EMERGENT TOPOLOGY OF LANGUAGE FROM ATTENTION

AMELIE SCHREIBER

ABSTRACT. We conduct a persistent homology analysis of the attention mechanism in transformers. Using a distance metric based on the KL-divergence we show how language has an emergent simplicial complex structure. An analysis of how this simplicial complex changes over time, that is, as new tokens are added is an open problem and may provide a way to improve our understanding of both persistent homology and of language as it is modeled in transformers. Analyzing linguistic structures captured by the simplicial complex of persistent homology in different contexts provides new avenues of research for linguists and those working on transformers for language tasks. We also give some hints at possible applications of the analysis for improving temporal coherence in generative text-to-video tasks, and improving language translation. For temporal coherence an extension to two-parameter persistent homology is required, but we show how this can be reduced to a one-parameter persistent homology with discrete time slices, making it possible for persistence diagrams and barcodes to be computed. This in turn is key in the potential applications to generative tasks and translation.

CONTENTS

1. INTRODUCTION

The emergence of transformer-based models has significantly impacted the field of natural language processing (NLP), establishing new standards in a variety of tasks such as machine translation, sentiment analysis, and text summarization. A critical component of transformer architectures is the attention mechanism, which enables the model to dynamically assess the significance of different tokens in the input sequence. Although these models exhibit remarkable performance, the inner workings of transformers and their attention mechanisms are still not fully understood. A deeper comprehension of their underlying structure may result in further progress in NLP tasks.

Date: May 25, 2023.
2020 *Mathematics Subject Classification.* Primary 55N31 68T07 68T50 Secondary 91F20 62R40 68T09 .

In this paper, we examine the attention mechanism in transformer-based models using persistent homology[1], a computational topology tool adept at analyzing high-dimensional data and detecting topological features that persist across multiple scales. By utilizing a distance metric based on Kullback-Leibler (KL) divergence, we unveil the presence of a simplicial complex structure in language, providing a fresh outlook on the linguistic structures captured by transformers.

We initially offer an in-depth analysis of the simplicial complex derived from the attention mechanism, discussing its implications for both persistent homology and language modeling in transformers. We also identify several open challenges, such as understanding the simplicial complex's evolution as new tokens are integrated. Addressing these challenges can result in valuable insights and advancements in transformer-based language tasks.

Besides the theoretical analysis, we investigate potential applications of our findings in improving temporal coherence in generative tasks and refining language translation. To tackle temporal coherence, we suggest an extension to two-parameter persistent homology, which can be simplified to a one-parameter approach with discrete time slices. This simplification enables the computation of persistence diagrams and barcodes, essential for the proposed applications in generative tasks and translation.

The contributions of this paper are as follows:

(1) We conduct a thorough examination of the attention mechanism in transformer-based models using persistent homology, uncovering the emergence of a simplicial complex structure in language.
(2) We emphasize open challenges and potential directions for future research in understanding and enhancing transformer models through the study of persistent homology.
(3) We introduce potential applications of our analysis in improving temporal coherence in generative tasks and refining language translation, focusing on extending persistent homology to address temporal aspects.

2. Jensen-Shannon Distance Metric

The Jensen-Shannon distance (JSD) is a metric derived from the Jensen-Shannon divergence (JSDiv), a symmetrized and smoothed version of the Kullback-Leibler (KL) divergence. KL-divergence is a popular measure of dissimilarity between probability distributions and has found widespread application in deep learning.

To define the Jensen-Shannon divergence, we first recall the KL-divergence between two probability distributions P and Q:

$$\mathrm{KL}(P\|Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}, \tag{1}$$

where i ranges over the elements in the support of the distributions. Note that KL-divergence is not symmetric, i.e., $\mathrm{KL}(P\|Q) \neq \mathrm{KL}(Q\|P)$. In order to derive a symmetrized measure, we introduce the Jensen-Shannon divergence:

$$\mathrm{JSDiv}(P, Q) = \frac{1}{2}\mathrm{KL}(P\|M) + \frac{1}{2}\mathrm{KL}(Q\|M), \tag{2}$$

where $M = \frac{1}{2}(P + Q)$ is the average of the two probability distributions. The Jensen-Shannon divergence satisfies the properties of being symmetric, i.e., $\mathrm{JSDiv}(P, Q) = \mathrm{JSDiv}(Q, P)$, and non-negative, i.e., $\mathrm{JSDiv}(P, Q) \geq 0$.

[1]The code for this paper can be found at https://github.com/Amelie-Schreiber/persistent_homology_of_attention

However, the Jensen-Shannon divergence is not a metric, as it does not satisfy the triangle inequality. To obtain a metric, we define the Jensen-Shannon distance as the square root of the Jensen-Shannon divergence:

$$\mathrm{JSD}(P, Q) = \sqrt{\mathrm{JSDiv}(P, Q)}. \tag{3}$$

The Jensen-Shannon distance satisfies the properties of a metric, including non-negativity, symmetry, the identity of indiscernibles, and the triangle inequality. This makes it a useful measure of dissimilarity between probability distributions for various applications, including deep learning.

3. Multi-head Self-Attention

3.1. **Attention and Attention Blocks.** The self-attention mechanism is a critical component of transformer-based models, enabling the model to weigh the importance of different tokens in the input sequence dynamically. In this section, we provide a technical explanation of the self-attention matrix used in these models, which can be defined as:

$$\mathrm{Attn}(X) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right), \tag{4}$$

where Q, K, and V represent the query, key, and value matrices, respectively, and d_k is the dimensionality of the key vectors.

Given an input sequence $X = x_1, x_2, \ldots, x_n$, where each x_i is a token in the sequence, the self-attention mechanism aims to compute a weighted sum of the input representations based on their contextual relevance. To achieve this, the input sequence is first linearly transformed into query, key, and value matrices using separate weight matrices W^Q, W^K, and W^V:

$$Q = XW^Q, \tag{5}$$

$$K = XW^K, \tag{6}$$

$$V = XW^V. \tag{7}$$

The query matrix Q represents the transformed input sequence, and each row in the key matrix K corresponds to the contextual importance of a token with respect to another token. By computing the dot product between the query and key matrices, we obtain an affinity matrix that measures the contextual relevance between each pair of tokens:

$$S = QK^T. \tag{8}$$

To ensure that the self-attention mechanism is invariant to the scale of the input, we normalize the affinity matrix by dividing each element by the square root of the key vector dimensionality, i.e., $\sqrt{d_k}$:

$$\widetilde{S} = \frac{S}{\sqrt{d_k}}. \tag{9}$$

Finally, we apply the softmax function to the normalized affinity matrix row-wise, obtaining the self-attention matrix $\mathrm{Attn}(X)$:

$$\mathrm{Attn}(X) = \mathrm{softmax}(\widetilde{S}). \tag{10}$$

Each row of the self-attention matrix represents the attention weights for a specific token in the input sequence, and these weights are used to compute a weighted sum of the value matrix V, resulting in the final output of the self-attention mechanism. The output of the attention layer is then calculated as:

$$Y = Attn(X)XW^V = Attn(X)V \tag{11}$$

with W^V the learned value matrix.

