TOPOLOGICAL ANALYSIS OF TRANSFORMERS

AMELIE SCHREIBER

Abstract. In what follows we give a topological analysis of transformers using tools from persistent homology to better understand the context vectors computed by individual attention heads, as well as the probability distributions obtained from the softmax of the attention matrix in each attention head. From this we gather several tools for analyzing the various models we study. We use the simplex tree to analyze how language models parse sentences. We use persistence diagrams to analyze semantic similarity of text corpora. Frèchet mean diagrams of baseline texts compared to potential anomalous diagrams are used to perform anomalous text detection. We also discuss how certain models perform at preserving persistent homology and compare them to one another, showing that some models perform better than others at this consistently across most attention heads. We also discuss how one performs document clustering and topic modelling with persistent homology, and some various tentative applications including to text-to-video and vision transformers.

Contents

1. Introduction

1.1. **Multiword Expressions, Collocations, Idioms, and Keyphrases.** While we go beyond idioms, considering multiword expressions and collocations that have no hidden non-literal meanings, they serve as a good example of the kinds of keyphrases we wish to focus on in text corpora. They represent groups of words that appear together more often than would be expected by chance, and may have a meaning which is not easily understood from the meanings of the individual words that make them up. They also serve as a difficulty in the context of machine understanding of text and

Date: June 19, 2023.
2020 *Mathematics Subject Classification.* Primary 68T07, 68T50 68T10, Secondary 68T45,

machine translation. So, we would like to study how they behave from a new perspective. While we do not restrict ourselves to Chinese, and take a more multilingual approach, the following serves as a good example of why this problem is of interest:

"Idioms are different from ordinary Chinese characters in that they have semantic unity and structural persistence; that is, they are semantically indivisible as a whole, and their overall meaning cannot be speculated from the individual words that make up the idiom. In terms of structure, the order cannot be changed casually, let alone the grammatical structure. This leads to the low accuracy of the existing models for idiom machine reading."[DLYF]

In [DLYF], the authors present an architectural modification to the usual transformer that captures a somewhat more hierarchical representation of idiomatic expressions. However, rather than propose an architectural change to the transformer model, we propose a modification of the objective, possibly only for Low Rank Adaptation (LoRA) of a pretrained model [Hu et. al.], for downstream tasks where extracting and understanding collocations and multiword expressions such as idioms might be important, such as machine translation. In particular, we propose including the persistent homology of collocations and multiword expressions (MWEs) in the objective for LoRA modified attention heads. Or, more generally, we posit that preserving the topology of a subset of a text corpus when the context is changed, may in fact be important to performance on various tasks related to information extraction and translation.

Our hypothesis is supported by preliminary experiments that show models known to perform better at translation, also perform better at preserving the topology, or more accurately, the persistent homology, of subsets of text when the context is changed. As an illustration, take the two sentences, *"Quantum information theory is an interesting field of study."* and *"Quantum information theory can inspire new deep learning methods."* The collocation *"Quantum information theory"* appears in both contexts (that is, both sentences) and due to differing contexts, the contextual embeddings obtained for the tokens for *"Quantum information theory"* will be different, that is, the context vectors for each token in the phrase will change depending on the context they are found in. However, we find that the topology of the (filtered) simplicial complex they make up, when their context vectors are treated as a point cloud, is preserved much better by models known to perform better at certain language tasks such as multilingual translation. Thus, we expect that preserving the persistent homology of the point clouds associated to some common subset of multiple text corpera may be an important behavior for models to learn in order to perform better at certain language modeling tasks like translation, keyphrase extraction, text summarization, and clustering text corpera based on content or sentiment.

2. Context Vectors

The context vector is a critical part of the multihead attention mechanism in transformers. As indicated by the provided formula:

$$\mathbf{c}_i^{l,h} = \sum_{j=1}^{n} a_{i,j}^{l,h} \mathbf{v}_j^{l,h}$$

the context vector $\mathbf{c}_i^{l,h}$ for the h^{th} head in layer l is computed as a weighted sum of the value vectors $\mathbf{v}_j^{l,h}$, where the weights are the attention scores $a_{ij}^{l,h}$. Here, n is the number of value vectors, corresponding to the number of input tokens for the attention mechanism.

The attention scores $a_{ij}^{l,h}$ for attention head h in layer l, themselves are computed using the query and key vectors. In the scaled dot-product attention mechanism typically used in transformers, the attention score between the i^{th} query $\mathbf{q}_i^{l,h}$ and the j^{th} key $\mathbf{k}_j^{l,h}$ is computed as:

$$a_{ij}^{l,h} = \frac{\exp(\mathbf{q}_i^{l,h} \cdot \mathbf{k}_j^{l,h} / \sqrt{d})}{\sum_{m=1}^{n} \exp(\mathbf{q}_i^{l,h} \cdot \mathbf{k}_m^{l,h} / \sqrt{d})}$$

where d is the dimensionality of the queries and keys, and $\cdot$ denotes the dot product. The division by $\sqrt{d}$ is a scaling factor that is used to prevent the dot product from growing too large in magnitude, which could lead to vanishing gradients during training. The softmax function is applied to the raw attention scores to ensure that they sum up to 1, allowing them to be interpreted as probabilities or relative importances.

The queries, keys, and values are themselves computed by applying learned linear transformations to the input embeddings. If $\mathbf{x}_i$ denotes the input embedding for the i^{th} token, then we have:

$$\begin{aligned}\mathbf{q}_i^{l,h} &= W_Q^{l,h}\mathbf{x}_i \\ \mathbf{k}_i^{l,h} &= W_K^{l,h}\mathbf{x}_i \\ \mathbf{v}_i^{l,h} &= W_V^{l,h}\mathbf{x}_i\end{aligned}$$

where $W_Q^{l,h}$, $W_K^{l,h}$, and $W_V^{l,h}$ are the weight matrices for the queries, keys, and values, respectively, for the h^{th} head in the l^{th} layer.

The context vectors $\mathbf{c}_{i,l}^h$ provide a summary of the input tokens, weighted by their relevance to the query. They can be thought of as a form of "contextualized" embedding, where the context is determined by the other tokens in the input sequence and their interaction weight given by the attention matrix $a_{ij}^{l,h}$ for head $h \in \{1, 2, ..., \mathcal{H}\}$. The multihead attention mechanism allows the model to capture different types of relevance or "attention" by using multiple heads, each with its own learned linear transformations.

In the multihead attention mechanism, the attention operation is not performed just once, but multiple times in parallel. The queries, keys, and values are transformed with different learned linear projections to $\mathcal{H}$ different sets of queries, keys, and values, where $\mathcal{H}$ is the number of heads. Then the attention mechanism is applied to each of these sets, yielding h output vectors, which are then concatenated and linearly transformed to result in the final output.

Let $W_Q^{l,h}$, $W_K^{l,h}$, and $W_V^{l,h}$ denote the weight matrices for the h^{th} head for the queries, keys, and values, respectively, in the l^{th} layer, and let W_O^l denote the output weight matrix for layer l. Then the output of the multihead attention mechanism for layer l is computed as:

$$\mathbf{c}_i^l = W_O^l[\mathbf{c}_i^{l,h}; \mathbf{c}_i^{l,h}; \ldots; \mathbf{c}_i^{l,\mathcal{H}}]$$

where $\mathbf{c}_i^{l,h}$ is the output of the attention mechanism for the h^{th} head in layer l, for token x_i, computed as:

$$\mathbf{c}_i^{l,h} = \sum_{j=1}^{n} a_{ij}^{l,h}\mathbf{v}_j^{l,h}$$

Each transformer layer comprises a multihead self-attention mechanism, followed by layer normalization, a position-wise feed-forward network, and another layer normalization. Let's denote the l^{th} transformer layer in the model, where l ranges from 1 to L (with L being the total number of layers).

