TopoBERT: Exploring the Topology of Fine-Tuned Word Representations

Journal Title
XX(X):1–23

DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Archit Rathore[1,2], Yichu Zhou[1], Vivek Srikumar[1,3], Bei Wang[1,2]

Abstract

Transformer-based language models such as BERT and its variants have found widespread use in natural language processing (NLP). A common way of using these models is to fine-tune them to improve their performance on a specific task. However, it is currently unclear how the fine-tuning process affects the underlying structure of the word embeddings from these models. We present TopoBERT, a visual analytics system for interactively exploring the fine-tuning process of various transformer-based models — across multiple fine-tuning batch updates, subsequent layers of the model, and different NLP tasks — from a topological perspective. The system uses the mapper algorithm from topological data analysis (TDA) to generate a graph that approximates the shape of a model's embedding space for an input dataset. TopoBERT enables its users (e.g., experts in NLP and linguistics) to (1) interactively explore the fine-tuning process across different model-task pairs, (2) visualize the shape of embedding spaces at multiple scales and layers, and (3) connect linguistic and contextual information about the input dataset with the topology of the embedding space. Using TopoBERT, we provide various use cases to exemplify its applications in exploring fine-tuned word embeddings. We further demonstrate the utility of TopoBERT, which enables users to generate insights about the fine-tuning process and provides support for empirical validation of these insights.

Keywords

Visualization, Topological Data Analysis, Transformers, Language Models, BERT, Word Representations

1 Introduction

Recent advances in deep learning have improved the state-of-the-art across various natural language processing (NLP) tasks. In particular, contextualized word embeddings such as BERT[1] and RoBERTa[2] have revolutionized NLP by providing general-purpose learned embeddings. A common theme across these models is that word embeddings are computed using several transformer layers[3], where the neuron activations (i.e., outputs of neurons) at a particular layer (typically the last one) are treated as the vector representations of the words. For simplicity, we use "embeddings" to refer to these high-dimensional vector representations produced by a particular layer of a model-task pair, and "embedding space" to refer to the space of these embeddings.

The attention mechanism of transformers[3] is widely considered to be the main reason behind their impressive performance, but how the embeddings generated by these models encode various types of linguistic information remains mostly unknown. The large size of these transformer-based models prohibits direct analysis of the model architecture, and computational limitations prohibit combinatorial methods such as feature ablation studies. Hence, there is a need for methods that can probe the embeddings produced by these models to understand the reasons behind the effectiveness of these models.

There are two ways to use a pretrained transformer-based model: as a feature extractor where the parameters of trained model are frozen, or by fine-tuning the parameters of a pretrained model for a downstream task. It has been observed that fine-tuning generally improves task-specific performance compared to the pretrained model[4]. However, the effect of fine-tuning on the embedding space is relatively less understood. Previous works[5–9] have studied embeddings from fine-tuned models through classifier-based probes and geometric analysis of the embedding space[10]. These methods, however, do not capture details of how the fine-tuned word embeddings are organized at intermediate and final layers.

Contributions. In this paper, we present TopoBERT, a visual analytics system to explore the topological structure of word embeddings during the fine-tuning process of a transformer-based model. It combines tools from topological data analysis (TDA) and visualization to enable interactive exploration of the embedding spaces obtained from models that are fine-tuned on a set of NLP tasks. In particular, TopoBERT leverages the *mapper graph* from Singh et al.[11] to summarize the topological structure of the embeddings. Each node of the mapper graph represents a cluster of embeddings, and two nodes are connected by an edge if their corresponding clusters have a nonempty intersection. Built upon the mapper graph, TopoBERT provides visualization and analysis capabilities for generating insights into the organization and evolution of embeddings, and validating them using subsequent experiments. Its targeted users are experts in NLP and linguistics. In summary:

[1] School of Computing, University of Utah. [2] Scientific Computing and Imaging Institute, University of Utah. [3] Allen Institute for AI
Email: beiwang@sci.utah.edu

1. We introduce TopoBERT, a visual tool to explore word embeddings during the fine-tuning process for transformer-based models using topological techniques. To the best of our knowledge, this is the first tool of its kind.
2. For a model-task pair, the tool provides an accessible, no-code pipeline to perform visual analytics on embeddings across multiple layers. TopoBERT can be easily extended to explore any embeddings across layers and tasks beyond fine-tuning (e.g., during the training process).
3. TopoBERT comes equipped with a number of unique features to explore the embedding space. It considers a cluster of embeddings associated with a node in the mapper graph as a *topological neighborhood*. It introduces the notion of *purity* for such a neighborhood using entropy to capture the mixing behavior of labels in the embedding space. It provides a method to study how embeddings of the validation examples are positioned with respect to the embedding space of training examples using *node attachment*.
4. We present various use cases demonstrating how TopoBERT can be used to explore embeddings from transformer-based models.
5. We demonstrate the utility of TopoBERT for generating and validating insights into the local and global structures of the embeddings and how the fine-tuning process affects the embedding spaces.