3.2. **Probability Distributions from Tokens.** Now, taking X_i the i^{th} row of the matrix of token embedding vectors X, we denote by

$$\langle q_i, k_j \rangle = (X_i W^Q)(X_j W^K)^T. \tag{12}$$

Then we have,

$$P(X_i) = \left(\text{softmax}_j\left(\frac{\langle q_i, k_j\rangle}{\sqrt{d}}\right)\right)_{j=1}^{n} = \text{softmax}\begin{pmatrix} \frac{\langle q_i,k_1\rangle}{\sqrt{d}} \\ \frac{\langle q_i,k_2\rangle}{\sqrt{d}} \\ \vdots \\ \frac{\langle q_i,k_n\rangle}{\sqrt{d}} \end{pmatrix} = \left(\frac{e^{\frac{\langle q_i,k_j\rangle}{\sqrt{d}}}}{\sum_{l=1}^{n} e^{\frac{\langle q_i,k_l\rangle}{\sqrt{d}}}}\right)_{j=1}^{n}. \tag{13}$$

To measure the dissimilarity between the attending behaviors of tokens X_i and X_j, we can compute the Kullback-Leibler (KL) divergence, denoted as $D_{KL}(P(X_i)\|P(X_j))$. The formula for KL divergence is:

$$KL(P(X_i)\|P(X_j)) = \sum_{k=1}^{n} P(X_i)_k \log_2 \frac{P(X_i)_k}{P(X_j)_k} \tag{14}$$

$$= \sum_{k=1}^{n} \frac{e^{\frac{\langle q_i,k_k\rangle}{\sqrt{d}}}}{\sum_{l=1}^{n} e^{\frac{\langle q_i,k_l\rangle}{\sqrt{d}}}} \log_2 \left(\frac{\frac{e^{\frac{\langle q_i,k_k\rangle}{\sqrt{d}}}}{\sum l=1^n e^{\frac{\langle q_i,k_l\rangle}{\sqrt{d}}}}}{\frac{e^{\frac{\langle q_j,k_k\rangle}{\sqrt{d}}}}{\sum_{l=1}^{n} e^{\frac{\langle q_j,k_l\rangle}{\sqrt{d}}}}} \right) \tag{15}$$

$$= \sum_{k=1}^{n} \frac{e^{\frac{\langle q_i,k_k\rangle}{\sqrt{d}}}}{\sum_{l=1}^{n} e^{\frac{\langle q_i,k_l\rangle}{\sqrt{d}}}} \left(\log_2 \left(\frac{e^{\frac{\langle q_i,k_k\rangle}{\sqrt{d}}}}{\sum_{l=1}^{n} e^{\frac{\langle q_i,k_l\rangle}{\sqrt{d}}}} \right) - \log_2 \left(\frac{e^{\frac{\langle q_j,k_k\rangle}{\sqrt{d}}}}{\sum_{l=1}^{n} e^{\frac{\langle q_j,k_l\rangle}{\sqrt{d}}}} \right) \right) \tag{16}$$

3.3. **Multihead Attention.** Multi-head self-attention is an extension of the self-attention mechanism, which allows the model to capture multiple contextual relationships among tokens in the input sequence simultaneously. The concept of multi-head self-attention involves dividing the original attention mechanism into multiple parallel sub-components, referred to as "heads." Each head computes its own attention matrix and output, which are then combined to form the final output.

Mathematically, the multi-head self-attention can be defined as follows:

$$\text{MultiHead}(X) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O, \tag{17}$$

where h is the number of heads, and each head_i is the output of the self-attention mechanism for the i-th head:

$$\text{head}_i = Y_i \tag{18}$$

$$= Attn_i(X)V_i \tag{19}$$

$$= \text{softmax}\left(\frac{Q_i K_i^T}{\sqrt{d_k}}\right)V_i \tag{20}$$

$$= \text{softmax}\left(\frac{(XW_i^Q)(XW_i^K)^T}{\sqrt{d_k}}\right)XW_i^V \tag{21}$$

with W_i^Q, W_i^K, and W_i^V being the weight matrices for the query, key, and value transformations for the i-th head, respectively. The final output of the multi-head self-attention is obtained by concatenating the outputs of all heads and applying a linear transformation using the weight matrix W^O.

The following transformer-based models are explored in the code for this paper and multi-head self-attention is employed with different configurations of layers and heads in each:

(1) GPT-2 (gpt2): GPT-2 consists of 12 layers in its base configuration, with each layer utilizing 12 attention heads.
(2) DistilBERT (distilbert-base-uncased): DistilBERT is a smaller, distilled version of BERT, comprising 6 layers, with each layer employing 12 attention heads.
(3) BERT (bert-base-uncased): BERT's base configuration consists of 12 layers, with each layer utilizing 12 attention heads.
(4) RoBERTa (roberta-base): RoBERTa is a variant of BERT and has a similar architecture. In its base configuration, it includes 12 layers, with each layer employing 12 attention heads.
(5) Aleph-BeRT (onlplab/alephbert-base): Aleph-BeRT is a Hebrew language model based on BERT. Its base configuration consists of 12 layers, with each layer utilizing 12 attention heads.

The use of multiple layers and heads in these models allows for the extraction of rich contextual information from the input sequences, enabling the models to achieve state-of-the-art performance in a wide range of natural language processing tasks. Note, there are also additional tools used in these models such as masking and positional encodings that also make the models distinct. The persistent homology analysis is based entirely on the attention matrix $Attn(X)$, so we feel it is important to develop the theory further to account for the mechanics of masking and positional encodings as well. We leave this for future work.

3.4. **Function Formulation of Attention.** In this subsection, we present a function formulation of the self-attention mechanism in transformers. This perspective emphasizes the functional relationships between different components of the mechanism and can be particularly helpful for understanding and analyzing the properties of self-attention.

In practice, the dimensions of the weight matrices W^Q, W^K, W^V, and W^O depend on the specific configuration of the transformer model. Let's consider the most common configuration:

- d_{model}: the dimension of the input token embeddings (and also the output dimension of the transformer)
- h: the number of attention heads
- d_k: the dimension of the keys and queries per head
- d_v: the dimension of the values per head

Under this configuration, the dimensions of the weight matrices are as follows:

- W^Q: This matrix is responsible for transforming the input token embeddings into query vectors. The matrix has dimensions of $d_{model} \times (h \times d_k)$.
- W^K: This matrix is responsible for transforming the input token embeddings into key vectors. The matrix has dimensions of $d_{model} \times (h \times d_k)$.
- W^V: This matrix is responsible for transforming the input token embeddings into value vectors. The matrix has dimensions of $d_{model} \times (h \times d_v)$.
- W^O: This matrix is responsible for transforming the concatenated context vectors from all heads back to the original embedding dimension. The matrix has dimensions of $(h \times d_v) \times d_{model}$.

Typically, d_k and d_v are chosen such that $d_k = d_v = \frac{d_{model}}{h}$. This choice ensures that the overall computational complexity of the multi-head attention mechanism is comparable to that of single-head attention with the full embedding dimension.