(1) Multihead Self-Attention Mechanism: The multihead self-attention mechanism in the l^{th} layer operates on the input embeddings $\mathbf{x}_i^l$, transforming them into queries $\mathbf{q}_i^{l,h}$, keys $\mathbf{k}_i^{l,h}$, and values $\mathbf{v}_i^{l,h}$ for each head h as before. Then the context vectors are computed, and concatenated as before, and the weight matrix W_O^l is applied to get $\mathbf{c}_i^l$

(2) Layer Normalization: Layer normalization stabilizes the learning process and reduces internal covariate shift by normalizing the multihead self-attention output across the hidden dimension. For the l^{th} layer, this is computed as:

$$\mathbf{c'}_i^l = \frac{\mathbf{c}_i^l - \mu^l}{\sigma^l}$$

where μ^l and σ^l are the mean and standard deviation of the layer outputs, computed as:

$$\mu^l = \frac{1}{H}\sum_{i=1}^{H} c_i^l$$

$$\sigma^l = \sqrt{\frac{1}{H}\sum_{i=1}^{H}\left(c_i^l - \mu^l\right)^2}$$

(3) Position-Wise Feed-Forward Network (FFN): The output of the layer normalization is then passed through a position-wise feed-forward network (FFN). The

The output of the layer normalization is then passed through a position-wise feed-forward network (FFN). The FFN consists of two linear transformations with a ReLU activation in between. For each position i, the FFN is applied to $\mathbf{c'}_i^l$ independently. Let's denote the weight matrices and bias vectors of the two linear transformations as W_1^l, b_1^l, W_2^l, and b_2^l, respectively. Then the output of the FFN is computed as:

$$\mathbf{d}_i^l = W_2^l \mathbf{max}(0, W_1^l \mathbf{c'}_i^l + b_1^l) + b_2^l$$

where $\max(0, x)$ denotes the ReLU activation function.

(4) Second Layer Normalization: Finally, the output of the FFN goes through another layer normalization step to compute the final output of the transformer layer:

$$\mathbf{o}_i^l = \frac{\mathbf{d}_i^l - \mu^l}{\sigma^l}$$

where again, μ^l and σ^l are the mean and standard deviation of the layer outputs, computed similarly as before but now with $\mathbf{d}_i^l$ instead of $\mathbf{c}_i^l$.

3. Application of Clustering Algorithms on Context Vectors

In the field of computational linguistics, the concept of context vectors, as generated by models like Transformers, has proved to be critical in capturing the semantic and syntactic nuances of language. These context vectors, which provide a representation of each token in the input sequence in terms of its context, can be further processed to extract meaningful patterns and relationships. In this section, we will discuss the application of clustering algorithms, specifically Density-Based Spatial Clustering of Applications with Noise (DBSCAN), and its topological data analysis cousin, persistent homology, on these context vectors.

3.1. **Context Vectors and Clustering.**

Given a Transformer model with L layers and $\mathcal{H}$ attention heads, for any token x_i in the input sequence, we can obtain a set of $L \times \mathcal{H}$ context vectors $\mathbf{c}_i^{l,h}$, with each context vector corresponding to a different attention head in a different layer.

For a given layer l and head h, the context vector $\mathbf{c}_i^{l,h}$ is a d-dimensional vector (where d is the dimensionality of the embeddings), which can be thought of as a point in a d-dimensional space. Thus, for all tokens in the input sequence, we can obtain a set of points in this d-dimensional space.

Clustering can be applied to this set of points to group together tokens that have similar context vectors, and hence, are likely to have similar semantic and syntactic roles. This can be useful for various applications, such as identifying synonyms, identifying words that often occur in similar contexts, or even for interpretability, by identifying what types of tokens each head and layer of the transformer is paying attention to.

3.2. **Density-Based Spatial Clustering of Applications with Noise (DBSCAN).** DBSCAN is a density-based clustering algorithm that groups together points that are densely packed in the input space. The algorithm is defined by two parameters: ϵ, which defines the neighborhood around a point, and $minPts$, which defines the minimum number of points required to form a dense region.

Given a set of n context vectors $\mathbf{c}_i^{l,h}$, $i \in \{1, 2, ..., n\}$, the DBSCAN algorithm can be applied as follows:

(1) For each point $\mathbf{c}_i^{l,h}$, compute the ϵ-neighborhood $N_\epsilon(\mathbf{c}_i^{l,h})$, which is the set of points that are within a distance ϵ of $\mathbf{c}_i^{l,h}$.
(2) For each point $\mathbf{c}_i^{l,h}$, if $|N_\epsilon(\mathbf{c}_i^{l,h})| \geq minPts$, mark $\mathbf{c}_i^{l,h}$ as a core point. Otherwise, if $\mathbf{c}_i^{l,h}$ is within the ϵ-neighborhood of some core point, mark $\mathbf{c}_i^{l,h}$ as a border point. All other points are marked as noise points.
(3) Form a cluster for each core point and its associated border points. Two core points are in the same cluster if they are within a distance ϵ of each other.

The resulting clusters group together tokens with similar context vectors, which can be used to identify patterns and relationships in the input data.

3.3. **Persistent Homology.** Persistent homology is a method from the field of topological data analysis that can be used to identify and quantify the shape or structure of a dataset. It can be thought of as a multi-scale analysis that identifies "features" of the data that persist over a range of scales.

Given a set of n context vectors $\mathbf{c}_i^{l,h}$, $i \in \{1, 2, ..., n\}$, the persistent homology analysis can be carried out as follows:

(1) Construct a simplicial complex K from the context vectors, where each vector is a 0-simplex (a point), and a k-simplex (a k-dimensional triangle) is formed whenever $k + 1$ points are within a distance ϵ of each other.
(2) For each $\epsilon \geq 0$, compute the homology groups $H_k(K_\epsilon)$ of the ϵ-sublevel set K_ϵ, which is the subcomplex of K consisting of all simplices that can be formed with a distance less than ϵ.
(3) For each homology group $H_k(K_\epsilon)$, compute the Betti numbers $\beta_k = \text{rank}(H_k(K_\epsilon))$, which count the number of k-dimensional "holes" in the data.

The result of this analysis is a set of "persistence diagrams", one for each k, which plot the birth and death scales of each feature (where the birth scale is the smallest ϵ at which the feature appears, and the death scale is the smallest ϵ at which the feature disappears).

These persistence diagrams can be used to identify and quantify the "shape" or "structure" of the context vectors. For instance, a high β_0 indicates a large number of disconnected clusters, a high β_1 indicates a large number of loops or cycles, and so on.

3.4. **Applications.** Clustering of context vectors using methods like DBSCAN and persistent homology can have a variety of applications in computational linguistics and natural language understanding:

(1) **Semantic and Syntactic Analysis:** Clusters of context vectors can reveal groups of tokens that are used in similar semantic or syntactic roles. For example, words that are synonyms or words that belong to the same part of speech might cluster together.

(2) **Model Interpretability:** Clustering can help to understand what features each head and layer of the transformer model is paying attention to. For example, certain heads might focus more on syntactic features, while others might focus more on semantic features.
(3) **Anomaly Detection:** Outliers in the context vectors, which would be identified as noise points by DBSCAN or as short-lived features by persistent homology, can indicate tokens that the model is having difficulty processing.
(4) **Model Compression:** Clustering can be used to identify redundancies in the model, such as heads or layers that produce similar context vectors with similar persistent topological features, which can be removed to reduce the size of the model without significantly impacting performance.

4. Semantic and Syntactic Analysis and Model Interpretability

In preliminary studies, it has been shown that using the persistent homology of a single attention head's context vectors can extract collocations and keyphrases from the text, providing a connection between analyzing the linguistic properties of the text and interpreting the model's behavior. To be more precise, it can be shown empirically that using a persistent homology barcode diagram, obtained from the persistent homology analysis of the context vectors computed by a single attention head, can be used to inform which distance threshold parameter to choose for a DBSCAN algorithm (also applied to the context vectors). Using this persistent homology informed BDSCAN of the context vectors, clusters of collocations and keyphrases were detected at a specific scale which correspond to low-lying H_1 persistence structures. The majority of these clusters contained few or no extraneous words and were made up only of collocations.

5. Topological Structures that Persist in Different Contexts

Suppose we are given a collocation or multiword expression (MWE) that appears in different contexts. The question arises, does the group of tokens representing the collocation or MWE have a topological structure that persists in different contexts? In other words, if we take the simplicial complex associated to the collocation or MWE, does the 1-skeleton with edges connecting nodes corresponding to tokens in our collocation or MWE, do the pairwise distances remain similar in multiple contexts? What about across languages before and after translation? As an example, take the multiword expression "*the tip of the iceberg*" in the following pargaph:

"In the realm of linguistics, it is crucial to foster a deep understanding of language structures and their interconnections. This understanding is not simply about memorizing vocabulary or grammar rules, but delving into the intricate details of phonetics, semantics, and syntax. However, the common elements that we often focus on, such as morphemes, phonemes, and syntax rules, are merely ***the tip of the iceberg****. Beneath the surface, there lies a vast field of cognitive, social, and historical factors that play significant roles in shaping a language. Grasping these underlying influences is what sets apart a competent linguist from a casual language learner."*

and in the paragraph:

"In the realm of deep learning, having a deep understanding of neural networks and their architectures is of paramount importance. It's not just about knowing how to code an algorithm, but also about comprehending how different layers in a network interact and learning to interpret the output. Yet, the algorithms and architectures we often discuss, like Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs), are just ***the tip of the iceberg****. Hidden below the surface are complex concepts like backpropagation, weight initialization, and optimization techniques, which truly drive the functionality of these networks. Mastering these intricacies is what differentiates an expert in the field from an amateur coder."*

How does the topology and the pairwise distances between the nodes representing the tokens for *"the tip of the iceberg"* change when we observe the tokens in the context of the first paragraph versus the second. How do different language translation models compare to one another in preserving this topology? What happens to the topological structure as we change languages? Is the topology easier to preserve in one language than in another? How do different heads both within models and across models compare to each other? These are some of the questions we aim to answer in this preliminary study.