We have also released an open-source implementation of our tool on GitHub (https://github.com/tdavislab/TopoBERT).

Overview. We review related work in Sect. 2. We give a brief introduction to the topological tool — the mapper graph — in Sect. 3. We then describe the current configuration of the NLP components within TopoBERT regarding datasets, embeddings, and models in Sect. 4. We introduce the design requirements of TopoBERT in Sect. 5 and outline the user interface and architecture of TopoBERT in Sect. 6. We present various use cases in Sect. 7. We illustrate the utility of TopoBERT in generating and validating insights regarding embeddings in Sect. 8. We present post-deployment expert evaluation in Sect. 9. Finally, we conclude with future directions in Sect. 10.

2 Related Work

Interpretability has become increasingly important in understanding how ML models give predictions. We review a number of recent works that analyze the internal representations of these models from TDA, ML, data visualization, and NLP communities. However, there are no existing tools that probe the embedding space using topology and visualization, in particular, to understand the fine-tuning process of transformer-type models.

2.1 TDA for ML and NLP

Tools from TDA have been integrated with ML and NLP in recent years. Hofer et al.[12] proposed a method to convert topological signatures into vector features usable for deep learning. Rathore et al.[13] proposed a visual analytics system using the mapper algorithm from TDA to visualize the embedding space from image classifiers and (to a lesser extend) pre-trained BERT models. Gabrielsson et al.[14] demonstrated the existence of certain layer-specific topological structures in convolutional neural network. Clough et al.[15], Hu et al.[16], and Chen et al.[17] proposed various topological loss functions for image segmentation tasks. Topology has also been used to propose criteria for a neural network's generalization properties[18–21]. TDA has been used in NLP for movie genre detection[22], textual entailment[23], document summarization[24], and analysis of sentence embeddings[25]. The topology of the attention layers has been leveraged for text classification[26,27], acceptability judgements[28], and robustness against adversarial attacks[29].

2.2 Visualization for ML Interpretability

Various visual analytics systems have been proposed for interpreting ML models[30–38]. Studies from Chen et al.[39], Yang et al.[40], Krause et al.[41], and May et al.[42] focused on understanding the distribution of the input data and feature selection. Other methods have visualized the intermediate representations from the hidden layers[43–47].

Systems such as ModelTracker[48], Squares[49], and Manifold[50] enable interactive visualization for debugging ML models, performing error examination, and understanding instance-level performance. HypoML[51] performs hypothesis-based evaluation of an ML model using visual analytics. iNNvestigate-GUI[52] provides a toolbox of feature visualization techniques for input visualization and model output explanation. See[53–55] for comprehensive surveys of visual analytics for machine learning.

2.3 Visualization for NLP

We also review work on visual analytics for interpreting deep NLP models; see[56] for a comprehensive survey of various visualization works for deep learning. Liu et al.[57] proposed new techniques for visualizing high-dimensional word embeddings beyond dimensionality reduction with a focus on capturing syntactic and semantic analogies. The explAIner system[58] is a visual analytics framework for understanding ML models by applying concepts from explainable AI (XAI) research such as LIME[30] and ANCHORS[59]. Liu et al.[60] proposed NLIZE, a visual analytics system that enables perturbation-driven exploration for inputs, intermediate embeddings, and outputs. The Melody system by Chan et al.[61] constructs a global overview of model and data behavior from local explanations by using information theory. Tools such as RNNVis[62] and LSTMVis[63] use correlation analysis to cluster hidden-state neuron activations for various tasks. Berger[64] proposed a system for analyzing contextualized embeddings from transformers and a related family of models by using pairwise co-occurence information of words and spans. Xiaonan et al.[65] proposed a system for exploring neural embeddings of documents and identifying salient features for task-specific applications. The BertViz system by Vig et al.[66] visualizes self-attention in transformer-based models to increase interpretability. In general, word embeddings from NLP models are high-dimensional vectors, so generic

high-dimensional visualization techniques such as PCA[67], t-SNE[68], and UMAP[69] are applicable (see Liu et al.[70] for a survey on visualizing high-dimensional data).