Consider a sequence of input tokens represented by a matrix X. We treat X as a vector-valued function $f : S \to \mathbb{R}^d$, where $S = \{1, ..., n\}$ is the set of token positions in the sequence, and $f(i) = x_i$ returns the i-th token's vector representation. Furthermore, let $L_{\mathbb{R}^k}(S) = \{g | g : S \to \mathbb{R}^k\}$ be the space of functions from the set S to $\mathbb{R}^k$.

We define three functions, $\phi_{qry}, \phi_{key}, \phi_{val} : L_{\mathbb{R}^d}(S) \to L_{\mathbb{R}^m}(S)$, as linear transformations corresponding to the Query (Q), Key (K), and Value (V) matrices:

(1) $\phi_{qry}(f(i)) = f(i) * W^Q = Q_i$
(2) $\phi_{key}(f(i)) = f(i) * W^K = K_i$
(3) $\phi_{val}(f(i)) = f(i) * W^V = V_i$

With these definitions, we can describe the self-attention mechanism as follows:

(1) Compute attention scores using the transformed functions: $score(\phi_{qry}(f(i)), \phi_{key}(f(j))) = \langle \phi_{qry}(f(i)), \phi_{key}(f(j)) \rangle / \sqrt{d_k}$
(2) Apply softmax to the attention scores to obtain attention weights:

$$attention\_weights_{ij} = \frac{e^{score(\phi_{qry}(f(i)), \phi_{key}(f(j)))}}{\sum_{k=1}^{n} e^{score(\phi_{qry}(f(i)), \phi_{key}(f(k)))}}$$

(3) Calculate the context vector using the attention weights and transformed value functions: $C_i = \sum_{j=1}^{n} attention\_weights_{ij} * \phi_{val}(f(j))$

To incorporate multi-head attention with h heads, we first compute the context vectors for each head $l = 1, 2, ..., h$: $C_i^l = \sum_{j=1}^{n} attention\_weights_{ij}^l * \phi_{val}^l(f(j))$. Then, we concatenate the context vectors from all heads: $C_i^{concat} = [C_i^1; C_i^2; \cdots; C_i^h]$. Finally, we apply a linear transformation using the output weight matrix W^O: $C_i = C_i^{concat} * W^O$

This perspective on attention is useful for many reasons. First, it provides a link to positional encodings and allows one to formulate equivariance of attention using positional encoding. It is also used in the proof of permutation equivariance, translation equivariance, and general group equivariance of various formulations of attention with different kinds of positional encodings. For our purposes, we are mainfly interested in the *context vectors*, as these will be studied later on, when we attempt tp locate topological structures that are present in different contexts. A precise notion of contextual mapping will be introduced in the section on applications.

Now, suppose we are given a way of clustering the probability distributions $P(x_i)$ associated to tokens by the softmax of the attention matrix. If a subset of the tokens are then found in a separate context (that is, a separate input text $X' = \{x'_1, x'_2, ...x'_n\}$), they will have different context vectors. Is it possible that if they are clustered tightly in one context, they will also be clustered tightly in a different context? This is one of the question we attempt to address in this article.

4. Persistent Homology

4.1. **One Parameter Persistent Homology.** Persistent homology is a powerful mathematical tool from the field of computational topology that provides a robust, multi-scale analysis of topological features in high-dimensional data. It quantifies the persistence of topological features such as connected components, loops, and voids over a range of scales, enabling the identification of significant structures in the underlying space. In this subsection, we provide a mathematically rigorous introduction to one-parameter persistent homology.

Given a topological space X and a continuous function $f : X \to \mathbb{R}$, we aim to study the evolution of the homology groups of X as the function f varies over its domain. To achieve this, we consider the sublevel sets of the function f, defined as:

$$X_a = f^{-1}((-\infty, a]), \tag{22}$$

where $a \in \mathbb{R}$. As the parameter a increases, the sublevel sets X_a form a growing sequence of nested spaces:

$$X_{a_1} \subseteq X_{a_2} \subseteq \cdots \subseteq X_{a_n}, \tag{23}$$

for any sequence of real numbers $a_1 \leq a_2 \leq \cdots \leq a_n$.

For each a_i, we compute the homology groups $H_p(X_{a_i})$ of the corresponding sublevel set, where $p \geq 0$ denotes the dimension of the homological features we are interested in (e.g., $p = 0$ for connected components, $p = 1$ for loops, etc.). As a increases, homological features may appear, merge, or disappear, resulting in changes in the homology groups.

To track the persistence of these homological features, we define an algebraic structure known as a persistence module. A persistence module M is a collection of vector spaces $\{M_a\}_{a\in\mathbb{R}}$ indexed by the real numbers, together with linear maps $ma, b : M_a \to M_b$ for all $a \leq b$, satisfying the following conditions:

(1) $m_{a,a} = \mathrm{id}_{M_a}$ (identity maps) for all $a \in \mathbb{R}$,
(2) $m_{a,c} = m_{b,c} \circ m_{a,b}$ (composition of maps) for all $a \leq b \leq c$.

The homology groups $H_p(X_a), a \in \mathbb{R}$ and the induced homomorphisms $\{f_{a,b} : H_p(X_a) \to H_p(X_b)\}_{a\leq b}$ form a persistence module, which encapsulates the evolution of the topological features over the filtration parameter a.

The key idea in persistent homology is to summarize the information contained in the persistence module using persistence diagrams, a compact representation of the birth and death of topological features over the filtration. A persistence diagram is a multi-set of points in the extended plane $\overline{\mathbb{R}}^2 = \mathbb{R}^2 \cup \infty$, where each point (b, d) represents a homological feature born at b and dying at d. The persistence of a feature is given by the difference $d - b$, and features with high persistence are considered more significant.

To compute persistence diagrams from the persistence module, one can employ various algorithms, such as the standard reduction algorithm or the faster matrix reduction algorithms, which leverage matrix representations of the boundary operators in the filtered simplicial complex. The matrix reduction algorithms transform the boundary matrices into a canonical form, from which the persistence pairs can be directly read off.

One crucial aspect of persistent homology is its stability under perturbations of the input data. This stability is captured by the concept of interleaving distance between persistence modules, which measures the similarity between the evolution of topological features in two different spaces. More specifically, two persistence modules M and N are said to be ϵ-interleaved if there exist linear maps

$m_a : M_a \to N_{a+\epsilon}$ and $n_a : N_a \to M_{a+\epsilon}$ for all $a \in \mathbb{R}$, such that $n_{a+\epsilon} \circ m_a = \mathrm{id}_{M_{a+\epsilon}}$ and $m_{a+\epsilon} \circ n_a = \mathrm{id}_{N_{a+\epsilon}}$. Intuitively, an ϵ-interleaving between two persistence modules means that the topological features in one module are matched to similar features in the other module within a tolerance of ϵ.