Understanding the persistent homology of the 1-skeleton associated to a collocation or MWE is a difficult task. With persistent homology we can compare the persistence diagrams (or equivalently the barcode diagrams), associated to the distance matrix of pairwise Euclidean distances between the context vectors associated to the tokens. Comparing barcode diagrams requires a notion of a distance metric between barcode diagrams. We use both the bottleneck distance, as well as the Wasserstein distance[1] to compare barcode diagrams in our preliminary analysis. We also compare two multilingual models, **xlm-roberta-large**[2] and **bert-base-multilingual-cased**[3]. We also compare seven languages, some of which are considered morphologically complex, and some which are considered low resource, namely:

(1) English
(2) Yiddish
(3) Hebrew
(4) Russian
(5) German
(6) French
(7) Bengali

In our preliminary experiments we find that *xlm-roberta-large* consistently outperforms *bert-base-multilingual-cased* at preserving the persistent homology across languages, and across contexts within a fixed language. For a fair comparison, context size in terms of tokens was to roughly 150 ± 30 tokens. The effects of drastically different context sizes is not studied here. For *bert-base-multilingual-cased*, the persistence diagrams tended to have significantly larger distances for Yiddish, Hebrew, and Bengali, compared to the other four languages. For *xlm-roberta-large* the difference was less pronounced and seemed significant only for Yiddish (relative to the other six languages).

While this is only a preliminary set of experiments and examples, with a limited number of languages, the findings were very consistent and compelling.[4]. We also note another study of the same phenomena for Hebrew for the following models is also being conducted:

(1) bert-base-multilingual-cased
(2) heBERT
(3) xlm-roberta-large
(4) AlephBERT
(5) dicta-il/alephbertgimmel-base

All of these can be found on Huggingface. In this case, we find that *xlm-roberta-large*, *AlephBERT*, and *dicta-il/alephbertgimmel-base* consistently outperform *bert-base-multilingual-cased* and

[1]Wasserstein distance is also called the Kontorovich distance.

[2]This model is available on HuggingFace at https://huggingface.co/xlm-roberta-large

[3]This model is available on HuggingFace at https://huggingface.co/bert-base-multilingual-cased

[4]Code in the form of some example notebooks can be found at https://github.com/Amelie-Schreiber/multilingual_persistent_homology

heBERT at preserving the persistent homology across contexts, but that within the two groups we find no clear winner in our examples.

6. Some Heuristics: Comparing Barcode Diagrams

It appears, based on the limited experiments run so far, that the Wasserstein distance exhibits trends that the bottleneck distance does not always exhibit. Recall, given two persistence diagrams D_1 and D_2, the **bottleneck distance** d_B is defined as:

$$d_B(D_1, D_2) = \inf_{\phi: D_1 \to D_2} \sup_{x \in D_1} \|x - \phi(x)\|_\infty,$$

where ϕ ranges over all bijections between the diagrams D_1 and D_2, and $\|\cdot\|_\infty$ represents the L_∞ norm. Essentially, the bottleneck distance measures the maximum difference between matched points in the two diagrams. It is notably robust to noise, as it focuses on the worst-case discrepancies between the diagrams. However, it may fail to capture finer distinctions between diagrams that exhibit numerous smaller differences.

In contrast, the **Wasserstein distance** offers a more nuanced comparison between persistence diagrams. The p-th Wasserstein distance W_p between D_1 and D_2 is defined as:

$$W_p(D_1, D_2) = \left(\inf_{\gamma \in \Gamma(D_1, D_2)} \int_{\mathbb{R}^2 \times \mathbb{R}^2} \|x - y\|^p \, d\gamma(x, y) \right)^{1/p},$$

where $\Gamma(D_1, D_2)$ denotes the set of all couplings of D_1 and D_2, and $\|\cdot\|$ represents the Euclidean distance. The p-Wasserstein distance measures the minimum "effort" required to transform one diagram into another, considering all possible bijections. It provides a holistic assessment of the diagrams, accounting for smaller but numerous variations that the bottleneck distance might overlook. Equivalently, we can define the p-Wasserstein distance $d_W^p(D_1, D_2)$ as

$$d_W^p(D_1, D_2) = \left(\inf_{\sigma: D_1 \to D_2} \sum_{x \in D_1} \|x - \sigma(x)\|_p^p \right)^{1/p}$$

where σ ranges over all bijections from D_1 to D_2, and $\|\cdot\|_p$ denotes the p-norm. When $p = 1$, we get the 1-Wasserstein distance (also known as the Earth Mover's distance), which intuitively measures the amount of "work" needed to transform one diagram into another, where "work" is the "mass" of a feature times the distance it has to be moved. When $p = 2$, we get the 2-Wasserstein distance, which is more sensitive to outliers. The Wasserstein distance is less stable than the bottleneck distance, but it can capture more subtle differences between diagrams.

Importantly, the persistence diagram space equipped with the Wasserstein distance exhibits mathematical properties that make it amenable to statistical analyses. Specifically, the space is complete and separable under the Wasserstein distance, enabling the definition of statistical tools such as expectation, variance, and conditional probability in the Fréchet sense.

The selection between the Wasserstein and bottleneck distances is not a one-size-fits-all decision; rather, it depends on the data's characteristics and the specific research question:

- **Bottleneck Distance**: Choose this metric when dealing with noisy data or when the interest is predominantly in the most significant topological features, as it is less sensitive to small, numerous variations.
- **Wasserstein Distance**: Opt for this metric when analyzing clean data, where capturing subtle differences across diagrams is essential. Moreover, if the aim is to perform statistical analysis on the diagrams, the Wasserstein distance is the preferred choice due to its favorable mathematical properties.

In essence, both the Wasserstein and bottleneck distances provide distinct yet complementary insights into the comparison of persistence diagrams. The choice between these two distances should be informed by the nature of the data and the research objectives at hand. Since we are using very high dimensional context vectors, and our tokens are a subset of collections of around 150 or so other tokens, it is not surprising that the Wasserstein distance proves more useful in capturing trends between languages, between models, and between individual attention heads.

6.1. **Comparing Models.** Comparing how well the persistent homology of a keyphrase, multiword expression, or collocation is preserved across contexts by different heads in different models is not a simple task. To do so, we need a compreshensive comparison of each head in one model, to each head in a second model, across a range of contexts and for a range of keyphrases, multiword expressions, and collocations. To do this, we set up the following pipeline:

(1) Choose a keyphrase, multiword expressions, or collocation.
(2) Produce a list of contexts (texts) containing the keyphrase, multiword expressions, or collocation exactly once.
(3) Compute the context vectors for the keyphrase, multiword expressions, or collocation in each context for each head in each layer of the first model, and for the second model.
(4) Compute the persistent homology of the context vectors of the keyphrase, multiword expressions, or collocation for the first model and for the second model.
(5) Compute the Wasserstein distances between each pair of persistence diagrams for the first model (that is across all contexts), and for each attention head, and then do the same for the second model.
(6) Compare the distance matrix obtained in this way for each head of the first model to the distance matrix obtained in this way for each head of the second model.
(7) Compute the percentage of entries below the diagonal that are negative, effectively computing what percentage of the time the second model has higher Wasserstein distance compared to the first model.
(8) Compute statistics on the array of probabilities to determine the minimum, maximum, median, mean, variance, standard deviation, and quantiles to determine just how well one model preserves persistent homology compared to the other. [5]

We find that in all comparisons *dicta-il/alephbertgimmel-base* outperforms all other models, which are drawn from the following list:

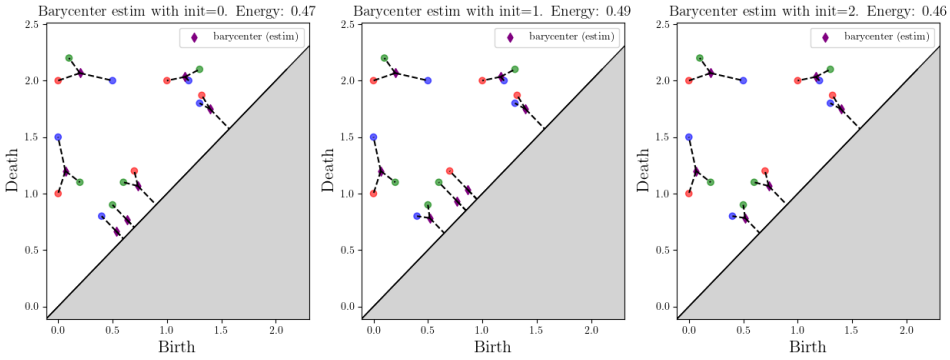
- *onlplab/alephbert-base*
- *bert-base-multilingual-cased*
- *TurkuNLP/wikibert-base-he-cased*

