Geometric Properties of Unsupervised Representations in Natural Language Processing: A Review

Michael Hu
Princeton University
myhu@princeton.edu

Abstract

Representation learning is a key step in natural language processing (NLP). Discovering good features for downstream tasks from unlabeled text is a core component of many successful methods in NLP, from word2vec to BERT. The embeddings derived from these methods often have interesting geometric properties. In this review, we recap several popular representation learning methods in the NLP field and discuss the geometric properties of each, as well as their limitations.

1 Introduction

Arora et al. (2019) define *unsupervised representation learning* as "using unlabeled data to learn a representation function f such that replacing data point x by feature vector $f(x)$ in new classification tasks reduces the requirement for labeled data." In the case of NLP, data points are words, and feature vectors are embeddings. Embeddings represent the words themselves, and we consider embeddings to be good if they boost performance on downstream tasks.

Here, we focus on four of the most successful representation learning techniques in NLP: word2vec, GloVe, ELMo, and BERT (Mikolov et al., 2013b; Pennington et al., 2014; Peters et al., 2017; Devlin et al., 2018). word2vec and GloVe represent the sub-topic of word embeddings, and ELMo and BERT represent the sub-topic of contextual word embeddings. We give a short exposition of each and explain their known geometric properties and limitations. For word embeddings, we dive into the parallelogram model of analogy, and ultimately identify its shortcomings. Central to the analysis of contextual word embeddings is the probing technique, where a simple classifier is used to interpret what a larger representation has learned. We discuss how probing uncovers evidence for syntax tree embeddings in the representations of ELMo and BERT. We close by studying a new direction in word embeddings related to hyperbolic geometry.

2 Background

Consider a vector x with its location defined as an n-tuple $(x_1, x_2, ..., x_n)$. The inner product between two vectors in Euclidean space is the dot product:

$$\langle x, y \rangle := x_1 y_1 + ... + x_n y_n = x^T y$$

Euclidean space, or $\mathbb{R}^d$, is simply all possible combinations of real-valued n-tuples, with the inner product as defined above. Euclidean space possesses a distance metric. A distance metric d must follow 3 main properties:

- **Minimality:** $d(x, y) \leq d(x, x) = 0$
- **Symmetry:** $d(x, y) = d(y, x)$
- **Triangle inequality:** $d(x, y) + d(y, z) \geq d(x, z)$

3 Word Embeddings

3.1 Word2Vec

Distributed representation was first described in detail by Rumelhart et al. (1986). In distributed representations, a concept like a word or a pixel is ingested and operated upon by many different neurons in a neural network. Thus, an individual word or pixel is represented in the outputs of many different neurons – hence, *distributed*. Mikolov et. al apply the idea of distributed representation in word2vec, a popular word embedding method published in 2013 (Mikolov et al., 2013b).

Word2vec broke with past work on embeddings (Collobert and Weston, 2008; Huang et al., 2012) by using the skip-gram with negative sampling (SGNS) model (Mikolov et al., 2013a), which reduces the problem of learning word embeddings

into a logistic regression problem. The logistic regression problem is: label shallow window word pairs from the corpus with 1, and label word pairs made at random with 0. "Shallow window" means that the words in the word pair must be within a few words of each other. Negative sampling refers to random generation of 0-labeled samples.

We now examine the objective function of word2vec. Let w_a, w_b refer to word a and word b, and $\boldsymbol{w_a}, \boldsymbol{w_b}$ refer to the feature vectors of these words. w_a and w_b are within a shallow window of each other. The word2vec objective is then:

$$\log \sigma(\boldsymbol{w_a}^T \boldsymbol{w_b}) + \sum_{i=1}^{k} \log \sigma(-\boldsymbol{w_i}^T \boldsymbol{w_b})$$

where $\boldsymbol{w_{i=1}}, ..., \boldsymbol{w_{i=k}}$ are feature vectors of words drawn at random from the corpus, and σ is the logistic function. This objective is maximized when $\sigma(\boldsymbol{w_a}^T \boldsymbol{w_b}) = 1$ and $\boldsymbol{w_i}^T \boldsymbol{w_b} = 0, \forall w_i$. So at best, $\boldsymbol{w_a}^T \boldsymbol{w_b} >> 0$, and $\boldsymbol{w_i}^T \boldsymbol{w_b} << 0$.