2.4 Probing Embeddings in NLP

Transformer-based models are widely used in contemporary NLP[71] applications, and various studies have focused on probing the contextualized word embeddings they construct. The most commonly used methodology involves training a classifier to predict linguistic properties[5–9] based on the embeddings. Different properties such as complexity[72] and minimum description length[73] of the learned classifiers have been used to evaluate the embeddings. In addition to classifier-based probing, various studies have analyzed the internal structure of embeddings and provided insights about the geometry of the embedding space[74–76]. For example, Hewitt and Manning[77] showed that syntactic dependency relationships can be recovered from the BERT embeddings by a simple linear transformation, and Ethayarajh[78] showed that the vectors in the embeddings occupy a narrow cone in the embedding space. Fine-tuning a model for a specific task is a common practice, but there are limited insights[10,79–82] into the process of fine-tuning. Specifically, few studies have attempted to understand how fine-tuning affects the model parameters and internal embeddings.

In this paper, we investigate contextualized word embeddings from a topological perspective, in contrast to a geometric perspective employed in previous works[74–76]. We discover new insights about the organization and evolution of embeddings using TopoBERT. As far as we are aware, this is one of the first works (besides[13]) to analyze the topological structure of word embeddings, and use it to examine how they encode linguistic information.

3 Topology Background

In this section, we review the technical background on mapper graphs, a widely used tool from TDA. We also describe quantitative measures associated with the mapper graph that are used in the experiments of Sect. 8.

3.1 Mapper Graph on Point Cloud Data

Given a high-dimensional point cloud $\mathbb{X} \subset \mathbb{R}^d$ equipped with a continuous function $f : \mathbb{X} \to \mathbb{R}$, the mapper graph[11] provides a topological summary of the data. It is, in a nutshell, a clustering of points in $\mathbb{X}$ induced by the function f. There are two concepts essential to the understanding of a mapper graph, namely, a cover and its nerve. We illustrate the construction of a mapper graph from a 2-dimensional point cloud $\mathbb{X}$ sampled from a toy dataset of "circle with three hairs". As shown in Fig. 1(a), $\mathbb{X}$ is equipped with a height function f, where red colored points are lower and blue colored points are higher along the height function.

An *open cover* of $\mathbb{X}$ is a collection $\mathcal{U} = \{U_i\}$ of open sets such that $\mathbb{X} \subset \bigcup_i U_i$. Given an open cover $\mathcal{U}$ of $\mathbb{X}$, the 1-dimensional *nerve*[83] of $\mathcal{U}$, denoted as $\mathcal{N}_1(\mathcal{U})$, is constructed as a graph: each cover element U_i is represented as a node i, and there is an edge between node i and node j if U_i and U_j have nonempty intersection. Intuitively, as illustrated in Fig. 1(a), imagine covering a set of points $\mathbb{X}$ with partially overlapping postage stamps (e.g., rectangles) such that no point in $\mathbb{X}$ is visible. To construct the nerve, each stamp is a cover element abstracted as a node, and there is an edge between nodes if their intersection contains points in $\mathbb{X}$; this is shown in Fig. 1(b). For example, cover elements U_1 and U_4 have a nonempty intersection in Fig. 1(a), hence there is an edge between node 1 and node 4 in the nerve.

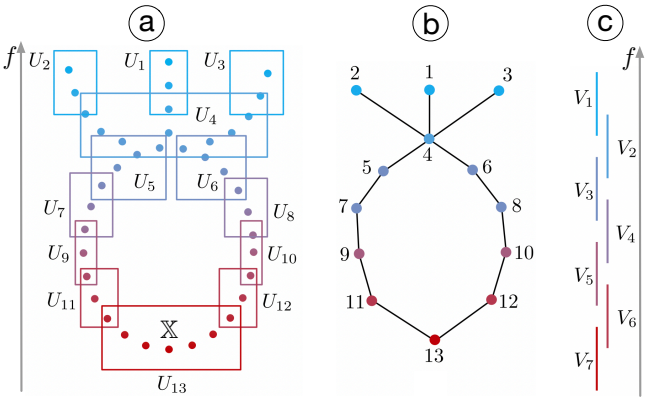


Figure 1. A simple example of computing a mapper graph. A point cloud $\mathbb{X}$ in (a) is colored by the height (e.g., y-coordinate) function $f : \mathbb{X} \to \mathbb{R}$. The cover $\mathcal{U} = \{U_1, \ldots, U_{13}\}$ of $\mathbb{X}$ is induced by a cover $\mathcal{V} = \{V_1, \ldots, V_7\}$ of $f(\mathbb{X})$ that contains 7 intervals with $1/3$ overlap in (c). The 1-dimensional nerve of $\mathcal{U}$ is the mapper graph in (b).