7. Frèchet Mean of Persistence Diagrams

In this section we discuss the possibility of averaging multiple barcodes or persistence diagrams for the context vectors associated to a collocation or multiword expression. We do this in order to find an average topological structure of a collocation or multiword expression, represented in many different contexts with each context containing roughly the same number of tokens. Finding such an average topological structure would then allow us to use its persistence diagram or barcode as a reference point in the space of all persistence diagrams when including persistent homology in the objective of a large language model during pretraining, effectively encouraging the model to learn contextual representations of the collocation or multiword expression that have similar persistent topological features to this mean topological structure. To do this, we need the concept of Frèchet

[5] This is done for multiple models in the repo https://github.com/Amelie-Schreiber/context_persistent_homology_comparisons.

mean; since the space of persistence diagrams is not a linear space, we cannot simply take the arithmetic mean. [MMH, TMMH, LCO, DL] A generic example of the barycenter diagram using optimal transport and the topological data analysis library Gudhi can be seen below. We refer to the notebook for an example with *xlm-roberta-large*.[6]



Note, this does not restrict the individual context vectors for each token of the collocation or MWE, nor does it impose any direct restrictions on the other tokens' context vectors. It only imposes a restriction on the relative distances between the context vectors of the collocation or MWE, so that the persistent homology is preserved, relative to some average that can be computed for many different contexts. This will encourage the model to learn a topological representation of the context vectors that is more stable across different contexts, which as we have already seen, is a behavior that models better at many multilingual tasks exhibit.

Conjecturally, this may be important in multilingual tasks such as machine translation, as models that are known to perform better at preserving the persistent homology also perform better at translation tasks. This could provide a way of boosting performance of smaller models that have not learned topologically stable representations of collocations and multiword expressions. Note, this could be used to train or fine tune single attention heads that detect collocations, MWE, or idioms, and using Low Rank Adaptations we may be able to accomplish this even after the fine-tuning phase, especially if we find that some subset of attention heads already preserved the persistent homology of collocations, or MWEs better than others.

8. 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:

[6]See the following repo https://github.com/Amelie-Schreiber/multilingual_persistent_homology

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

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., $\text{JSDiv}(P, Q) = \text{JSDiv}(Q, P)$, and non-negative, i.e., $\text{JSDiv}(P, Q) \geq 0$.

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:

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

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.

8.1. **Probability Distributions from Tokens.** Now, taking X_i the i^{th} row of the matrix of token embedding vectors X, for any given attention head, we denote by

$$\langle q_i, k_j \rangle = (W^Q X_i)(W^K X_j)^T. \tag{4}$$

Then we have,

$$P(X_i) = \left(\text{softmax}_j\left(\frac{\langle q_i, k_j\rangle}{\sqrt{d}}\right)\right)_{j=1}^{n} = \text{softmax}\begin{pmatrix} \frac{\langle q_i,k_1\rangle}{\sqrt{d}} \\ \frac{\langle q_i,k_2\rangle}{\sqrt{d}} \\ \vdots \\ \frac{\langle q_i,k_n\rangle}{\sqrt{d}} \end{pmatrix} = \left(\frac{e^{\frac{\langle q_i,k_j\rangle}{\sqrt{d}}}}{\sum_{l=1}^{n} e^{\frac{\langle q_i,k_l\rangle}{\sqrt{d}}}}\right)_{j=1}^{n}. \tag{5}$$

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:

$$KL(P(X_i)\|P(X_j)) = \sum_{k=1}^{n} P(X_i)_k \log_2 \frac{P(X_i)_k}{P(X_j)_k} \tag{6}$$

$$= \sum_{k=1}^{n} \frac{e^{\frac{\langle q_i,k_k\rangle}{\sqrt{d}}}}{\sum_{l=1}^{n} e^{\frac{\langle q_i,k_l\rangle}{\sqrt{d}}}} \log_2 \left(\frac{\frac{e^{\frac{\langle q_i,k_k\rangle}{\sqrt{d}}}}{\sum l=1^n e^{\frac{\langle q_i,k_l\rangle}{\sqrt{d}}}}}{\frac{e^{\frac{\langle q_j,k_k\rangle}{\sqrt{d}}}}{\sum_{l=1}^{n} e^{\frac{\langle q_j,k_l\rangle}{\sqrt{d}}}}} \right) \tag{7}$$

$$= \sum_{k=1}^{n} \frac{e^{\frac{\langle q_i,k_k\rangle}{\sqrt{d}}}}{\sum_{l=1}^{n} e^{\frac{\langle q_i,k_l\rangle}{\sqrt{d}}}} \left(\log_2 \left(\frac{e^{\frac{\langle q_i,k_k\rangle}{\sqrt{d}}}}{\sum_{l=1}^{n} e^{\frac{\langle q_i,k_l\rangle}{\sqrt{d}}}} \right) - \log_2 \left(\frac{e^{\frac{\langle q_j,k_k\rangle}{\sqrt{d}}}}{\sum_{l=1}^{n} e^{\frac{\langle q_j,k_l\rangle}{\sqrt{d}}}} \right) \right) \tag{8}$$

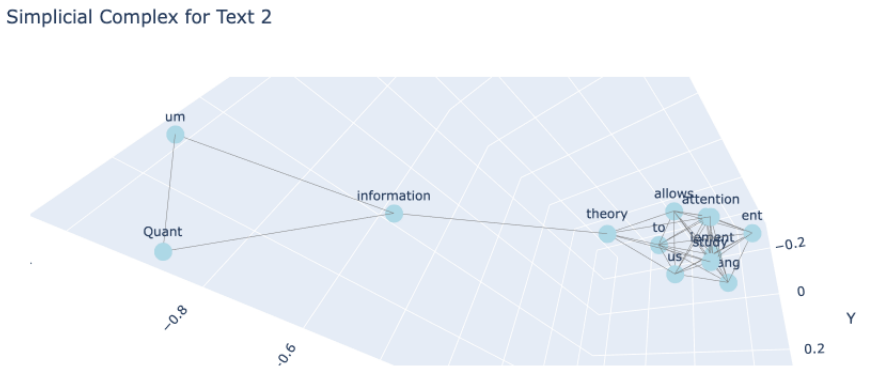
Next, we can compute the Jensen-Shannon distance between token probability distributions using

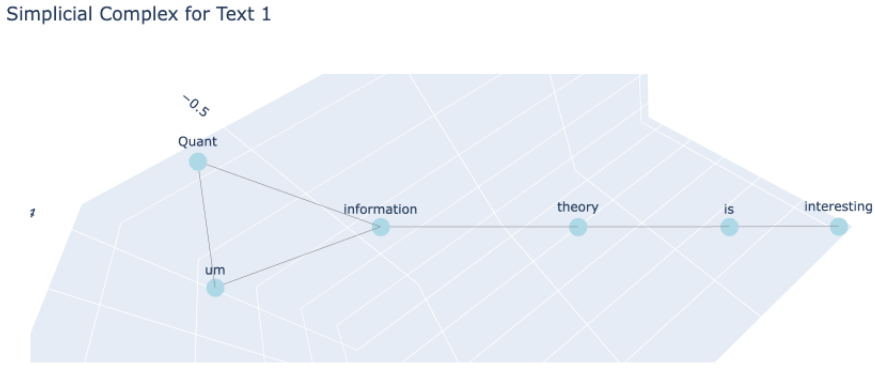
$$\text{JSDiv}(P, Q) = \frac{1}{2}\text{KL}(P\|M) + \frac{1}{2}\text{KL}(Q\|M), \tag{9}$$

and

$$\text{JSD}(P, Q) = \sqrt{\text{JSDiv}(P, Q)}. \tag{10}$$

This provides us with a genuine distance metric between probability distributions, which we can use to compute pairwise distances between some subset of token probability distributions. In so doing, we form a distance matrix which we can then use for computing persistent homology. The analysis then is very similar to our previous analysis of pairwise distances between context vectors. In particular, with a distance metric on the probability distribution we can see how the persistent homology of some subset of tokens changes as the context changes. As an example, we can look at the subset of tokens for "Quantum information theory" which are ['Quant', 'um', 'information', 'theory'] for GPT-2, and look at how they group together at a particular scale of the Jensen-Shannon distance metric, say 0.06:





Notice, in both contexts "*Quantum information theory is interesting.*", and "*Quantum information theory allows us to study attention with entanglement.*" we see a triangle (a 2-simplex) that forms between the nodes corresponding to the token ['Quant', 'um', 'information'] and a stem is created by the token node ['theory'] in the both plots. Such topological features persist for a range of scales for the Jensen-Shannon distance for attention probability distributions just as they do for context

vectors in the Euclidean distance metric.[7] While these topological features are present in both filtered simplicial complexes, that is, for both contexts, they persist for a different range of values for the Jensen-Shannon distance between token probability distributions, just as in the context vectors case for the Euclidean distance. So, we need to compare persistence diagrams (or barcode diagrams) to gain an accurate perspective on how persistent this feature is in different contexts.