The above objective holds that closeness between words is defined by their dot product. Words that appear together often should have feature vectors with a large dot product, and vice versa.

3.2 Geometric properties of word2vec

Mikolov et al. (2013b) identified that the word representations learned by word2vec are additively compositional, meaning that adding two word vectors element-wise produces a new word vector that approximately captures the meaning of both words. They give the example that adding vectors representing "Russian" and "river" produces a new vector similar to that of "Volga River," a river flowing through central Russia.

Word embedding methods like word2vec and GloVe also demonstrate the parallelogram model of analogy, first discovered by Rumelhart and Abrahamson (1973). The parallelogram model observes that analogies tend to exhibit linear structure when visualized in feature space. The classical example of the parallelogram model is *man:woman::king:??*. Mikolov et al. (2013a) solve this analogy by doing simple addition and subtraction on the word2vec vectors: $v_{king} - v_{man} + v_{woman} = v_{mystery}$, and v_{queen} is the vector closest to $v_{mystery}$.

To put the parallelogram model formally, analogies of form *a:b::c:??* can be solved by the expression:

$$\arg\min_d \|v_a - v_b - v_c + v_d\|^2$$

It is curious that word2vec embeddings capture relationships between words as lines, as such a property is not enforced in the training objective. Recall that the SGNS objective simply maximizes the inner products of feature vectors for words that appear closely together. Arora et al. (2015) explain this observation by first defining RAND-WALK, their own word embedding model with connections to word2vec and GloVe. They then assume that RAND-WALK feature vectors are distributed isotropically, or according to a spherical Gaussian in representation space. They support this observation with empirical results on RAND-WALK. Under this assumption, the following relation holds:

$$\log \frac{p(w|w')}{p(w)} = \frac{\langle w, w' \rangle}{d} \pm O(\epsilon) \qquad (1)$$

where d is the dimension of the embedding.

Arora et. al then provide an explanation for the linear structure of analogies like *man:woman::king:queen*. In these types of analogies, the following is true:

$$\frac{p(x|\text{king})}{p(x|\text{queen})} \approx \frac{p(x|\text{man})}{p(x|\text{woman})} = v_R(x) \cdot \text{noise} \qquad (2)$$

R is the relation common to *king:queen* and *man:woman*. To see why these ratios are approximately equal, consider when x is a word related to men. Then $p(x|\text{king}) \approx p(x|\text{man}) \approx 1$, and $p(x|\text{queen}) \approx p(x|\text{woman}) \approx 0$. So both fractions would be large.

Let's just consider *king:queen*. By (1):

$$v_R(x) \cdot \text{noise} = \frac{p(x|\text{king})}{p(x|\text{queen})}$$

$$\Rightarrow \quad \log v_R(x) + \text{noise} = \frac{\langle x, (v_{king} - v_{queen}) \rangle}{d}$$

$$\Rightarrow \quad \log v_R + \text{noise} = \boldsymbol{V} \frac{(v_{king} - v_{queen})}{d}$$

The last line is the previous line written in vector form. We notice that $v_{king} - v_{queen}$ is the solution to a linear regression problem in the form $y = \boldsymbol{W}x$! So under the isotropic assumption, $v_{king} - v_{queen}$ is the best linear approximation of the relationship captured by v_R. To generalize:

adding and subtracting word vectors produces linear approximations of the relationships between the words themselves.

However, it is important to note that Arora et al's results only hold under the assumption of isotropy. So for word embeddings that are *anisotropic*, or *not isotropic*, exactly why the parallelogram model holds is debated.

3.3 GloVe

In contrast with word2vec, GloVe (**Glo**bal **Ve**ctors) was designed to enforce the parallelogram model of analogy. We begin again from (2), fixing the notation slightly to match that of Pennington et al. (2014):

$$\frac{p(k|i)}{p(k|j)} = F(w_i, w_j, w_k)$$

If we assume a linear structure between feature vectors w_i and w_j, we can simply subtract the two with no loss of information. We further require F to be a homomorphism between vector differences and probability ratios. That is to say, we wish to preserve the linear structure we just assumed and imposed via subtraction.