The next question is how to construct a reasonable cover of $\mathbb{X}$ using information provided by the function f. The cover shown in Fig. 1(a) is, in fact, constructed as follows. We start with a finite cover $\mathcal{V} = \{V_j\}$ of the image $f(\mathbb{X})$, such that $f(\mathbb{X}) \subseteq \bigcup_j V_j$. Since f is a scalar function, V_j is an open interval in $\mathbb{R}$. Fig. 1(c) illustrates the set $\mathcal{V} = \{V_1, \ldots, V_7\}$ covering $f(\mathbb{X})$. We obtain the cover $\mathcal{U}$ of $\mathbb{X}$ by considering the clusters induced by points in $f^{-1}(V_j)$ for each j, as shown in Fig. 1(a). The 1-dimensional nerve of $\mathcal{U}$, denoted as $\mathcal{M} := \mathcal{N}_1(\mathcal{U})$, is referred to as the *mapper graph* of $(\mathbb{X}, f)$, as shown in Fig. 1(b).

The function f is called the *lens* (or *filter function*), through which we look at the data. Different lenses (such as density and eccentricity) provide different insights[11,84]. In this paper, we use the L_2-norm of the embeddings as the lens function across all our analysis. Such a lens function has been shown to produce meaningful results in the analysis of activation vectors from images[13]. Finding the best lens function for a particular dataset beyond best practices[84,85] remains an open problem.

Besides f, a number of parameters are associated with a mapper graph. To define the cover $\mathcal{V}$ of $f(\mathbb{X})$, the most common strategy is to use uniformly sized overlapping intervals. Two parameters define such a cover: the number of intervals n and the amount of overlap p between adjacent intervals. These parameters may be modified by the user via the interface of TopoBERT. We choose a default configuration by setting $n = 50$ and $p = 0.5$. These parameters are currently hand-tuned; however, there are studies detailing automatic parameter tuning methods[86,87]. As we compute the clustering of the points lying within $f^{-1}(V_i)$ and connect the clusters whenever they have nonempty intersection, additional parameters are associated with the clustering algorithm. A typical algorithm to use is

density-based DBSCAN[88], which requires two parameters: *minPts* is the number of samples in a neighborhood for a point to be considered as a core point, and ϵ is the maximum distance between two samples for one to be considered in the neighborhood of the other. TopoBERT uses the elbow method suggested in [88] and utilized in [89] to estimate ϵ automatically and allows the user to specify *minPts* through its interface. We use a default value of $minPts = 3$.

3.2 Topological Neighborhood Purity

The mapper graph captures the topological structure of a point cloud $\mathbb{X}$ with respect to a chosen lens function f (e.g., L_2-norm in our setting). It is by definition a graph, where each node consists of a cluster of points in the data, and there is an edge between two nodes if their corresponding clusters have a nonempty intersection. We thus define a cluster of points associated with a node in the mapper graph a *topological neighborhood*. In other words, each node in the mapper graph is a topological neighborhood, and the edges between these nodes encode the overlaps between these neighborhoods. A topological neighborhood of $\mathbb{X}$ is induced by f and is not necessarily the same as a Euclidean neighborhood: two points x and y are in the same topological neighborhood if they are close to each other in terms of a Euclidean metric, and their function values $f(x)$ and $f(y)$ fall in the same interval (i.e., a cover element) of $f(\mathbb{X})$.

Suppose an input point cloud $\mathbb{X}$ is equipped with k class labels, $L = \{l_i, \ldots, l_k\}$, and a labeling $l : \mathbb{X} \to L$ assigns each point a label in L. Let $X \subset \mathbb{X}$ denote a topological neighborhood consisting of m points $\{x_1, \ldots, x_m\}$, which corresponds to a node in the mapper graph. Let D_X be the observed distribution of labels $\{l(x_1), \ldots, l(x_m)\}$ for points in X. Let D be a uniform distribution of labels among m points. Let H denote the Shannon entropy of a distribution. We define the *purity* $p(X)$ of a topological neighborhood X to be

$$p(X) := 1 - \frac{H(D_X)}{H(D)}.$$

$p(X)$ describes the mixing behavior of labels in X and also referred to as the *node purity* to emphasize its association with a node in the mapper graph. It reaches the highest value of 1 when all points in X are from the same class, and the lowest value of 0 when the points are uniformly distributed over all classes. Note that this notion of purity is different from those recently introduced by Purvine et al.[90]

3.3 Interpreting Mapper Graph of Word Embeddings

We now describe how a mapper graph can be interpreted in the context of TopoBERT. The mapper graph of a high-dimensional point cloud $\mathbb{X}$ is a graphical representation of its topological structure. It represents the *shape* of the data with respect to the lens function by encoding topological neighborhoods via nodes and their proximity via edges.