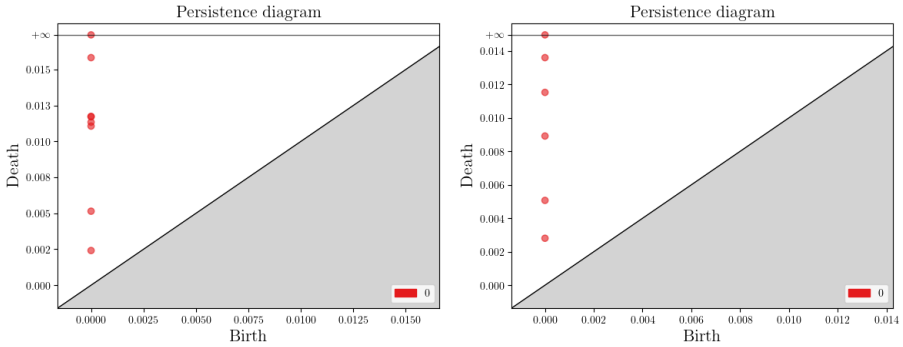
As another example, we place the subset text *"foster a deep understanding"* into different contexts, such as

"In the realm of linguistics, it is crucial to ***foster a deep understanding*** *of language structures and their interconnections. This understanding is not simply about memorizing vocabulary or grammar rules, but delving into the intricate details of phonetics, semantics, and syntax. However, the common elements that we often focus on, such as morphemes, phonemes, and syntax rules, are merely the tip of the iceberg. Beneath the surface, there lies a vast field of cognitive, social, and historical factors that play significant roles in shaping a language. Grasping these underlying influences is what sets apart a competent linguist from a casual language learner."*

and

"In the field of computer science, the aim is to ***foster a deep understanding*** *of algorithms and data structures. This knowledge goes beyond merely learning a programming language or writing a simple script. It involves diving into the core principles of computation and how data is manipulated to solve real-world problems. Basic coding skills and knowledge of a few algorithms are merely the tip of the iceberg. Beneath this surface, lies the unexplored depth of complexity theory, cryptography, machine learning, and more. Only by engaging with these advanced topics can one truly become proficient in the discipline of computer science."*

Then, computing the persistent homology of the probability distributions associated to say attention head 3 in layer 5 of GPT-2, we get two persistence diagrams:



We can now compute the Bottleneck, or Wasserstein distance between the two diagrams to see how well the persistent homology is preserved, finging a value of 0.02138564 for the Wasserstein distance. Comparing this to the value of 0.06569868 for *bert-base-multilingual-cased*, and the value of 0.00341813 for *xlm-roberta-large*, we see that there is a definitive ranking with *xlm-roberta-large* < GPT-2 < *bert-base-multilingual-cased*. This trend holds for several other contexts, making it clear that *xlm-roberta-large* consistently outperforms GPT-2, which outperforms *bert-base-multilingual-cased* at preserving the persistent homology of the attention probability distributions when changing the context in which the tokens for the subset text **foster a deep understanding** are embedded. For example, we find for four contexts the following distance matrix for GPT-2 between persistence diagrams:

[7] We can make similar plots of the 1-skeleton for context vectors and their filtered simplicial complexes given by persistent homology.

$$\begin{pmatrix} 0. & 0.02138564 & 0.01660325 & 0.02107501 \\ 0.02138564 & 0. & 0.03451074 & 0.00951788 \\ 0.01660325 & 0.03451074 & 0. & 0.02756028 \\ 0.02107501 & 0.00951788 & 0.02756028 & 0. \end{pmatrix}$$

Comparing this to *bert-base-multilingual-cased*:

$$\begin{pmatrix} 0. & 0.06569868 & 0.04190926 & 0.04404224 \\ 0.06569868 & 0. & 0.04042774 & 0.02504077 \\ 0.04190926 & 0.04042774 & 0. & 0.04991917 \\ 0.04404224 & 0.02504077 & 0.04991917 & 0. \end{pmatrix}$$

and to GPT-2-large:

$$\begin{pmatrix} 0. & 0.04486272 & 0.03353521 & 0.04105272 \\ 0.04486272 & 0. & 0.06252785 & 0.02158706 \\ 0.03353521 & 0.06252785 & 0. & 0.05065936 \\ 0.04105272 & 0.02158706 & 0.05065936 & 0. \end{pmatrix}$$

and to *xlm-roberta-large*:

$$\begin{pmatrix} 0. & 0.00341813 & 0.00055738 & 0.0015527 \\ 0.00341813 & 0. & 0.00305058 & 0.00261262 \\ 0.00055738 & 0.00305058 & 0. & 0.00129407 \\ 0.0015527 & 0.00261262 & 0.00129407 & 0. \end{pmatrix}$$

we can see that the trend *xlm-roberta-large* < GPT-2-Large < GPT-2 < *bert-base-multilingual-cased* continues. As another example, we take the phrase "the tip of the iceberg" in the same four contexts as before, and find that in all cases except one, the persistent homology is preserved better by GPT-2-large than the multlingual BERT model, that GPT-2 is slightly better than GPT-2 large, and that *xlm-roberta-large* outperforms all of the models in all cases.

9. Parsing and Simplex Trees

In order to understand how a model parses a sentence, we can leverage the notion of *persistent homology* - a concept from the field of computational topology that allows us to extract and quantify the topological features of a dataset at different scales. For the case of sentences processed by a transformer-based model such as BERT, we compute the persistent homology of the context vectors of an individual attention head to generate what we call a *simplex tree*. This tree provides a visual and mathematical representation of the hierarchical relationships between words in the sentence as understood by the model.

Persistent homology provides a multiscale analysis of a dataset by studying the evolution of its topological features (connected components, loops, voids, etc.) as a function of a particular parameter. In the case of context vectors, this parameter can be considered as a distance threshold. We start with all points (context vectors corresponding to tokens) isolated, and as we increase this threshold, points within the threshold distance get connected, forming higher-dimensional structures (edges, triangles, tetrahedra, etc.) called simplices.

Mathematically, given a set of vectors $V = \{v_1, v_2, \ldots, v_n\}$, we construct a *filtration* of simplicial complexes $K_1 \subset K_2 \subset \ldots \subset K_m$, where each K_i is a simplicial complex formed by connecting vectors within a distance d_i. The simplicial complex K_i is a collection of simplices, where a k-simplex is a set of $k + 1$ vectors that are pairwise connected.

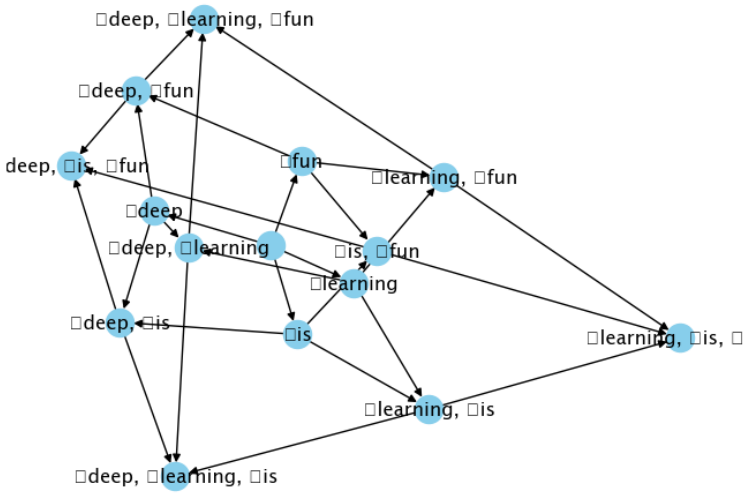
In other words, we start with K_1 where no vectors are connected (0-simplices), and as we increase the distance threshold, more vectors get connected forming 1-simplices (edges), 2-simplices (triangles), and so on.