Under these two assumptions, we obtain the relation

$$\log p(k|i) = \langle w_i, w_k \rangle$$

If X_{ik} is the number of times w_k occurs in the context of w_i in a corpus, and X_i is the number of times w_i occurs, then $p(k|i) = \frac{X_{ik}}{X_i}$. We apply this and fit the inner product $\langle w_i, w_k \rangle$ using weighted linear regression. This brings us to the objective function, where k is replaced with j:

$$J = \sum_{i,j}^{V} f(X_{ij})(\langle w_i, w_j \rangle + b_i + b_j - \log(X_{ij})^2)$$

Thus, we see that GloVe makes more explicit assumptions about the linearity of word vectors. These assumptions influence the loss function. Both word2vec and GloVe manipulate the inner products between word vectors to fit certain properties, whether it be a binary label (word2vec) or global co-occurrence statistics (GloVe).

Unsurprisingly, GloVe embeddings also exhibit additive compositionality and the parallelogram model of analogy (Pennington et al., 2014). These properties are enforced by the objective function.

3.4 Geometric Limitations of Word Embeddings

Aside from the previously discussed favorable geometric features, static word embeddings like word2vec and GloVe also have significant geometric limitations. A limitation specific to SGNS systems like word2vec is that the resultant word vectors tend to reflect hyperparameter choices in negative sampling. Mimno and Thompson (2017) found that the average dot product between word vectors and the mean word vector increases as the number of negative samples increases. These findings suggest that as k for negative sampling increases, all word vectors point closer and closer to the mean word vector. Ideally, word embedding methods should not be highly sensitive to hyperparameter choice, as hyperparameters are unrelated to word meaning. The authors do not give an explanation for this phenomenon.

This finding also implies that SGNS vectors are anisotropic: they are not distributed in feature space as a spherical Gaussian, as assumed by Arora et al. (2015). Rather, the vectors are distributed in a narrow cone in the first orthant, around the mean word vector. Thus, Arora et al's assumption may be too strong to generalize to word2vec. A canonical explanation for the parallelogram model is still under debate (Allen and Hospedales, 2019; Ethayarajh et al., 2019; Gittens et al., 2017).

At a more generic level, word embedding methods transform words into vectors in $\mathbb{R}^d$ by enforcing some heuristic on the inner product. This is not a new concept: cognitive scientists have explored embedding concepts into metric spaces like $\mathbb{R}^d$ for decades! Thus, applying teachings from cognitive science to the analysis of word embeddings gives us good intuition as to how word embeddings are limited.

3.4.1 Metric Critiques

The central critique of cognitive models based on metric spaces is that human cognition often violates metric axioms. In 1977, Amos Tversky criticized geometric models of similarity that used distance metrics, noting that many human concepts violate the symmetry property of metric spaces (Tversky, 1977). (See Background.) For example, we say "like father, like son," but not "like son, like father." The implication here is that the son is more like the father, than the father is like the son. So a perfect similarity metric would encode that

$sim(\text{son}, \text{father}) > sim(\text{father}, \text{son})$. But distance metrics are symmetric by definition, so they cannot encode these complexities. Thus, any model of similarity depending on a distance metric is doomed to fail with regard to symmetry.

Human concepts, words especially, also violate the triangle inequality. In terms of word associations, the triangle inequality implies if associations $a \rightarrow b$ and $b \rightarrow c$ are high, then the association $a \rightarrow c$ must also be high. Griffiths et al. (2007) give the following counterexample to the triangle inequality: *asteroid* is strongly associated with *belt*, *belt* is strongly associated with *buckle*, but *belt* and *buckle* have no association with each other. Thus, tasks like word association also cannot be solved using a metric-based model of similarity.

3.4.2 Applying Metric Critiques to Word Embeddings

Neither word2vec nor GloVe strictly define a similarity metric between vectors (although cosine similarity is used as a similarity metric informally). However, both are firmly committed to the parallelogram model of analogy. In fact, Mikolov et al. (2013c) go so far as to define the parallelogram model as part of the evaluation scheme for word embeddings.

Chen et al. (2017) evaluate the parallelogram model of analogy in the context of metric criticisms from Tversky and Griffiths and find that word2vec and GloVe suffer from similar issues as metric models of similarity. Namely, word2vec and GloVe give solutions incompatible with human judgment when queried with analogies relating to symmetry and the triangle inequality.

