it predicts a probability distribution over the set of non-terminal labels. In order to handle variable-length spans as input, their probe includes a self-attention head and a pooling operator, followed by a two-layer MLP. The probes trained using the mixing method performed the best, achieving a binary F1 score of 86.7 on BERT-base and 87.0 on BERT-large.

One potential criticism of this probe is its complexity. Compared to the probes discussed above, the combination of a self-attention head together with a MLP is quite powerful, and it is at least plausible that the network is functioning as more of a parser than a probe. That is, because the probe is so complex, it is difficult to say with confidence that its success is attributable to processing done by BERT rather than by processing done by the probe. This could potentially be accounted for by training the probe on a control task in which it was only presented with non-contextual word embeddings.

More recently, Arps et al. (2022) developed a family of probes with the goal of reconstructing an entire constituency parse tree from contextual word embeddings. They use three separate probes to reconstruct partial information, together with a traditional discrete algorithm to assemble these pieces of data into a complete parse tree. The three probes, each consisting of a simple linear classifier trained on contextual word embeddings, predict (a) the label of the least common ancestor of a pair of adjacent tokens, (b) the relative depths of the least common ancestor of a word with its neighbors immediately to the left and the right, and (c) the label of single-word constituents.**To do** (5) The resulting probes achieved a labeled F1 score of 80.42.

Like the Tenney et al. (2019) probe, this probe takes as input a combination of the activations from multiple layers of BERT. While this investigation does suggest that BERT's contextual word embeddings do expose syntactic data that can be recovered by simple probes, the discrete nature of the tree-reconstruction algorithm means that very little can be said about how well the geometry of the embeddings reflects the broader syntactic structure.

4. Methods

4.1. **Tree Reconstruction.** Following Chen et al. (2021) and Hewitt and Manning (2019), we train a family of probes which map contextual embeddings into Riemannian manifolds while preserving information about the tree-distance between words. In contrast to the dependency parses used in these earlier works, our probe is trained to reconstruct the tree distance in a dependency parse of a sentence. Along the lines of Auyespek et al. (2021), our probes are trained to perform isometric embeddings rather than squared-distance embeddings. In fitting with the geometric framework described by Hauser and Ray (2017), we treat each layer of BERT as a geometric deformation of the embedding space. Therefore, rather than training probes on some combination of multiple layers, we train our probes on a single layer at a time to understand how the geometry is deformed layer-by-layer.

If $s = (w_1, \ldots, w_n)$ is an input sentence consisting of n words, denote the contextual word embedding for w_i by $x_i \in \mathbb{R}^d$. Following Hewitt and Manning (2019), our Euclidean probes consist

simply of a linear transformation $p : \mathbb{R}^d \to \mathbb{R}^k$ trained on the loss function

$$\sum_{i,j=1}^{n} \Big| d(w_i, w_j) - \|p(x_i - x_j)\| \Big|,$$

where d is the tree-distance between w_i and w_j in a constituency parse of s. As discussed in Section 3.2 this probe structure is equivalent to training a rank-k quadratic form B on $\mathbb{R}^d$ with the loss function

$$\sum_{i,j=1}^{n} \left| d(w_i, w_j) - \sqrt{B(x_i - x_j)} \right|.$$

Our hyperbolic probe will use the structure described in Chen et al. (2021). The structure is effectively the same as that of the Euclidean probe, but we identify the embedding space $\mathbb{R}^k$ with the tangent space to $\mathbb{D}_c^k$ at the origin, and map the embeddings into $\mathbb{D}_c^k$ by an exponential map. The training objective is

$$\sum_{i,j=1}^{n} \Big| d(w_i, w_j) - d_{\mathbb{D}}(\exp_0(Bx_i), \exp_0(Bx_j)) \Big|.$$

Once such a probe has been trained, we can apply standard hierarchical clustering techniques to construct a full binary parse tree. In general, we will use a recursive algorithm in which we build a tree by recursively merging the closest pair of constituents. After each step, there are several choices to measure the distance a newly formed constituent to each other constituent in the tree. These include:

i) The average of the distance to each of the children,
ii) The distance to the nearest child,
iii) The distance to the furthest child,
iv) The distance to the midpoint along the geodesic connecting the two children,
v) One less than the average/minimum/maximum distance to the children.

All of these choices are dependent directly upon the geometry of the Riemannian manifold. We assess the quality of the resulting parse trees using the PARSEVAL metric of Black et al. (1991).

One challenge in reconstructing constituency parses based on distance matrices is the fact that distance matrices are highly sensitive to assumptions about the grammar. One approach to stabilizing the training process is to bias the probe towards reconstructing the gold parse tree. We achieve this

by adding a second loss function. For each sentence, this second function computes the sequence of merges which would have to occur to reconstruct that sentence, and assigns as loss the sum of the distances between all of the merged pairs. In minimizing this loss, the probe is biased towards reconstructing the correct parse tree.

4.2. **Data.** One of significant challenge in training structural probes is the necessity for a large quantity of labeled linguistic data. Following Hewitt and Manning (2019), we train our probes on the Penn Treebank (Marcus et al., 1993), which contains sentences from 2,499 Wall Street Journal articles labeled with constituency trees.

5. Results

Report the results of the experiments described in the previous section. This will include quantitative data presented in tables, and also qualitative data including examples of trees predicted by our hyperbolic probe.

6. Conclusion

Interpret results in the broader context established in the introduction. Detail directions for future inquiry.