In the context of TopoBERT, a mapper graph is constructed by taking the data $(\mathbb{X}, f)$ as input, where $\mathbb{X}$ is a point cloud of high-dimensional word embeddings, and $f : \mathbb{X} \to \mathbb{R}$ is the L_2-norm. In particular, $\mathbb{X}$ contains activations of input tokens (i.e., words in a sentence) from a layer of a BERT-type model during a batch-update of the fine-tuning process (see Sect. 4.2 for details). The L_2-norm of a point in $\mathbb{X}$ captures the magnitude of the activation, that is, how strongly the model is "activated" by the input token. Therefore, embeddings are clustered into the same node of a mapper graph if (a) they have similar activation magnitude when passed through the model, and (b) they are close to each other in the high-dimensional space under a Euclidean metric.

The set of embeddings within a single node may have different class labels. TopoBERT encodes the distribution of class labels as a pie chart (see Fig. 3). This encoding allows users to quickly inspect the node purities (see Sect. 3.2) in different regions of the graph.

Exploring the mapper graph provides users a way to reason about the embedding space from a topological perspective. We highlight a number of use cases in Sect. 7.

3.4 Mapper Graph Node Attachment

For a given NLP task, we work with a point cloud comprised of the embeddings of training, test, or validation examples (i.e., words in their context). In TopoBERT, the mapper graph is constructed from embeddings of the training examples. However, it is often useful to understand how embeddings of the validation examples are positioned with respect to the embedding space of training examples. To that end, we propose a simple heuristic to attach the embedding of a validation example to nodes in the mapper graph (constructed from training examples).

First, we compute the mapper graph $\mathcal{M}$ of the embeddings $\mathbb{X}$ from the training examples. Next, for each embedding y of a validation example, we compute its nearest neighbor x in $\mathbb{X}$. By construction, x belongs to at most two nodes in $\mathcal{M}$. If x belongs to a single node X in $\mathcal{M}$, then we attach y to X. If x belongs to two nodes X and X' in $\mathcal{M}$, then we attach y to the node with a closer centroid (w.l.o.g., assume y is attached to X). To reduce visual clutter, we group all points y that are attached to the same node X in $\mathcal{M}$ into a single super-node and create an edge connecting the super-node with X in the visualization; see Fig. 9 for an example.

3.5 Distance Between Mapper Graphs

TopoBERT allows users to explore the mapper graphs of embedding spaces across multiple fine-tuning batch updates, subsequent layers of the model, and different NLP tasks. It is therefore useful to compare two mapper graphs to quantify their differences. We particularly focus on mapper graphs of embeddings that arise from the same set of training examples, as they go through a neural network whose weights are changing over the course of fine-tuning. In other words, given an embedding space of training examples before fine-tuning, we are interested in the evolution of its mapper graph as the underlying embedding space changes across batch updates.

By construction, a mapper graph $\mathcal{M}$ constructed from a point cloud $\mathbb{X}$ could be modeled as a hypergraph $\mathcal{H}$: each point $x \in \mathbb{X}$ is a node in $\mathcal{H}$, and each subset of points that constitutes a node in $\mathcal{M}$ is a hyperedge in $\mathcal{H}$. By representing a mapper graph as a hypergraph, we employ a hypergraph

Table 1. A summary of pretraining method, dimension of embeddings, and number of parameters for the models available within TopoBERT. MLM stands for the Masked Language Model objective [1] for pretraining transformer-based models.

Rep	Pretraining	Dim	#Param
BERT-base	MLM	768	110.1M
RoBERTa-base	MLM	768	125M
BERT-Tiny	distillation	128	4.4M

distance based on co-optimal transport [91] to compute the distance between two mapper graphs.

4 Datasets, Embeddings, and Models

In this section, we describe the configuration of the various NLP components used in TopoBERT. These are easily generalizable to other datasets, embeddings, and models.

4.1 Datasets

We conduct our analysis on three NLP tasks, covering syntactic and semantic aspects of languages. Here, we provide a brief description of these tasks.

Preposition supersense disambiguation is the task of predicting coarse semantic categories of prepositions called supersenses. There are two sets of labels—**Supersense Role** and **Supersense Function**—and correspondingly two separate classification tasks. Following previous work [92], we make predictions for single prepositions using the annotations from Streusle v4.2 corpus; we obtain the Streusle dataset from https://github.com/nert-nlp/streusle.

Dependency relation refers to the task of assigning a dependency label to a pair of tokens in a sentence. These labels describe the syntactic relation between the two tokens. To generate the embeddings for this classification task, we concatenate the embeddings of these tokens. The concatenated embedding is used as the vector representation of the token pair. We use the English portion of the parallel universal dependency (PUD) treebank [93], where the PUD treebank is downloaded from https://github.com/UniversalDependencies/UD_English-PUD.