The interleaving distance provides a solid foundation for the stability of persistence diagrams, as it can be shown that the bottleneck distance between persistence diagrams is bounded by the interleaving distance between their corresponding persistence modules. The bottleneck distance is a popular metric for comparing persistence diagrams and is defined as the minimum value δ such that each point in one diagram can be matched to a point in the other diagram within a distance of δ in the ℓ_∞-norm.

A $\mathbb{C}[x]$-module is an algebraic structure that combines the notion of a vector space over the complex numbers, $\mathbb{C}$, with the action of polynomials in one variable, x. More formally, a $\mathbb{C}[x]$-module is a set M along with an operation $+ : M \times M \to M$ and an action of $\mathbb{C}[x]$ on M, denoted by $\cdot : \mathbb{C}[x] \times M \to M$, such that:

(1) $(M, +)$ is an abelian group.
(2) The action of $\mathbb{C}[x]$ on M is compatible with the addition in M and the multiplication in $\mathbb{C}[x]$.

At a first glance, these two concepts may seem unrelated, but there is a connection between them through the notion of a "graded module." A graded module is a module that has a direct sum decomposition indexed by a partially ordered set, which in our case is the parameter t.

Persistence modules can be seen as graded modules indexed by the parameter t. Similarly, $\mathbb{C}[x]$-modules can be considered as graded modules indexed by the degree of the polynomial. The connection between these concepts is more apparent when considering a specific subclass of persistence modules called "pointwise finite-dimensional" (PFD) persistence modules. A persistence module is PFD if all its vector spaces V_t are finite-dimensional. PFD persistence modules can be represented using a $\mathbb{C}[x]$-module structure, where the algebraic structure of the persistence module is induced by the action of the polynomial ring $\mathbb{C}[x]$ on the module. This connection helps translate topological problems involving persistence modules into algebraic problems involving $\mathbb{C}[x]$-modules, which are often easier to work with and study. Moreover, we can always decompose such modules as[2],

$$U \simeq \bigoplus_i x^{t_i} \cdot \mathbb{C}[x] \oplus \left(\bigoplus_j x^{r_j} \cdot (\mathbb{C}[x]/(x^{s_j} \cdot \mathbb{C}[x])) \right). \tag{24}$$

One-parameter persistent homology provides a robust, multi-scale analysis of topological features in data by tracking the evolution of homology groups over a filtration parameter. Persistence diagrams and barcode offer a compact representation of the birth and death of topological features, and their stability under perturbations is guaranteed by the interleaving distance between persistence modules. Persistent homology has found applications in a wide range of fields, including shape analysis, data clustering, and the study of complex networks, among others.

4.2. **Two Parameter Persistent Homology.** The theory of multiparameter persistence arises in many contexts in computational topology and data science. Often a point finite cloud $P \subset \mathbb{R}^d$ is obtained from some observational data, and one would like to understand the significant features of the data cloud. It is typical that the data set will have some *"noise"*, and such noise may need to be filtered out. It is in general a very difficult problem to analyse a data set and determine what subset of the data is noise and what is a significant feature of the data. In this case, one can use topological data analysis and persistence homology. This method has proven very robust and effective in applications.

[2]It is important to note that the coefficient field does not necessarily need to be $\mathbb{C}$. In [AS] we show how to study the commutative algebra and algebraic geometry over $\mathbb{Z}$.

Definition 4.1. A **bifiltered space** X, is a topological space with a family of subspaces $\{X_v \subseteq\}_{v\in\mathbb{N}^2}$, with inclusion maps $X_u \hookrightarrow X_v$ whenever $u \leq v$ in the standard partial order on $\mathbb{N}^2$. We require the following diagram to commute:

$$\begin{array}{ccc} X_u & \longrightarrow & X_{v_1} \\ \downarrow & & \downarrow \\ X_{v_2} & \longrightarrow & X_w \end{array}$$

whenever $u \leq v_1, v_2 \leq w$.

Definition 4.2. Let $\mathbb{K}$ be a field. A **persistence module** M, is a family of $\mathbb{K}$-modules $\{M_u\}_{u\in\mathbb{N}^2}$ with maps

$$\phi_{u,v} : M_u \to M_v$$

for all $u \leq v$ such that $\phi_{v,w} \circ \phi_{u,v} = \phi_{u,w}$ for all $u \leq v \leq w$. Let M be a persistence module and let $R = \mathbb{K}[x, y]$. Define

$$\alpha(M) = \bigoplus_v M_v$$

where the $\mathbb{K}$-module structure is the direct sum structure, and $x^{u-v} : M_u \to M_v$ is $\phi_{u,v}$ when $u \leq v$ in $\mathbb{N}^2$. This gives an equivalence of categories between the category of finite persistence modules over $\mathbb{K}$, and the category of $\mathbb{N}^2$-graded finitely generated modules over R.

This means we may use commutative algebra and algebraic geometry to study 2-parameter persistent homology. Moreover, many of the invariants that arise from this study can be efficiently computed using computer algebra software such as SageMath and Macaulay2. In the next subsection we investigate how we can reduce out study to discrete time, allowing for use of one-parameter persistent homology tools such as persistence diagrams, barcodes, and distance metrics defined on them such as the bottleneck and Wasserstein distance.

4.3. **Discrete Time Reduction.** In the context of the attention mechanism in transformer-based models, we can employ two-parameter persistent homology to study the topological properties of the attention distribution and how they evolve over time. In this case, the two parameters correspond to the Jensen-Shannon distance and time (as discrete steps in the input sequence). This approach provides a richer understanding of the underlying linguistic structures captured by the attention mechanism in transformer models.

The first parameter, Jensen-Shannon distance, is a symmetrized and smoothed version of the Kullback-Leibler divergence, which measures the dissimilarity between two probability distributions. Given the attention mechanism's output, we can view it as a collection of probability distributions across tokens in the input sequence. The Jensen-Shannon distance can be used to construct a distance matrix between all pairs of tokens, which in turn can be used to build a weighted simplicial complex to represent the relationships between tokens.

The second parameter, time, represents the discrete steps in the input sequence. As tokens are added to the input sequence, the attention mechanism updates its representation of the sequence and the relationships between tokens. This time component allows us to capture the temporal evolution of the attention mechanism and investigate how linguistic structures change as the input sequence grows.