With regards to symmetry for analogies, note that some analogies are asymmetrical. For example, consider the analogy *angry:smile::exhausted:run*. The relationships are clearly antithetical. However, when we switch the order to create *exhausted:run::angry:smile*, the new relationship is more ambiguous. One might think: "running makes you tired, but smiling doesn't make you angry." This analogy is asymmetrical, and Chen et al. (2017) showed that people perform worse on asymmetrical analogies stated in reverse. The relation in the analogy is clearer when stated in a certain way. But word2vec and GloVe achieve the same performance, because the underlying parallelogram in feature space is the same. Word2vec and GloVe see a strong connection between words when one doesn't actually exist.

Chen et. al also show that certain analogy pairs violate the triangle inequality. For example, *nurse:patient::mother:baby* and *mother:baby::frog:tadpole* are both analogies that make sense. But *nurse:patient::frog:tadpole* does not make sense. Human performance suffered on these triangle-inequality-violating analogies, but the performance of word2vec and GloVe actually improved.

The experiments of Chen et. al demonstrate that word2vec and GloVe embeddings imperfectly render the relationships between words. Rather, these embeddings break down in ways reminiscent of metric models of similarity.

4 Contextual Word Embeddings

Outside of such geometric critiques, there are simpler arguments for why static word embeddings are insufficient. For one, a word may have multiple homonyms, or different meanings under different contexts. Nevertheless, word2vec and GloVe assign all homonyms the same vector representation, regardless of context. This motivates our discussion of *contextual* word embeddings.

4.1 Sesame Street: ELMo and BERT

The most popular contextual word embedding system is ELMo (**E**mbeddings from **L**anguage **Mo**dels) (Peters et al., 2018). In ELMo, a 2-layer forward LSTM is trained to predict the next word in the sentence, and a 2-layer backward LSTM is trained to predict the previous word. Predicting the next word is the classic language modeling objective. To generate such a prediction, the output of the LSTM is fed into a softmax.

The main idea of ELMo is that the hidden states of the LSTMs can be used as a contextual word embedding. To create an embedding, Peters et al. (2018) concatenate the hidden states of the forward and backward LSTMs at each layer. They then compute a linear combination of the layers to arrive at a contextual word embedding.

Not long after ELMo came the wave of model architectures based on the Transformer, created by Vaswani et al. (2017). These deep models were designed to be trained end-to-end from text and do not require consuming the embeddings of another model. In effect, these Transformer-based architectures learn embeddings on the fly. The most popular such architecture is BERT (**B**idirectional

Encoder **R**epresentations from **T**ransformers) (Devlin et al., 2018).

Nevertheless, BERT still falls under the category of unsupervised representation learning due to its *pretrain, fine-tune* training split (Radford et al., 2018). In unsupervised pretraining, a model like BERT is trained on a large corpus of unlabeled text. For BERT, the objective during unsupervised pretraining is the masked language model (MLM) objective, where BERT predicts the identity of words that have been masked out. (Basically, BERT fills in the blanks.) The goal during unsupervised pretraining is to learn good representations for the downstream task. The downstream task is then optimized during discriminative fine-tuning.

Thus, we can consider the representations learned by BERT after pretraining as a type of contextual word embedding.

4.2 Probing

Analysis of contextual word embeddings is tricky, as the vector representation of any given word is variable. By virtue of being contextual, each embedding now encodes some knowledge about the entire sentence as well. Futhermore, LSTMs and Transformers are orders of magnitude more complex in terms of parameters compared to the word2vec and GloVe. Thus, it can be difficult to understand exactly what deep models like ELMo and BERT are learning.

Enter probing. Probing is a technique designed to help interpret the information encoded by deep representation. The intuition is this: feed the embeddings (i.e. the internal representation) of a model like BERT or ELMo into a very simple classifier, and train the classifier on some task requiring some specific linguistic understanding. For example, performing well on part-of-speech (POS) tagging would require a model to learn parts of speech, or syntax. If this simple classifier can correctly predict POS, then deep representation must encode knowledge about POS as well.

Formally, let $x_{1:T} = x_1, x_2, ..., x_T$ represent a sentence. Pass the sentence through ELMo or BERT to obtain vector representations of each word: $h_{1:T}$. If a classification task is to assign labels $y_{1:T}$ to the words $x_{1:T}$, then the probe f_θ's job is to produce predictions $\widehat{y_{1:T}} = f_\theta(h_{1:T})$ by consuming the vector representations $h_{1:T}$ instead of the words themselves. If the predictions $\widehat{y_{1:T}}$ are highly accurate, we take this to mean the representations $h_{1:T}$ reflect linguistic knowledge. (Notation from Hewitt and Liang (2019).)