4.2 Embeddings

In this work, we consider three representative embedding models from the BERT family: BERT-base [1], BERT-Tiny [94] and RoBERTa-base [2]. Table 1 summarizes the method used to train these models, the dimensionality of the corresponding embeddings, and the number of parameters in each of these models.

4.3 Fine-Tuning

We fine-tune the models from Sect. 4.2 on the datasets described in Sect. 4.1. Following the methods used in previous work [10], we fine-tune BERT-base and RoBERTa-base for 3 epochs and BERT-Tiny for 10 epochs. During the fine-tuning process, we save the checkpoints of these models at every fixed number of updates (5 updates for BERT-base and RoBERTa-base, 15 updates for BERT-Tiny) to track how these embeddings change. After fine-tuning, we generate the embeddings in all layers of the model using these checkpoints. All the models are fine-tuned using HuggingFace library [95] and using the AdamW [96] optimizer with a batch size of 32. A linear weight scheduler with 10% warmup steps is used. We use a learning rate of 3×10^{-4} for all the models.

5 Design Requirements

In this section, we outline the design requirements that have guided the development of TopoBERT. Our goal is to design a tool for exploring embeddings from language models that are fine-tuned for specific tasks from a topological perspective.

TopoBERT has been designed to address the requirements from (1) NLP experts involved in model understanding and analysis, and (2) linguists working on taxonomy and categorization that lead to task definitions. See Sect. 6.3 for some expert feedback during design and development, and Sect. 9 for a post-deployment expert evaluation. Using TopoBERT, we aim to help these users explore word embeddings using qualitative visual exploration followed by quantitative analysis.

R1. ***Summarizing the underlying structure of word embeddings from a topological perspective.***

The word embeddings from a transformer model are vectors in a high-dimensional space, endowed with a rich structure that reflects the model's understanding of lexical, syntactic, and semantic concepts.

Common approaches for summarizing word embeddings are centered around clustering and dimensionality reduction techniques. Clustering techniques (such as K-Means [97] and DBSCAN [98]) group similar data points in a cluster but do not explicitly preserve the relationships between clusters, that is, the intracluster information. Dimensionality reduction techniques (such as PCA [67], t-SNE [68], and UMAP [69]) transform data from a high-dimensional space into a low-dimensional (oftentimes 2- or 3-dimensional) space so that the low-dimensional representation retains certain properties of the original data. However, they introduce distortions and may not preserve local (or intercluster) information, e.g., data points far away in the high-dimensional space are projected near each other in the low-dimensional space.

In comparison to the above common approaches, the mapper graph utilized in TopoBERT provides a graph-based representation that aims to preserve the topological structure in high dimension. Locally similar points are grouped into nodes (clusters), thus preserving intercluster information, whereas intracluster relationships are encoded explicitly as edges between the nodes (clusters). In particular, the mapper graph of an embedding space captures the local structure that encodes fine-grained complexities in the language, as well as the global structure that reflects coarse-grained concepts. As demonstrated in Sect. 7, such a graph-based representation summarizes the topological structure of high-dimensional embeddings, and enables novel explorations of the embedding space.

R2. ***Supporting interactive exploration with structural summaries across model-task combinations.***

Word embeddings are associated with multifaceted metadata, have complex structures, and may arise from various data sources. To understand these embeddings, a visual analytics system should not only support interactions with their global summary structures ("overview first"[99]), but also allow drilling down into the associated metadata ("details on demand"[99]), e.g., the sentences associated with or class labels attached to certain words (tokens). The system should also enable users to focus on a subset of the embeddings, via selection, search, and highlighting ("zoom and filter"[99]). For generalizability, the system should be adaptable to different models and tasks. Finally, users should be able to change parameters of the algorithms used to obtain the summary structures.

R3. ***Enabling the generation and validation of insights for word embeddings during the fine-tuning process.***

An important aspect of interactive exploration is to enable users to generate insights into word embeddings from two perspectives. First, how does a model's representation of the data give rise to interesting structures in the embedding space? Second, how do the structures captured with a model relate to the linguistic aspects of the data? Generating insights into both the data and the model is important, since NLP experts are interested in the model's representation, whereas linguists aim to identify and design annotation schemes for various language tasks. Additionally, the system should provide a way for the users to validate these insights easily through follow-up analysis and experiments.

6 Implementation and User Interface

6.1 Architecture and Implementation

TopoBERT is built using a server-client architecture, as illustrated in Fig. 2. The web-based frontend is implemented using Vue.js and D3.js. The backend is developed using Python and Flask and consists of the computation engine, web server, and the embedding data store. Mapper graphs are precomputed using a particular set of parameters and then cached in the browser during interactive exploration. Any change to parameters that are not already cached triggers a computation on the backend. A parallelized version of the mapper algorithm from Zhou et al.[89] is used to compute the mapper graphs efficiently on the fly.