A simplex tree is a data structure that efficiently encodes a simplicial complex. Each node in the tree represents a simplex in the complex, and the parent-child relationships in the tree correspond to the face-subface relationships between simplices. The root of the tree is the empty set, and the children of a node are the simplices that can be obtained by adding one vertex to the simplex corresponding to the node.

In the context of our study, each node in the simplex tree corresponds to a set of connected tokens (or a single token), and an edge between two nodes indicates that the set of tokens at the child node can be obtained by adding one token to the set at the parent node. The filtration value of a node in the simplex tree corresponds to the distance threshold at which the corresponding simplex appears in the filtration.

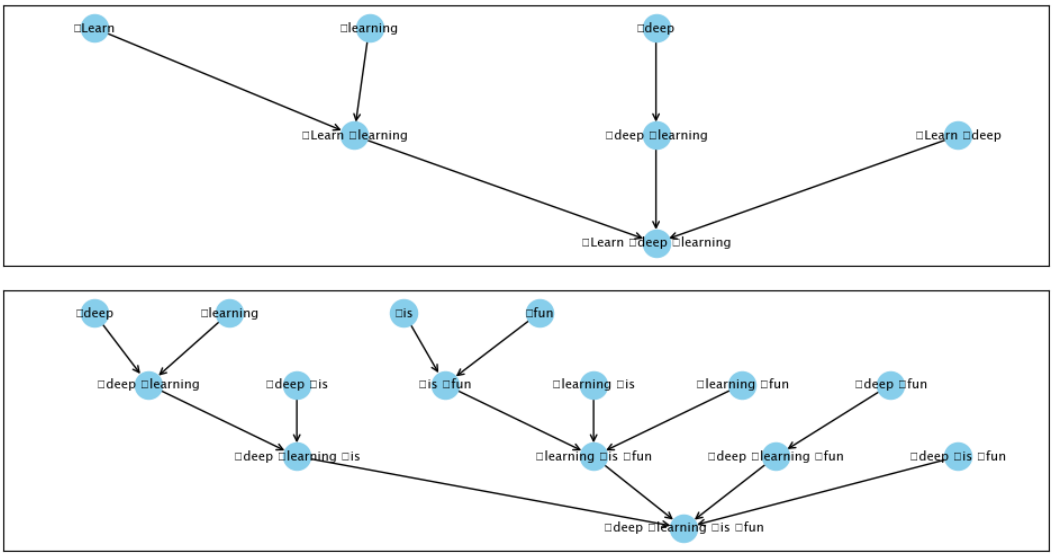
The structure of the simplex tree allows us to understand how the model is parsing the sentence. For example, an edge between nodes corresponding to the tokens ”the” and ”the cat” suggests that ”cat” is added to ”the” at a certain point in the parsing process. Moreover, the filtration values provide a measure of the relative importance of different relationships - the smaller the filtration value, the more significant the relationship. This is because simplices that appear earlier in the filtration (i.e., at lower distance thresholds) correspond to stronger connections between the corresponding vectors, and hence tokens.

The simplex trees naturally have a Hasse diagram structure, for example, the following is obtained from head 1 in layer 2 of *xlm-roberta-large*



This can be given a tree structure if we only include edges between consecutive dimensional simplexes that occur first in the filtration order. For example, we get the following tree from *xlm-roberta-large* for the texts, *”Learn deep learning”* and *”Deep learning is fun”*[8]

[8]The sentence simplex tree structures are computed in this notebook.



10. Tentative Applications

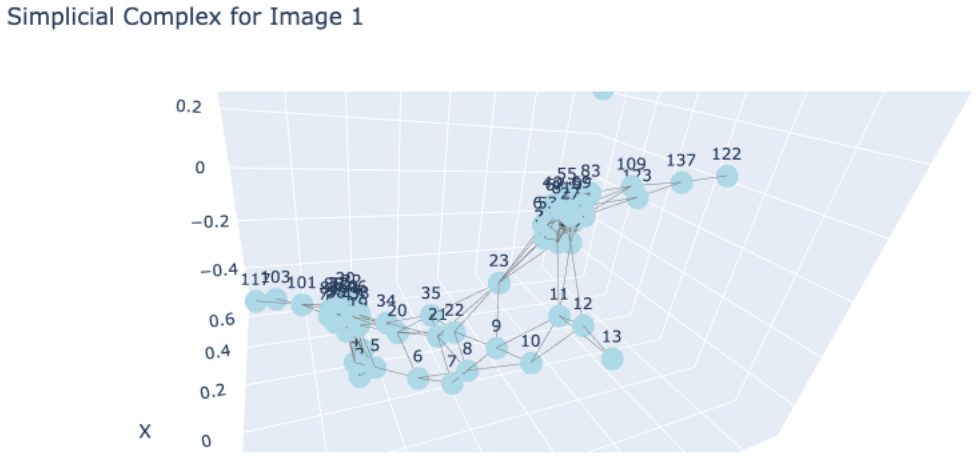
We note once again that preserving the topology does not explicitly restrict the values of the token probability distributions, it only imposes conditions on how they relate to each other in a *relative* way; so, relative to one another, the pairwise probability distributions values may change significantly depending on the context, but the pairwise (relative) distances may still have the very similar persistent homology, which can be measured by the bottleneck or Wasserstein distances.

10.1. **Text-to-Video.** Computing the persistent homology of the context vectors of image patches of $2D$-frames of a video and then using the Wasserstein distance as a measure of how close two consecutive frames are topologically would provide a way to maintain temporal coherence of videos generated by text-to-video models using transformers. We could also identify substructures in multiple contexts (multiple frames or multiple videos) analogous to subsets of text like keyphrases, and identify which models are preserving the topology of these substructures well. This would be analogous to identifying a particular attention head that understands certain multiword expressions or collocations better than others. In this way we could determine which attention heads are useful for maintaining coherence of certain kinds of substructures such as human forms.

Vision Transformers and Text-to-Image

When computing the persistent homology of the probability distributions of transformers for vision tasks such as ViT, BEiT, and DEiT, we find interesting topological structures in the persistent homology. We find that similar inamge have similar persistent homology at the same distance threshold. Below we see na image of the 1-skeleton of the simplicial complex at a fixed distance threshold for an image using ViT.[9]:

[9]The images from ViT were generated using this notebook



While the image is not easily interpetable, we can compute the Wasserstein distance between the persistence diagrams of two images to determine their similarity.[10] This provides us with some ideas of how to apply persistent homology to vision transformers and text-to-image models. The methods used in the code and the observations you described, regarding the similarities in the persistent homology of similar images, open up several interesting potential applications and use cases in the field of computer vision and machine learning. Here are a few:

(1) Image Classification: Persistent homology can capture complex structures in the data and could be used as an additional feature for image classification tasks. If similar images have similar persistent homology, these features could potentially improve the classification accuracy.
(2) Content-Based Image Retrieval (CBIR): If images with similar content have similar persistent homology, this could be used in a CBIR system to find images that are similar to a given query image.
(3) Model Interpretability: Persistent homology can provide insights into the internal workings of a transformer model. By comparing the persistent homology of different layers or heads, we can gain a deeper understanding of what each part of the model is doing, which can help us interpret and improve the model.
(4) Model Comparison: If different models process the same input differently, these differences will be reflected in the persistent homology. This can be used to compare different models, helping us understand their strengths and weaknesses and guiding the selection or design of models for specific tasks.
(5) Anomaly Detection: If normal images have one kind of persistent homology and anomalies have a different kind, this can be used to detect anomalies in image datasets.

[10] We compute the Wasserstein distance between the persistence diagrams associated to two images in this notebook. This can be used for anomalous image detection, image clustering and classification, similar to the notebooks for anomalous text detection and document clustering.

(6) Transfer Learning: The persistent homology of the context vectors, attention matrix, or hidden states could potentially be used as features in a transfer learning setting. The transformer model could be pre-trained on a large dataset, and then the persistent homology of its outputs could be used as input to another model trained on a smaller, task-specific dataset.
(7) Data Augmentation: By examining the persistent homology of transformed images (e.g., rotations, translations), we might be able to learn about the invariances captured by the model and guide the creation of new, synthetic training data.
(8) Generative Models: The similarities in persistent homology could potentially be used to guide the generation of new, synthetic images that share certain characteristics with a given set of images.

10.2. **Topic Modelling.** In [SDM, THJ, TM] clustering of contextualized word embeddings is discussed for the application of topic modeling. Considering we are able to extract collocations and multiword expressions from individual attention heads using a persistent homology informed DBSCAN, this seems like a very likely application of our methods as well. We advocate the use of persistent homology of both context vectors, and attention probability distributions of individual attention heads as a way of analyzing the topics most captured by individual attention heads, and as a way to perform topic modeling of text input. Note, we could also substitute persistent homology for the hierarchical clustering method HDBSCAN in BERTopic for a new take on hierarchical topic modelling[BERTopic, G].