Probing resonates well with Arora et. al's definition of *unsupervised representation learning*, quoted at the beginning of this review. Here, probing is the new classification task. A good feature vector $f(x)$ from ELMo or BERT should be amenable for classification by the probe.

4.3 Recovering Geometry from Deep Representations

Hewitt and Manning (2019) use probing to demonstrate that ELMo and BERT encode syntax trees in their representations. Hewitt and Manning's *structural probe* learns a linear transformation that reproduces parse tree distance between two words from the deep representation of these words. In practice, this linear transformation can be low-rank. Increasing the rank of the underlying transformation matrix above 128 has little effect on the accuracy of the predicted distance.

We can think of the structural probe as learning a low-rank linear filter. Let h_i, h_j be the deep representations or contextual word embeddings corresponding to two words w_i, w_j. The learned linear transformation is then a matrix B. This B matrix transforms the vectors h_i, h_j into a lower-rank Euclidean space. We then take the Euclidean distance between Bh_i and Bh_j. Hewitt and Manning find that the squared Euclidean distance between Bh_i and Bh_j precisely corresponds to the tree distance between words w_i and w_j in the ground-truth parse tree.

Other papers certainly establish that ELMo and BERT learn syntax (Liu et al., 2019; Kovaleva et al., 2019), but (Hewitt and Manning, 2019) was the first paper to concretely demonstrate how syntactical knowledge is encoded in contextual representations.

4.4 Why Squared Distance?

ELMo and BERT embedding syntax trees via squared Euclidean distance is a curious result, and Coenen et al. (2019) attempt to explain it. First, Coenen et. al prove that the only trees embeddable into Euclidean space isometrically, or with out distortion of tree lengths, are chains. Chains are what they sound like: a straight path of nodes from parent to child. Since most syntax trees are not chains. the Euclidean metric is fundamentally unsuitable for embedding syntax trees. However, Coenen et.

al also show that any tree of n nodes can be embedded into $\mathbb{R}^{n-1}$ with arbitrarily low distortion. They call these embeddings *Pythagorean embeddings*. Thus, Hewitt and Manning (2019) found evidence that ELMo and BERT learn Pythagorean embeddings of syntax.

Why ELMo and BERT learn Pythagorean embeddings specifically is an open question. There certainly exist other embedding schemes: we explore hyperbolic embeddings in the next section. Coenen et. al hypothesize that Pythagorean embeddings are amenable for ELMo and BERT to learn because the embedding scheme is simple.

4.5 Geometric Properties of Contextual Word Embeddings

Ethayarajh (2019) discovered that contextual word embeddings from ELMo, BERT, and GPT-2 (another Transformer-based language model) are all anisotropic. Curiously, Mu et al. (2017) demonstrate that increasing isotropy helps the performance of static word embeddings. Given that contextual word embeddings tend to outperform static word embeddings on downstream tasks (Peters et al., 2018), it is unclear how these two discoveries interact.

As a sanity check, Ethayarajh demonstrates that contextual word embeddings are not identical to each other. Furthermore, on average, less than 5% of the variance of contextual word embeddings can be explained by their first principal component. This is concrete evidence that static word embeddings would be a poor substitute for contextual word embeddings. Summarizing the point cloud of contextual word embeddings down to a single vector does not produce a good representative.

5 What's the Best Way to Embed a Tree?

Aside from Pythagorean embeddings, to which ELMo and BERT's syntax tree embeddings seem to be related, there exist tree embedding methods that preserve distances between tree nodes with very low distortion. These embedding methods require hyperbolic geometry.

5.1 Hyperbolic Geometry and the Poincaré Disk (Intuition)

Here, we simply wish to give some intuition as to why embedding into hyperbolic space makes sense for trees. Concretely, hyperbolic geometry is the way points and lines behave on a surface of a hyperboloid.