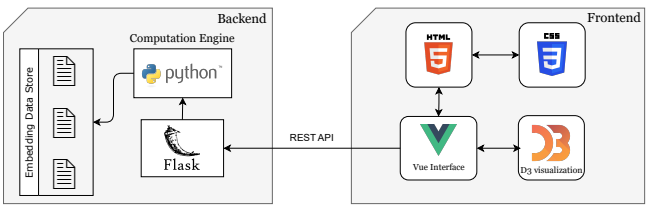


Figure 2. System architecture for TopoBERT.

6.2 User Interface

As illustrated in Fig. 3, the interface of TopoBERT consists of two primary components: the mapper graph panel (a) and the control panel (b). The mapper graph panel (a) shows a graph-based topological summary of the embeddings from a transformer-based model fine-tuned on a linguistic task. The mapper graph is visualized by a force-directed layout. It supports panning, zooming, and selection of a subset of the nodes. Each node in the mapper graph represents a topological neighborhood in the high-dimensional embedding space. It contains a number of input data points (e.g., embeddings of training examples), and is visualized by a pie chart that denotes the distribution of class labels among its data points, which allows for a quick inspection of the neighborhood class composition and node purities. The panel also contains a class composition view (c) that displays the distribution of class labels in the selected nodes.

The control panel (b) supports the inspection of metadata and the selection of parameters for the mapper algorithm. To address the design requirement R2, it allows users to select from the model-task pair in Sect. 4.1 via the *data source parameters*. The control panel also provides functionalities to display metadata (e.g., sentences that contain certain tokens/words) and lens function distribution associated with the selected nodes via the *selected nodes panel*. In addition to browser-cached mapper graphs, it allows users to tweak the parameters for computing the mapper graphs on the fly via the *mapper parameters* panel. The panel further provides capabilities for searching and highlighting nodes in the graph by class labels or specific words via the *search and highlight* panel. Finally, it shows the PCA projection of embeddings for comparative purposes via the *dataset PCA projection*.

6.3 Expert Feedback During Design and Development

During the design and development of TopoBERT, experts in NLP and computational linguistics have been part of the collaborative effort. Two NLP experts (both coauthors) have been involved in the entirety of the collaboration. In particular, their inputs have helped to draft and refine the design requirements (Sect. 5) as well as the initial user interface (Sect. 6).

We also conducted a 60-minute demo session to collect feedback from two independent experts in computational linguistics. We include a number of key comments from the session below (denoted by **C**s).

C1: A better tutorial is needed for introducing the mapper graphs to domain experts who are unfamiliar with topology.

To address **C1**, we are creating a tutorial on mapper graphs for experts in NLP and linguistics who wish to employ topological analysis driven by TopoBERT, but might not be familiar with topology.

C2: When exploring the mapper graphs, TopoBERT should enable users to dive deeper into the metadata associated with the embeddings. In particular:

C2a. When selecting a node in the mapper graph, a user should be able to observe the class label associated with each embedding and differentiate among different labels.

C2b. A distribution of lens (filter function) values is needed to gauge the variation among embeddings.