Topic modeling is a type of statistical modeling for discovering the abstract "topics" that occur in a collection of documents. The persistent homology of context vectors can potentially be used for this purpose. We hypothesize that different topics will be associated with different topological features in the space of context vectors.

Given a set of texts $T = \{t_1, t_2, ..., t_m\}$, we compute the persistent homology $H(t_i)$ for each t_i. We then perform clustering on the set of persistent homologies $\{H(t_1), H(t_2), ..., H(t_m)\}$ using a suitable distance metric $d(\cdot, \cdot)$ and a clustering algorithm such as k-means or DBSCAN. Each cluster can be interpreted as a topic, and the texts associated with the persistent homologies in a cluster are the texts associated with that topic. We also note, we should, ideally, compute a Frèchet mean diagram across some of all of the attention heads. However, due to computational constraints we were unable to do so.[11] This is closely related to the next application.

10.3. **Anomaly Detection.** In the context of anomaly detection, a common task is to identify inputs that cause the model to behave in an unusual way. Given a baseline of normal behavior encoded in a set of persistence diagrams, such as a Frèchet mean of the persistent homology of "normal documents", we can use the Wasserstein distance to quantify the deviation of the behavior on a new input from this baseline. Concretely, for each new input text, we compute its persistence diagram and calculate the Wasserstein distance to each of the baseline Frèchet mean diagrams (one Frèchet mean diagram for each baseline topic). If these distances exceed a certain threshold, we classify the new input as an anomaly, since it is not close to any of the topics' Fréchet mean diagram. This threshold can be determined based on the distribution of distances within the baseline set.

This method provides a way to detect anomalies based on the internal behavior of the model, as opposed to the input or output data alone. Moreover, the persistence diagram of an anomalous input can provide insight into the nature of the anomaly by revealing the features that are most responsible for the large Wasserstein distances.

[11] Document clustering is performed in this notebook.

To reiterate in more mathematical terms, given a text t_i, we compute its persistent homology $H(t_i)$. We then define the Frèchet mean M of the persistent homologies of a set of normal (non-anomalous) texts $T_{\text{norm}} = \{t_1, t_2, ..., t_n\}$ as:

$$M = \arg \min_{m \in H(T_{\text{norm}})} \sum_{i=1}^{n} d(m, H(t_i))^2$$

where $H(t_i)$ is the persistent homology of the text t_i and $d(\cdot, \cdot)$ is a suitable distance metric on the space of persistent homologies (such as the bottleneck or Wasserstein distance). The Fréchet means represent the "average" shape of the persistent homologies of the normal texts, and we should expect one for each topic.

Given a new text t, we compute its persistent homology $H(t)$. If $d(H(t), M)$ is greater than a threshold θ, for each Fréchet mean M, we classify t as an anomaly. This is based on the assumption that anomalies will have a different topological structure, and therefore a significantly different persistent homology, compared to normal texts.[12]

We have performed anomaly detection with *xlm-roberta-base* and *xlm-roberta-large* at three levels of abstraction. We first performed the following analysis:

(1) Compute the context vectors of some input text for s pecific 'head' in a specific 'layer' of a language model.
(2) Compute the pairwise Euclidean distances between them.
(3) Use the distance matrix to compute persistent homology and a persistence diagram for the text.
(4) Do this for multiple baseline texts on the same topic.
(5) Compute the Fréchet mean of the baseline texts' persistence diagrams.
(6) Find outliers.
(7) Removing those outliers with large Wasserstein distance from the Fréchet mean may improve results but this is not done in the notebooks.
(8) Compute persistent homology and persistence diagrams for new potentially anomalous texts.
(9) Compute the Wasserstein distances between the potentially anomalous texts persistence diagrams and the Fréchet mean of the baseline texts persistence diagrams.
(10) Find outlier Wasserstein distances and classify the corresponding texts as anomalous.

This was done on for different heads in different layers, and results varied but were promising overall. We suggest running the above procedure on each head, and looking at the average behavior over all head in the model. However, due to computational constraints we have not yet done this.

We also ran the above analysis for hidden state vectors of the models, as apposed to the context vectors computed by individual heads. This also yields promising results in the small experiments we were able to run. These are the vectors $\mathbf{o}_i^l = \frac{\mathbf{d}_i^l - \mu^l}{\sigma^l}$ described at the end of §2.

10.4. **Measuring Semantic Similarity.** Determining the semantic similarity of two text corpora involves quantifying the extent to which the meanings of the texts in the two corpora are similar. Semantic similarity is a concept based on the study of semantics, or meaning in language. In NLP, semantic similarity often refers to the similarity of concepts that are expressed in text. For instance, the sentences "The dog chases the cat" and "The cat is chased by the dog" are semantically similar because they express the same basic concept, even though their wording is different.

There are many different techniques that can be used to determine the semantic similarity of two text corpora, including both rule-based and machine learning methods. For example:

[12] Anomalous text detection is performed in the notebooks found in this Github repo.

(1) TF-IDF vectors: This method involves computing the term frequency-inverse document frequency (TF-IDF) for each word in the corpora, which reflects how important a word is to a document in a collection or corpus. These vectors can then be compared using a measure like cosine similarity to quantify the similarity of the two corpora.
(2) Word embeddings: In this method, words are transformed into high-dimensional vectors that capture their meanings based on their usage in the corpus. The semantic similarity of the corpora can then be computed by comparing the word embeddings.
(3) Topic modeling: Techniques like Latent Dirichlet Allocation (LDA) can be used to identify the main topics that are present in each corpus. The similarity of the corpora can then be computed based on the similarity of their topic distributions.
(4) Semantic role labeling: This method involves analyzing the sentences in the corpora to identify the roles that different words play in the sentences' meanings. The semantic similarity of the corpora can then be computed based on the similarity of their semantic role patterns.
(5) Deep learning techniques: More advanced methods involve using deep learning models, like transformers, to create representations of the sentences or documents in the corpora. These representations can then be compared to determine the semantic similarity of the corpora.

Similar to our last application of anomaly detection, we can take a set of baseline texts, compute the Frèchet mean of the collection, and compare this to a document that may or may not have high semantic similarity to the baseline texts. We can determined their similarity using the Wasserstein distance between the persistence diagram of the Frèchet mean diagram, and the new text we are testing for similarity. We could also simply perform a pairwise comparison.

As another approach, we might use the simplex tree as a method of soft clustering to inform a hard clustering approach like DBSCAN using the filtration values as a guide. We can then look at the clusters formed by DBSCAN on the context vectors or the token probability distributions, similar to the methods in [SDM] or [TM]. We might then choose centroids as representatives of the clusters and as potential topics. We might also use a hierarchical method like HDBSCAN, again informed by persistent homology, and follow methods similar to BERTopic, where TF-IDF is used as a final step in the pipeline.

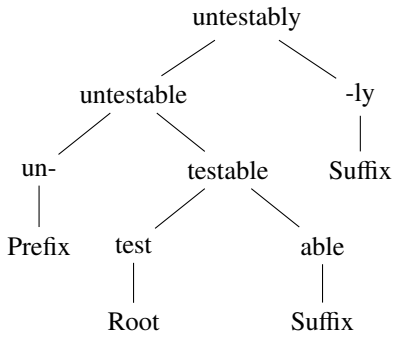
10.5. **Knowledge Distillation.** Knowledge distillation is a technique for transferring knowledge from a large, complex model (the teacher) to a smaller, simpler model (the student) with the aim of achieving comparable performance with less computational resources. The key idea is to train the student model to mimic the teacher model's behavior, as opposed to simply replicating its outputs.

The topological data analysis technique can be used to align the attention mechanisms of the teacher and student models. For each input text, we compute the persistence diagrams of the teacher and student models, and use the Wasserstein distance as a loss function that the student model seeks to minimize. This approach ensures that the student model not only produces similar outputs to the teacher model, but also reasons about the input in a similar way. Moreover, the use of persistent homology allows the student model to replicate the teacher model's attention behavior at multiple scales, which may be beneficial for handling complex inputs.