References

Arps, D., Samih, Y., Kallmeyer, L., & Sajjad, H. (2022). Probing for constituency structure in neural language models. *Findings of the Association for Computational Linguistics: EMNLP 2022*, 6738–6757.

Auyespek, T., Mach, T., & Assylbekov, Z. Hyperbolic embedding for finding syntax in BERT. In: *CEUR workshop proceedings*. *3078*. 2021, 58–64.

Bacon, G., & Regier, T. (2019). Does BERT agree? Evaluating knowledge of structure dependence through agreement relations. https://doi.org/10.48550/ARXIV.1908.09892

Belinkov, Y., Durrani, N., Dalvi, F., Sajjad, H., & Glass, J. (2017). What do neural machine translation models learn about morphology? *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 861–872. https://doi.org/10.18653/v1/P17-1080

Black, E., Abney, S., Flickenger, D., Gdaniec, C., Grishman, R., Harrison, P., Hindle, D., Ingria, R., Jelinek, F., Klavans, J., Liberman, M., Marcus, M., Roukos, S., Santorini, B., & Strzalkowski, T. (1991). A procedure for quantitatively comparing the syntactic coverage of English grammars. *Speech and Natural Language: Proceedings of a Workshop Held at Pacific Grove, California, February 19-22, 1991*.

Chen, B., Fu, Y., Xu, G., Xie, P., Tan, C., Chen, M., & Jing, L. (2021). Probing BERT in hyperbolic spaces.

Chomsky, N. (1957). Syntactic structures. *Syntactic structures*. de Gruyter.

De Sa, C., Gu, A., Ré, C., & Sala, F. (2018). Representation tradeoffs for hyperbolic embeddings. https://doi.org/10.48550/ARXIV.1804.03329

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. https://doi.org/10.48550/ARXIV.1810.04805

Ganea, O., Becigneul, G., & Hofmann, T. (2018a). Hyperbolic entailment cones for learning hierarchical embeddings. In J. Dy & A. Krause (Eds.), *Proceedings of the 35th international conference on machine learning* (pp. 1646–1655). PMLR.

Ganea, O., Becigneul, G., & Hofmann, T. (2018b). Hyperbolic neural networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in neural information processing systems*. Curran Associates, Inc.

Hauser, M., & Ray, A. (2017). Principles of Riemannian geometry in neural networks. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in neural information processing systems*. Curran Associates, Inc.

Hewitt, J., & Liang, P. (2019). Designing and interpreting probes with control tasks. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the*

9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), 2733–2743. https://doi.org/10.18653/v1/D19-1275

Hewitt, J., & Manning, C. D. (2019). A structural probe for finding syntax in word representations. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4129–4138. https://doi.org/10.18653/v1/N19-1419

Jawahar, G., Sagot, B., & Seddah, D. (2019). What does BERT learn about the structure of language? *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 3651–3657. https://doi.org/10.18653/v1/P19-1356

Lee, J. M. (2013). *Introduction to smooth manifolds*. Springer.

Lee, J. M. (2018). *Introduction to Riemannian manifolds*. Springer.

Marcus, M. P., Marcinkiewicz, M. A., & Santorini, B. (1993). Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, *19*(2), 313–330.

Mathieu, E., Le Lan, C., Maddison, C. J., Tomioka, R., & Teh, Y. W. (2019). Continuous hierarchical representations with poincaré variational auto-encoders. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems*. Curran Associates, Inc.

Nickel, M., & Kiela, D. (2017). Poincaré embeddings for learning hierarchical representations. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in neural information processing systems*. Curran Associates, Inc.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2227–2237. https://doi.org/10.18653/v1/N18-1202

Pimentel, T., Valvoda, J., Maudslay, R. H., Zmigrod, R., Williams, A., & Cotterell, R. (2020). Information-theoretic probing for linguistic structure. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 4609–4622. https://doi.org/10.18653/v1/2020.acl-main.420

Reif, E., Yuan, A., Wattenberg, M., Viegas, F. B., Coenen, A., Pearce, A., & Kim, B. (2019). Visualizing and measuring the geometry of BERT. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems*. Curran Associates, Inc.

Tenney, I., Xia, P., Chen, B., Wang, A., Poliak, A., McCoy, R. T., Kim, N., Van Durme, B., Bowman, S. R., Das, D., & Pavlick, E. (2019). What do you learn from context? probing for sentence structure in contextualized word representations. https://doi.org/10.48550/ARXIV.1905.06316

Warstadt, A., Cao, Y., Grosu, I., Peng, W., Blix, H., Nie, Y., Alsop, A., Bordia, S., Liu, H., Parrish, A., Wang, S.-F., Phang, J., Mohananey, A., Htut, P. M., Jeretic, P., & Bowman, S. R. (2019). Investigating BERT's knowledge of language: Five analysis methods with NPIs. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2877–2887. https://doi.org/10.18653/v1/D19-1286

TO DO...

- ☐ 1 (p. 6): I think there is some problem where the number of parameters or the computation cost of the attention grows as $O(n^2)$ in a problematic way. Figure this out.
- ☐ 2 (p. 10): Describe the tangent space of the sphere. Also explain that the derivative of a curve is a velocity vector.
- ☐ 3 (p. 14): Demonstrate what the geodesics and isometries of the Poincaré ball look like.
- ☐ 4 (p. 16): There are reasons that I disagree with the analysis in Pimentel et al. (2020)
- ☐ 5 (p. 21): This task seems a bit odd... maybe look at their code to ensure I am understanding correctly.