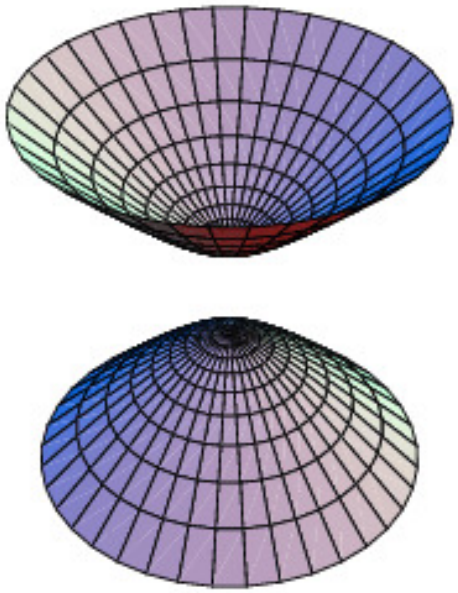

Figure 1: Example of a hyperboloid. From Weisstein.

Instead of embedding into a space that is flat, or Euclidean, we embed into hyperbolic space, a space that curves away from the origin. As a thought experiment, consider cutting the hyperboloid into slices of equal height. The first slice is a cup, and the rest are successively larger onion rings. Given that each ring is larger the farther it is from the origin, we can embed more data on rings farther from the origin while keeping data density the same. This is good for embedding trees, as trees branch and grow exponentially from the root node. We give a more formal example of this reasoning in the context of Poincaré disks.

The Poincaré disk is the hyperboloid projected onto the unit circle. It's a way to visualize 2D hyperbolic space. The edge of the circle is then the hyperboloid extending out to infinity.

The following example is adapted from Sa et al. (2018). Apriori, we note that the shortest path between two tree nodes runs through its least common ancestor (Condition 1). The hyperbolic distance metric defined on the Poincaré disk is:

$$d_H(x,y) = acosh\left(1 + \frac{2\left\|x-y\right\|^2}{(1-\left\|x\right\|^2)(1-\left\|y\right\|^2)}\right)$$

The Euclidean distance metric is the same:

$$d_E(x,y) = \left\|x-y\right\|$$

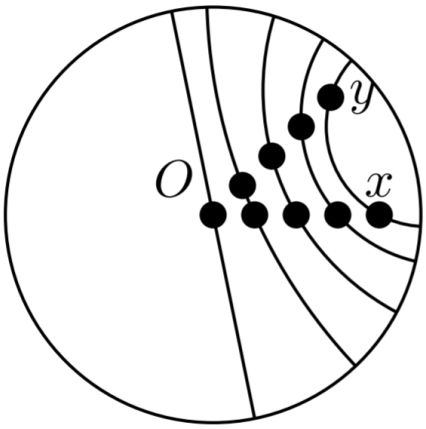

Figure 2: Poincaré disk. Lines are the shortest paths between x and y in hyperbolic space. From Sa et al. (2018).

We now use $f_{(.)}(x,y) = \frac{d_{(.)}(x,y)}{d_{(.)}(x,0)+d_{(.)}(0,y)}$ as a measure of how suitable each distance metric is for embedding a tree. f measures how closely a shortest path runs to the origin. When $f = 1$, the shortest path between x and y passes through the origin. Intuitively, we want a space with distance metric such that $f \approx 1$. This means that shortest paths curve towards the origin, which in turn allows us to embed ancestor nodes closer to the origin, thereby satisfying the apriori (Condition 1).

We now consider the distance metrics. As $\|x\|, \|y\| = t \to 1$, $f_H(x,y) \to 1$ (Sa et al., 2018). But $f_E(x,y)$ remains constant with respect to t:

$$\begin{aligned} f_E(x,y) &= \frac{d_E(x,y)}{d_E(x,0)+d_E(0,y)} \\ &= \frac{\sqrt{x^Tx - 2x^Ty + y^Ty}}{2t} \\ &= \frac{t\sqrt{1 + 2x^Ty/x^Tx + y^Ty/x^Tx}}{2t} \\ &= \frac{1}{2}\sqrt{2 + 2\cos(\theta)} \end{aligned}$$

This means that by embedding points closer to the edge of the Poincaré disk, we can adjust how closely the shortest path in hyperbolic space runs to the origin. But we cannot with Euclidean space. Note that squared Euclidean space suffers from the same issue: simply square the numerator in the manipulations above.