5. Examples with 1-parameter Persistent Homology

The following code computes the pairwise Jensen-Shannon distance matrix between the probability distributions associated to tokens by the attention mechanism. It computes the distance matrix for two text inputs, and allows for selecting the layer and head to use:

Listing 1. Jensen-Shannon Distance Matrices Calculation

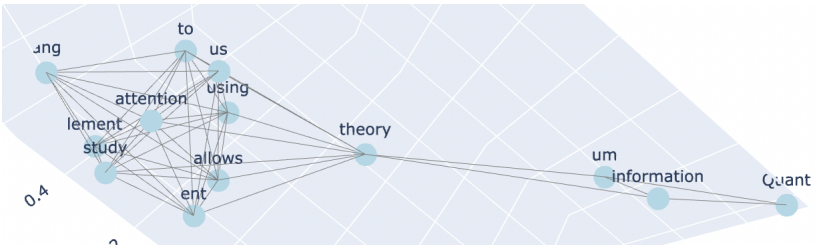
```
import torch
from transformers import BertTokenizer, BertModel
import numpy as np
from scipy.spatial.distance import jensenshannon
from ipywidgets import interact, widgets

def js_distance_matrix(layer: int, head: int, input_text1: str, input_text2: str):
    def get_distance_matrix(input_text):
        model_name = 'bert-base-uncased'
        tokenizer = BertTokenizer.from_pretrained(model_name)
        model = BertModel.from_pretrained(model_name, output_attentions=True)

        input_ids = torch.tensor(tokenizer.encode(input_text)).unsqueeze(0)
        outputs = model(input_ids)

        attention_weights = outputs.attentions[layer - 1][0, head - 1].detach().numpy()

        p_x = np.apply_along_axis(lambda x: np.exp(x) / np.sum(np.exp(x)), 1, attention_weights)

        num_tokens = p_x.shape[0]
        js_distance_mat = np.zeros((num_tokens, num_tokens))

        for i in range(num_tokens):
            for j in range(num_tokens):
                js_distance = jensenshannon(p_x[i], p_x[j])
                js_distance_mat[i, j] = js_distance

        return js_distance_mat

    js_distance_matrix_result1 = get_distance_matrix(input_text1)
    js_distance_matrix_result2 = get_distance_matrix(input_text2)

    return js_distance_matrix_result1, js_distance_matrix_result2


def interactive_js_distance_matrix(layer: int, head: int, input_text1: str, input_text2: str):
    js_distance_matrix_result1, js_distance_matrix_result2 =
                                    js_distance_matrix(layer, head, input_text1, input_text2)

    print("Matrix 1 dimensions:", js_distance_matrix_result1.shape)
    print(js_distance_matrix_result1)
    print("Matrix 2 dimensions:", js_distance_matrix_result2.shape)
    print(js_distance_matrix_result2)

interact(
    interactive_js_distance_matrix,
    layer=widgets.IntSlider(min=1, max=12, step=1, value=1),
    head=widgets.IntSlider(min=1, max=12, step=1, value=1),
    input_text1=widgets.Text(value='Example text 1', description='Text 1:'),
    input_text2=widgets.Text(value='Example text 2 is longer', description='Text 2:')
```

Now, because these distance matrices are in fact genuine distance martices (using Jensen-Shannon-distance), we can compute the persistent homology associate to them. We treat each token like a node in a weighted graph, with edge weights corresponding to the JS-distance between the probability distributions associated to the tokens by the attention mechanism. This gives a filtration of the weighted graph and an associated persistence diagram describing the persistent homology of the distributions. If we do this for two text inputs, and allows ourselfves to compare the attention in different layers and heads of the transformer, we can gain insight into the behavior of the attention mechanism. In particular, we see how the distributions associated to tokens cluster in an information theoretic way.

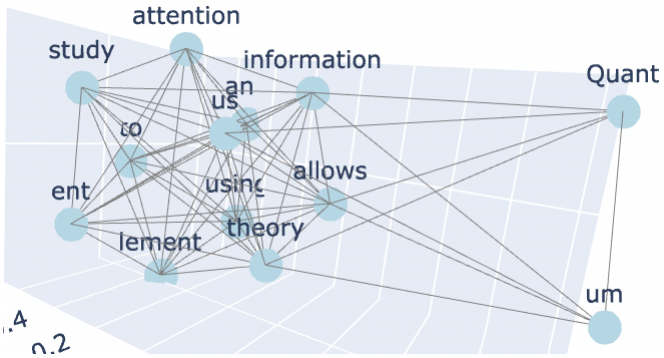
From here, we can compute the persistent homology, persistence diagrams, and barcodes using Gudhi. Gudhi is a library for topological data analysis and provides efficient tools to compute persistent homology and persistence diagrams from a distance matrix. In the case of the Jensen-Shannon distance matrix, the matrix is constructed using pairwise distances between probability distributions associated with tokens. Here, we present a step-by-step explanation of how Gudhi computes persistent homology and persistence diagrams for H_0 and H_1, and how it derives a simplicial complex for plotting.

(1) Jensen-Shannon Distance Matrix: Given a set of probability distributions associated with tokens, compute the pairwise distances between these distributions using the Jensen-Shannon distance metric. This results in a symmetric distance matrix.
(2) Vietoris-Rips Complex: From the Jensen-Shannon distance matrix, Gudhi constructs a Vietoris-Rips complex, which is a simplicial complex built from a set of points in a metric space. The complex is constructed by including a simplex for each subset of points whose pairwise distances are less than or equal to a certain filtration value ϵ. As the filtration value increases, the simplicial complex becomes more connected, capturing the underlying structure of the data.
(3) Simplicial Complex Filtration: Gudhi applies a filtration process to the Vietoris-Rips complex by incrementally increasing the filtration value ϵ. At each filtration step, new simplices are added to the complex, and the homology groups are updated. The filtration captures the evolution of topological features in the data, such as connected components (H_0) and loops (H_1), as the filtration value increases.
(4) Persistent Homology Computation: Persistent homology is computed from the filtered simplicial complex. Gudhi tracks the birth and death of topological features throughout the filtration process. For each homology group (H_0, H_1, etc.), Gudhi creates persistence pairs representing the birth and death filtration values of each topological feature. These persistence pairs are then used to compute the persistence diagrams or barcodes.
(5) Persistence Diagrams (or barcodes): For each homology group, Gudhi constructs a persistence diagram, which is a scatter plot of the persistence pairs in the birth-death plane. Points in the persistence diagram represent topological features, and their distance from the diagonal indicates their persistence. Points far from the diagonal correspond to significant topological features that persist across a wide range of filtration values, while points close to the diagonal are considered noise or less significant features.
(6) Simplicial Complex for Plotting: To visualize the structure captured by the persistent homology computation, Gudhi can extract a simplicial complex from the Vietoris-Rips complex at a specific filtration value. This simplicial complex can be plotted to visualize the relationships between tokens in the language model, based on the Jensen-Shannon distance metric.

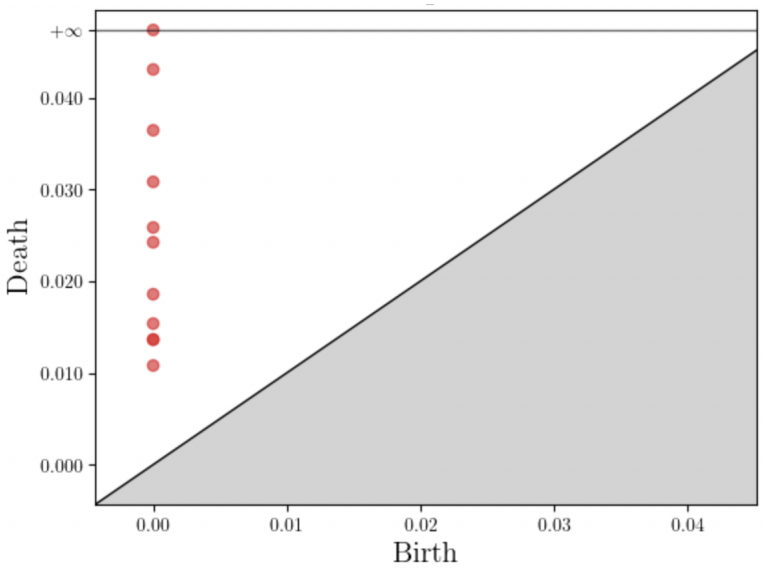
Below, we can see the 1-skeleton of the simplicial complex associated to one of the attention heads (and for some fixed JS-distance thresholds):