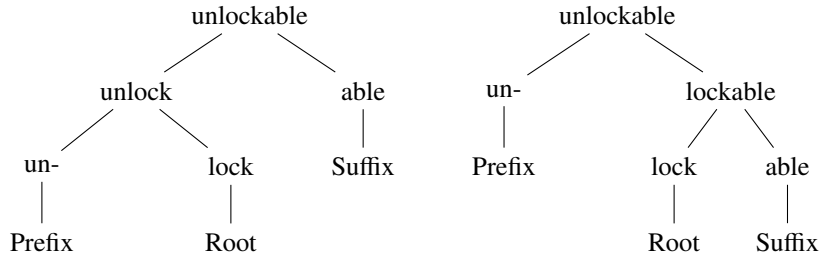
10.6. **Hierarchical Morphological Segmentation.** Hierarchical morphological segmentation is a task that involves identifying morphemes, the smallest meaningful units of a language, and the relationships among them in a word encoded as a tree structure. The proposed methodology can potentially be applied to this task as follows. If we treat morphemes as hierarchical structures, as mentioned in [CKS], the way in which morphemes cluster in a transformer should be informed by this hierarchical structure, in particular, we would expect the simplex tree resulting from applying persistent homology to an individual attention head to mimic the tree structure of a morphological

segmentation of an input word in some way. We could include this into the objective of the character level transformer as a topological prior to improve morphological segmentation.

Hierarchical morphological segmentation aims to decompose words into morphemes, the smallest meaningful units of language. The topological structure formed by the simplex tree might be applied to this task by providing a hierarchical understanding of how tokens relate to each other. Consider the morpheme connectivity tree T where each node represents a morpheme and each edge e_{ij} exists if morphemes i and j occur together in a word, joined in a particular hierarchical order, as is illustrated by the following morpheme tree.



Note that this should be a context sensitive process, and that some words have multiple valid morpheme trees, as can be seen in the next example.



The context vectors for these morphemes are used to generate the simplex tree using persistent homology. At a certain distance threshold ϵ, morphemes that should join together in a hierarchical way will form a simplicial complex, indicating a potential morphological structure similar to the morpheme tree above. As ϵ increases, we observe more and more connections between morphemes, forming a hierarchical structure of morphological units. This provides a data-driven, unsupervised approach to discover morphological structures in a language. However, it is unclear to what extent the simplex tree might be applied to morphological segmentation.

10.7. **Knowledge Graph Alternative.** Similar to the above example of a character level transformer for morphological segmentation, we now mention a "word" level application. A knowledge graph is a graph-based data structure used to represent knowledge in a structured form. Entities are represented by nodes, and their relationships are represented by edges. The context vectors of these entities can be used to generate a simplex tree using persistent homology, providing a topological structure to the knowledge graph.

Each node in the simplex tree represents an entity, and each edge represents a potential relationship between entities determined by the transformer. The distance threshold ϵ can be interpreted as the degree of confidence in the existence of a relationship. As ϵ increases, more relationships are formed, potentially revealing more complex relationships between entities.

This could also be thought of as a semantic parsing of the text into something akin to a constituency parse tree or a dependency parse tree. It would be interesting to see if the simplex tree changes in a way analogous to the parse tree as tokens are added. To be more specific, if a token is added to a non-terminal vertex of the constituency parse tree, will the hierarchy of the simplex tree obtained from the text mimic this change? What about at the level of persistent homology of attention probability distributions using the Jensen-Shannon distance or at the level of hidden states output by a layer?

11. Concluding Remarks

We must ask the questions, *"Why should a model preserve the persistent homology of some keyphrase, collocation, or multiword expression at the level of context vectors, or at the level or probability distributions associated to tokens by attention matrices?"* and *"What about a model implies that it preserves such topological features of text better than another and why should there be any clear trend in which models perform better at this?"* This is naturally an ongoing investigation, but the evidence suggests larger scale study is needed, and a better understanding of how and why a model might preserve the persistent homology of either context vectors or probability distributions associated to tokens by the attention mechanism seems an important topic of study.

In our preliminary studies, it appears that parameter count, as well as higher vocabulary size seems to be correlated with better preservation of persistent homology. However, this intuition is reversed for autoregressive models when working at the level of persistent homology of token probability distributions using the Jensen Shannon distance. We were unable to study the autoregressive GPT-2 models at the level of context vectors as they do not allow exposing the context vectors of individual heads of the model in the same way that most BERT models do.

In terms of context vectors, we note that on the one hand, higher dimensional contextual embeddings, that is, higher dimensional context vectors, would seem to imply that there is more freedom in "where" the context vectors for the tokens of the keyphrase can be placed, *while still preserving their pairwise distances (relative to each other) and thus their persistent homology*. On the other hand, higher vocabulary leads to less word splitting, which implies that the model will have fewer tokens to keep track of when learning their various contextual embeddings in various contexts. This could also explain why the probability distributions given by the softmax of the attention matrix of individual attention heads would have their topology preserved. In particular, higher dimensional weight matrices for the queries and keys $W_Q^{l,h}$ and $W_K^{l,h}$, mean that the computations of the attention probabilities

$$P(x_i) = \left(\text{softmax}_j \left(\frac{\langle q_i^{l,h}, k_j^{l,h} \rangle}{\sqrt{d}} \right) \right)_{j=1}^{n}$$

have more degrees of freedom in "where" $W_Q^{l,h} x_i = q_i^{l,h}$ and $W_K^{l,h} x_j = k_j^{l,h}$ can be placed, while still preserving the pairwise Jensen-Shannon distances between the $P(x_i)$ associated to the tokens of the keyphrase in question. Similarly, a larger vocabulary, means fewer tokens and thus fewer probability distributions to keep track of, and thus a simpler topology and simpler persistent homology to preserve. While at this point it seems clear there is a correlation between models with higher dimensional weight matrices and higher vocabulary sizes, and the preservation of the persistent homology of keyphrases, we do see this intuition broken with GPT-2 outperforming GPT-2-large and GPT-2-xl, which is likely due to the autoregressive nature of the GPT-2 models.

It is still not clear what role preserving persistent homology might play in performance. We still need to answer the question of whether including preservation of persistent homology of keyphrases

of interest in the model's objective is a promising way of improving performance on various downstream tasks. One applications that is topological in nature, protein folding, comes to mind as a case where a topological prior may be helpful. We also need to answer the question of *when* this should be included in the model's objective. Should this be included in pretraining, or is fine tuning enough? What about using Low Rank Adaptation (LoRA) of a pretrained model? Moreover, we also need to answer the question of *where* in particular this might be included. If it is observed that some attention heads might do this better than others for a certain subset of the vocabulary, should we add a LoRA to that attention head in particular, to other heads, or should we encourage the entire model to preserve persistent homology?

References

[B] Matthew Berger, *Visually Analyzing Contextualized Embeddings*, https://arxiv.org/abs/2009.02554

[BERTopic] Grootendorst, Maarten, *BERTopic: Neural topic modeling with a class-based TF-IDF procedure*, https://maartengr.github.io/BERTopic/index.html

[CKS] Ryan Cotterell, Arun Kumar, Hinrich Schütze, *Morphological Segmentation Inside-Out*, https://arxiv.org/abs/1911.04916v2

[DLYF] Yu Dai, Yuqiao Liu, Lei Yang, Yufan Fu, *An Idiom Reading Comprehension Model Based on Multi-Granularity Reasoning and Paraphrase Expansion*, Appl. Sci. 2023, 13, 5777. https://doi.org/10.3390/app13095777

[DL] Vincent Divol, Théo Lacombe*Understanding the Topology and the Geometry of the Space of Persistence Diagrams via Optimal Partial Transport*, https://arxiv.org/abs/1901.03048

[G] Maarten Grootendorst*BERTopic: Neural topic modeling with a class-based TF-IDF procedure*, https://arxiv.org/abs/2203.05794

[Hu et. al.] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, *LoRA: Low-Rank Adaptation of Large Language Models*, https://arxiv.org/abs/2106.09685

[LCO] Théo Lacombe, Marco Cuturi, Steve Oudot, *Large Scale computation of Means and Clusters for Persistence Diagrams using Optimal Transport*, https://arxiv.org/abs/1805.08331

[MMH] *Probability measures on the space of persistence diagrams*, Yuriy Mileyko, Sayan Mukherjee, John Harer, https://math.hawaii.edu/ yury/papers/probpers.pdf

[SDM] Suzanna Sia, Ayush Dalmia, Sabrina J. Mielke, *Tired of Topic Models? Clusters of Pretrained Word Embeddings Make for Fast and Good Topics too!*, https://aclanthology.org/2020.emnlp-main.135/

[THJ] Mozhgan Talebpour, Alba Garcia Seco de Herrera, Shoaib Jameel, *Topics in Contextualised Attention Embeddings*, https://arxiv.org/abs/2301.04339

[TM] Laure Thompson, David Mimno, *Topic Modeling with Contextualized Word Representation Clusters*, https://arxiv.org/abs/2010.12626

[TMMH] Katharine Turner, Yuriy Mileyko, Sayan Mukherjee, John Harer, *Fréchet Means for Distributions of Persistence diagrams*, https://arxiv.org/abs/1206.2790

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