5.2 When to Use Hyperbolic Space

Aside from intuitive arguments, we have stronger guarantees for hyperbolic space outperforming Euclidean space in terms of tree embeddings. Let n be the number of nodes in a tree. Sarkar (2012) developed an algorithm for embedding trees into the Poincaré disk with $O(1+\epsilon)$ distortion. In other words, using hyperbolic embeddings allows one to perfectly embed a tree using only 2 dimensions. (Distortion measures the difference between graph distances and embedding distances.)

Conversely, any d-regular tree embedding into Euclidean space is guaranteed to suffer $\Omega(\log n)$ distortion, for any dimension size (Linial et al., 1994). Pythagorean embeddings can be used to avoid this lower bound, but recall that the dimensionality required to embed trees isometrically using Pythagorean embeddings is $O(n)$.

This result can be taken as theoretical evidence that BERT does not scale beyond the sentence. If each additional word in a sentence requires an additional dimension for isometric Pythagorean embeddings, then eventually BERT would run out of representative capacity for its syntax tree representation. So BERT has an upper limit for sentence length. (This is also trivially true, as BERT has a maximum input size.) Thus, teaching BERT to represent syntax hyperbolically may be a way to increase its capacity.

5.3 Hyperbolic Word Embeddings

Hyperbolic geometry was recently applied to static word embeddings. Tifrea et al. (2018) developed a method for embedding words into the Poincaré ball by adapting the GloVe training objective. Recall that the GloVe training objective is:

$$J = \sum_{i,j}^{V} f(X_{ij})(\langle \boldsymbol{w_i}, \boldsymbol{w_j} \rangle + b_i + b_j - \log(X_{ij})^2)$$

To convert this training objective into one suitable for hyperbolic embeddings, Tifrea et al. change the bolded inner product to $h(d_H(w_i, w_j))$, where d_H is the hyperbolic distance metric defined on page 6, and h is a nonlinear function like $cosh^2$. Intuitively, optimizing over the distance metric d_H forces the feature vectors to occupy the Poincaré ball. Why using h makes sense is an open question, by admission of the authors.

Note that Tifrea et al's construction does not change the dimensionality of GloVe embeddings, meaning they do not embed into the Poincaré disk itself. Instead, this objective maps words into the Poincaré ball, the n-dimensional version of the Poincaré disk. Concretely, if GloVe produces feature vectors in $\mathbb{R}^{100}$, then Poincaré GloVe produces

feature vectors in $\mathbb{H}^{100}$, where $\mathbb{H}^2$ is the Poincaré disk.

The main advantage of this work, then, is that hyperbolic spaces can better represent hierarchy between words. The authors demonstrate this by showing that Poincaré GloVe outperforms vanilla GloVe on the Hyperlex and WBLESS tasks (Baroni and Lenci, 2011; Vulic et al., 2016). Both tasks evaluate lexical entailment. Performance on other tasks for Poincaré GloVe is approximately equal to that of vanilla GloVe.

5.4 Future Work

Hyperbolic embeddings are enticing because they allow us to embed trees hierarchically using only two dimensions. Recent work extending hyperbolic embeddings to words mostly takes advantage of hierarchy, keeping dimensionality the same. Whether we can reduce the dimensionality of our models using hyperbolic embeddings is an open question. Furthermore, as Tifrea et al. (2018) demonstrated, we can use beliefs about geometry to guide our construction of loss functions. Extending the spirit of Tifrea et. al's work to other models like ELMo or BERT is a potentially interesting avenue of research.

6 Conclusion

Understanding the geometry of unsupervised representations is a rich area of research, with many possible connections to other fields. The parallelogram model of analogy pre-dates word2vec and GloVe by 40 years, and several time-tested analyses from cognitive science can be used to analyze modern word embeddings. Why the parallelogram model arises at all in static word embeddings is actively debated. Interestingly enough, Hewitt and Manning (2019) found evidence that ELMo and BERT learn Pythagorean embeddings of syntax trees, sparking new questions as to how these models learn Pythagorean embeddings, why they learn them, and which embeddings the models *should* learn for optimal performance. Finally, hyperbolic geometry, a natural choice for embedding trees, is beginning to see application to word embeddings.

References

Carl Allen and Timothy Hospedales. 2019. Analogies explained: Towards understanding word embeddings. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 223–231, Long Beach, California, USA. PMLR.

Sanjeev Arora, Hrishikesh Khandeparkar, Mikhail Khodak, Orestis Plevrakis, and Nikunj Saunshi. 2019. A theoretical analysis of contrastive unsupervised representation learning. *CoRR*, abs/1902.09229.