For another head and value of the JS-distance threshold we can see the simplicial complex changes:



The persistence diagram corresponding to layer 1 and head 2 of GPT-2 for the input text ”Quantum information theory allows us to study attention using entanglement” is pictured below:



6. Global Persistent Homology of a Transformer

6.1. **Distance Metric Between Heads.** In this section we focus on computing the distance between two attention heads to analyze the behavior of attention heads in deep learning models such as BERT and GPT-2. We define the distance metric between two heads H_i and H_j as follows:

$$D(H_i, H_j) = \sum_{\text{token} \in \text{input data}} JS\left(H_i(token), H_j(token)\right)$$

Here, JS denotes the Jensen-Shannon divergence. In order to perform a persistent homology analysis, which necessitates a true distance metric, we must ensure that our distance metric satisfies the properties of non-negativity, identity of indiscernibles, symmetry, and triangle inequality. The Jensen-Shannon divergence already satisfies the first three properties; thus, we need only address the triangle inequality. We achieve this by employing the square root of Jensen-Shannon divergence, known as Jensen-Shannon distance (JSD), as our distance metric:

$$D(H_i, H_j) = \sum_{\text{token} \in \text{input data}} \sqrt{JS\left(H_i(\text{token}), H_j(\text{token})\right)}.$$

With this modification, our distance metric now satisfies all the necessary properties (which we prove below). We subsequently compute the persistent homology and plot the barcode diagram for BERT and GPT-2 individually, followed by a joint analysis, and calculate the bottleneck distance between their barcode diagrams. This approach is inspired by the analysis in §6 of the article [What Does BERT Look At? An Analysis of BERT's Attention](https://arxiv.org/abs/1906.04341), which investigates the similarity and grouping of attention heads based on their behavior.

The findings suggest that attention heads within the same layer exhibit similar attention distributions, forming distinct clusters. This observation is intriguing considering that [Multi-Head Attention with Disagreement Regularization](https://arxiv.org/abs/1810.10183) demonstrated that encouraging diverse attention head behavior can enhance Transformer performance in machine translation. One possible explanation for this redundancy in BERT's attention heads is the use of attention dropout during training. Consequently, we can infer that persistent topological features may negatively impact model performance, prompting the need to discourage such features during training. By promoting larger Jensen-Shannon distances between specific attention heads that form clusters or higher-dimensional topological features, we enable the model to increase pairwise distances $D(H_i, H_j)$ without dictating the distribution of individual attention weights of a single head, thereby retaining flexibility in the distribution of attention weights while hopefully enhancing overall performance.

6.2. **Proving the distance metric properties.** Now, for an individual token, $\sqrt{JS\left(H_i(\text{token}), H_j(\text{token})\right)}$ is a distance metric. If we want to prove that $D(H_i, H_j) = \sum_{\text{token} \in \text{input data}} \sqrt{JS\left(H_i(\text{token}), H_j(\text{token})\right)}$ is a distance metric we technically need to show that the sum of two distance metrics is a distance metric since for two or more tokens we are summing two or more distance metrics. It suffices to show that the sum of two distance metrics is a distance metric.

To determine if the sum of two distance metrics is also a distance metric, let's review the properties that a function must satisfy in order to be considered a distance metric. A function $d(x, y)$ is a distance metric if it satisfies the following four conditions for all points x, y, and z:

(1) Non-negativity: $d(x, y) \geq 0$
(2) Identity of indiscernibles: $d(x, y) = 0$ if and only if $x = y$
(3) Symmetry: $d(x, y) = d(y, x)$
(4) Triangle inequality: $d(x, y) + d(y, z) \geq d(x, z)$

Now, let's assume we have two distance metrics $d_1(x, y)$ and $d_2(x, y)$. We'll create a new function $d(x, y) = d_1(x, y) + d_2(x, y)$ and check if it satisfies the properties of a distance metric.

(1) Non-negativity: Since $d_1(x, y)$ and $d_2(x, y)$ are both non-negative, their sum will also be non-negative.
(2) Identity of indiscernibles: If $x = y$, both $d_1(x, y)$ and $d_2(x, y)$ will be 0, so their sum will be 0. If $d(x, y) = 0$, that means $d_1(x, y) + d_2(x, y) = 0$. Since both d_1 and d_2 are distance metrics, they must be non-negative, so the only way for their sum to be 0 is if both $d_1(x, y)$ and $d_2(x, y)$ are 0. This implies $x = y$.
(3) Symmetry: $d(x, y) = d_1(x, y) + d_2(x, y) = d_1(y, x) + d_2(y, x) = d(y, x)$
(4) Triangle inequality: $d(x, y)+d(y, z) = (d_1(x, y)+d_2(x, y))+(d_1(y, z)+d_2(y, z)) = (d_1(x, y)+d_1(y, z)) + (d_2(x, y) + d_2(y, z))$

Since d_1 and d_2 are distance metrics, they both satisfy the triangle inequality. So, $d_1(x, y) + d_1(y, z) \geq d_1(x, z)$ and $d_2(x, y) + d_2(y, z) \geq d_2(x, z)$. Adding these inequalities, we have: $(d_1(x, y) + d_1(y, z)) + (d_2(x, y) + d_2(y, z)) \geq d_1(x, z) + d_2(x, z)$ which simplifies to: $d(x, y) + d(y, z) \geq d(x, z)$. Since $d(x, y)$ satisfies all four properties of a distance metric, we can conclude that the sum of two distance metrics is also a distance metric.

7. Applications

7.1. **Information Compression and Reducing Complexity of the Attention Mechanism.** As highlighted in [SJENMC], transformers face two main limitations, one being their quadratic complexity ($O(N^2)$) due to pairwise affinity computation, posing a significant challenge for video applications. Consequently, a method to compress the information needed to represent the attention mechanism in a transformer is highly desirable. In this context, we explore how persistent homology might aid in achieving this goal. Given a "significant" persistent topological structure, we may want to replace the distributions associated with tokens by the softmax of the attention matrix, using a representative akin to choosing a cluster representative in data or a centroid for probability distributions. Persistent homology allows for the continued use of clusters, but with the added benefit of capturing additional topological features that persist over a wide range of filtration values.

If such a persistent topological feature exists in the attention mechanism for specific input text, we could replace it in a principled manner with one or more representatives, reducing the amount of information necessary to represent the attention mechanism applied to that input text. This concept might be incorporated into model training, potentially leading to more efficient text representation.

7.2. **Temporal Coherence.** Considering the addition of tokens to an input in a sequential fashion as "time" passing, with one discrete time step per token, we can ask questions such as "Which topological features might persist in time?" and "How might the persistent homology for a text corpus change over time?" Utilizing discrete time and associating a persistence diagram with each time segment allows us to model the topology of language as time progresses. This approach introduces an entirely new mathematical perspective for studying linguistics.

7.3. **Language Translation.** Do topological features in language persist after translation into another language? Can we identify persistent homology fingerprints that are similar, or that undergo a principled transformation when translating from one language to another? Are there unique structures specific to certain languages? These are all crucial questions that arise when employing persistent homology to study language through the attention mechanism of transformers. Investigating the topological properties of various models can also reveal how different models represent languages more efficiently than others. In particular, we might discover persistent topological features that exist in some models but not in others, shedding light on their efficiency in representing languages.

7.4. **Sentiment Analysis and Contextual Mapping.** Now, recall our explanation of the context vectors: Sure, let's revisit the function formulation of self-attention in the context of the Transformer architecture. Given an input sequence of tokens represented by the matrix X, we can view it as a vector-valued function $f : S \to \mathbb{R}^d$ that maps the indices of the tokens in the set $S = \{1, \ldots, n\}$ to their corresponding d-dimensional embeddings.

In the self-attention mechanism, we compute the query, key, and value matrices Q, K, and V by multiplying the input matrix X with weight matrices W^Q, W^K, and W^V. Using function notation, we can represent these operations as linear maps:

$$\phi_{qry} : L_{\mathbb{R}^d}(S) \to L_{\mathbb{R}^{d_k}}(S)$$
$$\phi_{key} : L_{\mathbb{R}^d}(S) \to L_{\mathbb{R}^{d_k}}(S)$$
$$\phi_{val} : L_{\mathbb{R}^d}(S) \to L_{\mathbb{R}^{d_v}}(S)$$

These linear maps are applied to the function f, which results in the following functions for queries, keys, and values:

$$q_i = (\phi_{qry} \circ f)(i)$$
$$k_j = (\phi_{key} \circ f)(j)$$
$$v_j = (\phi_{val} \circ f)(j)$$

The attention scores are computed using the dot product of the query and key vectors, normalized by the square root of the key dimension:

$$s_{ij} = \frac{\langle q_i, k_j \rangle}{\sqrt{d_k}}$$

Applying the softmax function, we can obtain the attention matrix:

$$\text{AttentionMatrix}_{ij} = \frac{\exp(s_{ij})}{\sum_{t \in S} \exp(s_{it})}$$

Finally, the context vectors for each token can be computed by multiplying the attention matrix with the value vectors:

$$c_i = \sum_{j \in S} \text{AttentionMatrix}_{ij} \cdot v_j$$

It is likely that certain sentiments have a topological signature or shape that is detectible with deep learning methods. In this subsection, we investigate the topological signatures of various sentiments, which can potentially be detected using deep learning methods. By analyzing the 1-skeleton of simplicial complexes obtained from persistent homology, we observe common structures across different layers and heads of transformer models, such as GPT-2 and BERT, for a range of Jensen-Shannon distance thresholds. This observation is made for different text corpora containing common tokens, suggesting that the detection of these topological structures may be indicative of the models' ability to produce expressive contextual mappings.

To further illustrate this concept, we provide a code example implementing "contextual mapping" as defined in [YBRRK]. We closely follow the definition and framework presented in [YBRRK] for embedding two sentences, "I am happy" and "I am Bob". These sentences are represented as sequences of embeddings: X for the first sentence and X' for the second sentence, with individual tokens ('I', 'am', 'happy', and 'Bob') having corresponding d-dimensional embeddings (v_I, v_{am}, v_{happy}, and v_{Bob}). The notion of contextual mapping is important because the word 'I' appears in both sentences, but its meaning or context is different in each case. To ensure the seq2seq

model can handle these different contexts, it should map the two occurrences of 'I' to different values.

A contextual mapping (denoted as q) is a function that maps a sequence of embeddings (L) to a unique value ($\mathbb{R}^{1n}$), ensuring that the same token in different contexts gets different values. The definition in [YBRRK] presents two conditions for a proper contextual mapping:

(1) For any $L \in \mathbb{L}$, the n entries in $q(L)$ are all distinct.
(2) For any $L, L' \in \mathbb{L}$, with $L \neq L'$, all entries of $q(L)$ and $q(L')$ are distinct.

By meeting these conditions, the contextual mapping function q can capture the precise context of each token in a sequence. This is important because it enables the seq2seq model to understand and differentiate between the contextual meanings of words in different sentences.

The text also mentions that this contextual mapping can help subsequent token-wise functions, such as the feed-forward layers in Transformer networks, to realize the outputs of any arbitrary sequence-to-sequence functions. In other words, by ensuring that tokens in different contexts are properly distinguished, the seq2seq model can be more effective in tasks like machine translation, sentiment analysis, or text summarization, among others.

Now, we note, it may be the case that having clusters that are clearly visible to a human eye in various different contexts but with different vector representations each time is an indication of an expressive model with good contextual mapping abilities. It shows the model can map these token embedding vectors to different contexts and is at the same time still able to cluster them in a meaningful way in different contexts. By examining the simplicial complexes derived from the model's embedding space, we can analyze the topological structures and identify persistent features, such as connected components, loops, or higher-dimensional structures, which are preserved across a range of distance thresholds.

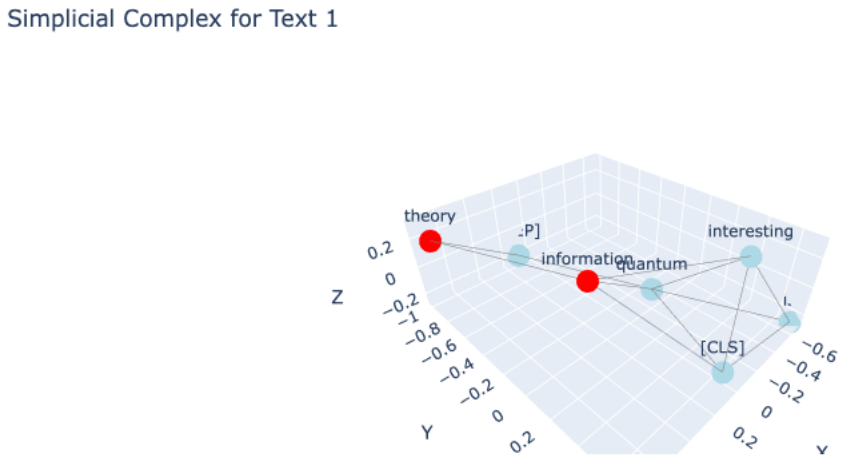
In the context of natural language processing and transformer models, persistent homology analysis can provide insights into the topological properties of the embedding space generated by the model. By identifying persistent topological features in the embeddings produced by different layers and heads of the model, we can gain a deeper understanding of the model's ability to capture context and represent the underlying structure of the input data.