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. 2015. Random walks on context spaces: Towards an explanation of the mysteries of semantic word embeddings. *CoRR*, abs/1502.03520.

Marco Baroni and Alessandro Lenci. 2011. How we BLESSed distributional semantic evaluation. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*, pages 1–10, Edinburgh, UK. Association for Computational Linguistics.

Dawn Chen, Joshua C. Peterson, and Thomas L. Griffiths. 2017. Evaluating vector-space models of analogy. *CoRR*, abs/1705.04416.

Andy Coenen, Emily Reif, Ann Yuan, Been Kim, Adam Pearce, Fernanda B. Viégas, and Martin Wattenberg. 2019. Visualizing and measuring the geometry of BERT. *CoRR*, abs/1906.02715.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, page 160–167, New York, NY, USA. Association for Computing Machinery.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Kawin Ethayarajh. 2019. How contextual are contextualized word representations? comparing the geometry of BERT, ELMo, and GPT-2 embeddings. 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 55–65, Hong Kong, China. Association for Computational Linguistics.

Kawin Ethayarajh, David Duvenaud, and Graeme Hirst. 2019. Towards understanding linear word analogies. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3253–3262, Florence, Italy. Association for Computational Linguistics.

Alex Gittens, Dimitris Achlioptas, and Michael W. Mahoney. 2017. Skip-gram âˆ' Zipf + uniform = vector additivity. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 69–76, Vancouver, Canada. Association for Computational Linguistics.

Thomas L. Griffiths, Mark Steyvers, and Joshua B. Tenenbaum. 2007. Topics in semantic representation. *Psychological Review*, 114(2):211–244.

John Hewitt and Percy Liang. 2019. Designing and interpreting probes with control tasks. 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 2733–2743, Hong Kong, China. Association for Computational Linguistics.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *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)*, pages 4129–4138, Minneapolis, Minnesota. Association for Computational Linguistics.

Eric Huang, Richard Socher, Christopher Manning, and Andrew Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 873–882, Jeju Island, Korea. Association for Computational Linguistics.

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 2019. Revealing the dark secrets of BERT. 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 4365–4374, Hong Kong, China. Association for Computational Linguistics.

N. Linial, E. London, and Y. Rabinovich. 1994. The geometry of graphs and some of its algorithmic applications. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, SFCS '94, page 577–591, USA. IEEE Computer Society.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019. Linguistic knowledge and transferability of contextual representations. In *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)*, pages 1073–1094, Minneapolis, Minnesota. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg S. Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013c. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, Atlanta, Georgia. Association for Computational Linguistics.

David Mimno and Laure Thompson. 2017. The strange geometry of skip-gram with negative sampling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2873–2878, Copenhagen, Denmark. Association for Computational Linguistics.

Jiaqi Mu, Suma Bhat, and Pramod Viswanath. 2017. All-but-the-top: Simple and effective postprocessing for word representations. *CoRR*, abs/1702.01417.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *In EMNLP*.

Matthew E. Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. *CoRR*, abs/1705.00108.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *CoRR*, abs/1802.05365.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

David E Rumelhart and Adele A Abrahamson. 1973. A model for analogical reasoning. *Cognitive Psychology*, 5(1):1 – 28.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1986. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536.

Christopher De Sa, Albert Gu, Christopher Ré, and Frederic Sala. 2018. Representation tradeoffs for hyperbolic embeddings. *CoRR*, abs/1804.03329.

Rik Sarkar. 2012. Low distortion delaunay embedding of trees in hyperbolic plane. In *Graph Drawing*, pages 355–366, Berlin, Heidelberg. Springer Berlin Heidelberg.

Alexandru Tifrea, Gary Bécigneul, and Octavian-Eugen Ganea. 2018. Poincaré glove: Hyperbolic word embeddings. *CoRR*, abs/1810.06546.

Amos Tversky. 1977. Features of similarity. *Psychological Review*, 84(4):327–352.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Ivan Vulic, Daniela Gerz, Douwe Kiela, Felix Hill, and Anna Korhonen. 2016. Hyperlex: A large-scale evaluation of graded lexical entailment. *CoRR*, abs/1608.02117.

Eric W. Weisstein. Two-Sheeted Hyperboloid.