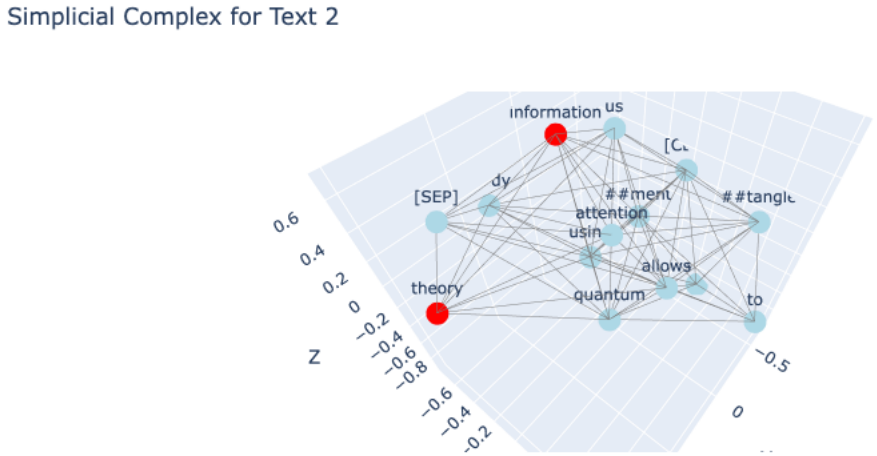
One potential application of persistent homology analysis in this context is to investigate the model's ability to create distinct and meaningful clusters in the embedding space. If the model is able to generate clearly visible clusters for different contexts while maintaining different vector representations for each node, it suggests that the model can effectively map token embeddings to different contexts while still preserving meaningful relationships between tokens.

This ability to create meaningful clusters can be assessed by studying the subcomplexes of the simplicial complexes derived from persistent homology analysis. If we observe common subcomplexes across different layers and heads, but with different vector representations for individual nodes, it implies that the model is capable of representing tokens with similar meanings in a coherent manner while still distinguishing their unique contexts.

Moreover, by analyzing the persistence diagrams of these topological features, we can quantify the model's ability to capture context-dependent relationships and its robustness to changes in distance thresholds. In this way, persistent homology offers a valuable tool to assess the performance of transformer models in generating expressive and context-dependent embeddings.

8. Linguistic Analysis and Expressivity of Attention Heads

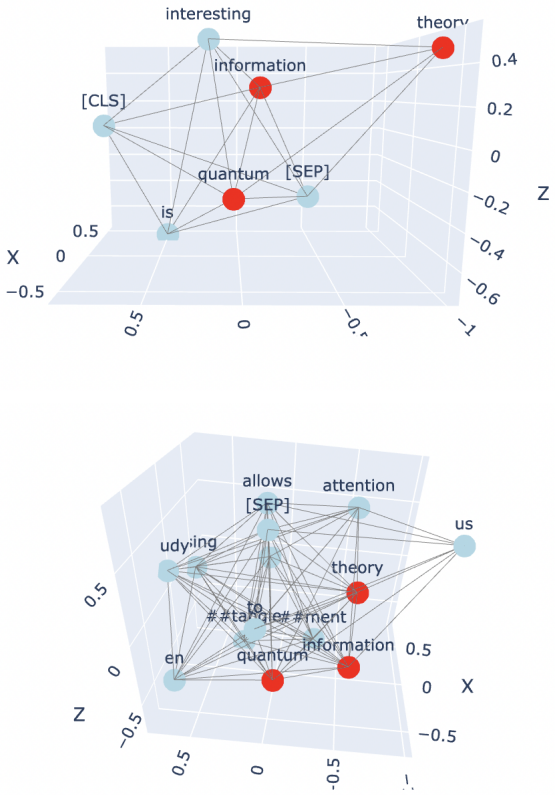




So far, we know using the Jensen-Shannon distance metric to form a distance matrix of pairwise distances between the distributions associated to tokens by the softmax of the attention matrix, gives a way to analyze the text input and the attention mechanism using persistent homology. In particular we can plot the 1-skeleton of the simplicial complex, and plot the associated persistence diagrams and barcode. We would then like to identify meaningful linguistic structures that might be encoded in the persistent homology. For example, in the following images, we see the 1-skeleton for the

simplicial complex obatined from two different text inputs in the BERT model, for layer 1, head 2. The JS-distance threshold is set to 0.2. We have highligted two nodes corresponding to the tokens ”information” and ”theory”, from the two text inputs ’Quantum information theory is interesting’ and ’Quantum information theory allows us to study attention using entanglement’. We see the two nodes become connected very near the threshold value of 0.2, and once connected the remain connected for all values greater than 0.2.

This simple analysis illustrates that while the context may change, the topological structure of some subset of tokens may not. This may or may not be a good thing. From a linguistic perspective it is fascinating. It may also be related to the expressivity of the model. It is to be determined whether or not persistent topological features are correlated with expressivity of the model, or of the particular attention head or layer in that model. Below, as another example we see a simplex present for both text inputs, that is born near the threshold value of 0.18 (the birth parameter value for the second text input).



References

[BL] Magnus Bakke Botnan, Michael Lesnick, *An Introduction to Multiparameter Persistence*, https://arxiv.org/abs/2203.14289.

[HBB] Loic Herviou, Soumya Bera, Jens H. Bardarson, *Multiscale entanglement clusters at the many-body localization phase transition*, https://arxiv.org/abs/1811.01925

[HOST] Heather A. Harrington, Nina Otter, Hal Schenck, Ulrike Tillmann. *Stratifying multiparameter persistent homology* https://arxiv.org/abs/1708.07390

[Ho] M. Hochster, *Topics in the Homological Theory of Modules over Commutative Rings*, CBMS Regional Conference Series, No. 24 (1974). https://bookstore.ams.org/cbms-24

[LW] Michael Lesnick, Matthew Wright, *Computing Minimal Presentations and Bigraded Betti Numbers of 2-Parameter Persistent Homology*.

[O] Bart Olsthoorn, *Persistent homology of quantum entanglement*, https://arxiv.org/abs/2110.10214

[Rivet] Michael Lesnick, Matthew Wright, Madkour Abdel-Rahman, Anway De, Bryn Keller, Phil Nadolny, Simon Segert, David Turner, Alex Yu, Roy Zhao, *Rivet*, https://github.com/rivetTDA/rivet/

[RC] David W. Romero, Jean-Baptiste Cordonnier, *Group Equivariant Stand-Alone Self-Attention For Vision*, https://arxiv.org/abs/2010.00977

[AS] Amelie Schreiber, *Multiparameter Persistent Homology: Generic Structures and Quantum Computing*, http://arxiv.org/abs/2210.11433

[SJENMC] Javier Selva1, Anders S. Johansen, Sergio Escalera1, Kamal Nasrollahi, Thomas B. Moeslund, Albert Clapes, *Video Transformers: A Survey*, https://arxiv.org/pdf/2201.05991.pdf

[YBRRK] Chulhee Yun, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank J. Reddi, Sanjiv Kumar, *Are Transformers universal approximators of sequence-to-sequence functions?*, https://arxiv.org/abs/1912.10077

Email address: amelie.schreiber.math@gmail.com