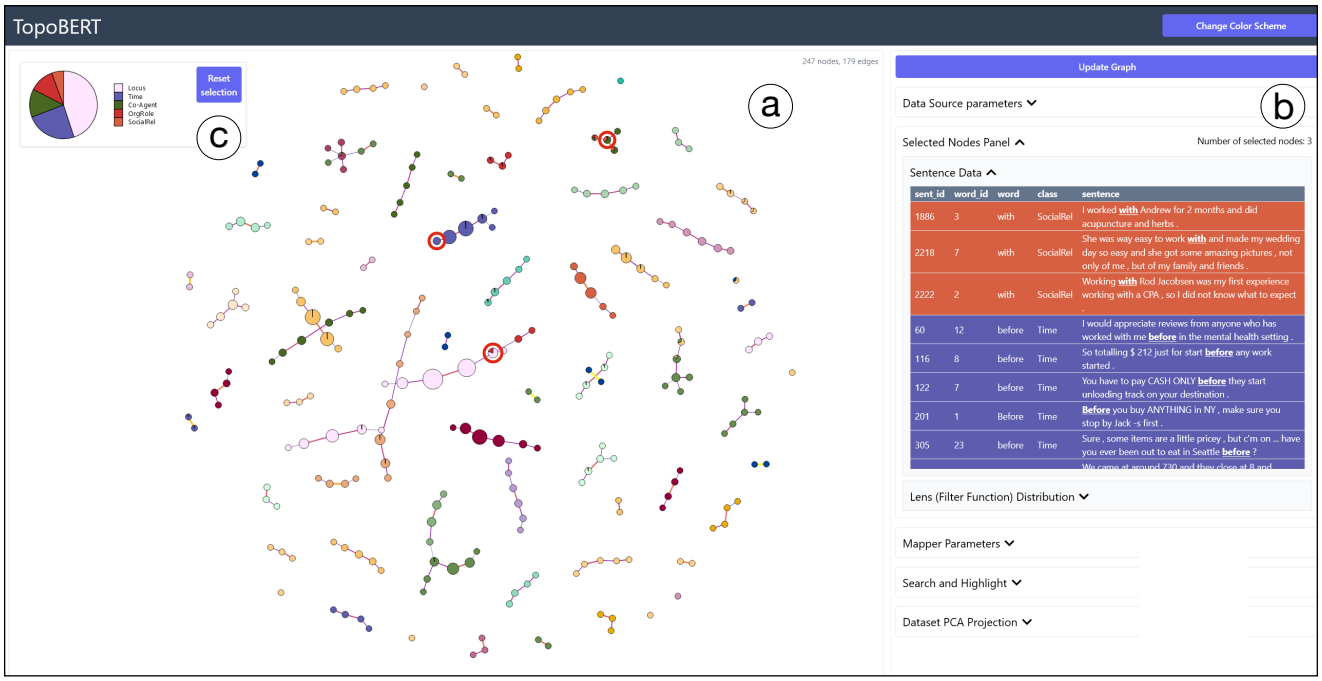


Figure 3. The visual interface of TopoBERT. The mapper graph panel (a) provides a graph-based topological summary of the embeddings. The control panel (b) supports the inspection of metadata and the selection of parameters for the mapper algorithm. The class composition view (c) displays the distribution of class labels in the selected nodes of a mapper graph. Each node in the mapper graph represents a topological neighborhood of embeddings. It is visualized by a pie chart that encodes its class composition. Edges between nodes encode overlaps between two nearby topological neighborhoods.

C2c. An enlarged pie chart would be useful to dive deeper into the composition of topological neighborhoods.

Based on **C2**, we added or enhanced several features of the visual interface shown in Fig. 3, described next.

Enhancement to the metadata view. We enhanced the metadata view, whose features are visible in Fig. 3(b) under *Sentence Data*. In addition to showing the plain *sentence* information in the initial prototype, we provide high-level labels from the metadata table, highlight target tokens in the context of sentences, and enrich the color encoding of class labels for easier differentiation.

Enhancement to the control panel. We added visualization that highlights the distribution of lens function (*Lens Distribution*). The *Search and Highlight* feature were also improved based on the feedback.

Addition of the composition view. We added the composition view to display the distribution of class labels in the selected nodes of a mapper graph, which helps experts better investigate topological neighborhoods locally.

7 Use Cases for Linguistic Phenomena

We now present various use cases for exploring contextualized word embeddings using TopoBERT, considering experts in NLP and linguistics as users.

7.1 Global Structures of Embeddings

Understanding the global structures of embeddings at the intermediate and final layers and their evolution during fine-tuning is a key step to understanding models and improving them. We present two use cases for how TopoBERT can be used to better understand these structures and suggest actions for model improvement. TopoBERT is generalizable to study the organization and evolution of embeddings at intermediate layers during training or transfer learning.

Structural differences for the same task across different models. First, a user may employ TopoBERT to explore structural differences among embeddings generated for the same task across different models. We focus on embeddings at the final layer (i.e., layer 12) from BERT-base, RoBERTa-base, and BERT-Tiny, all of which are fine-tuned on the Supersense-Role task.

We visualize the mapper graphs constructed from embeddings of training examples in Fig. 4. The mapper graphs for BERT-base and RoBERTa-base are similar, and both of which contain isolated chains of nodes with high purity. Such a similarity indicates that these two models map different classes to different regions in the embedding space, which implies good predictive performance of the model. In contrast, the mapper graph for BERT-Tiny contains several large chains of impure nodes in the center. These impure chains indicate that BERT-Tiny may perform worse on inputs from the classes in these impure chains. Whereas the poor predictive performance of BERT-Tiny is expected because it has far fewer parameters (see Table 1), TopoBERT goes beyond just a single number to estimate performance, and gives us both a mechanistic explanation for the observed accuracy, and also an identification of specific labels and examples that are confused by the model.

These observations provide natural actionable items for improving model performance, namely: (1) increasing model capacity by introducing more layers, (2) fine-tuning for a larger number of batch updates, (3) fine-tuning by emphasizing classes found in the impure chains, and (4) adding new loss terms that incorporate this topological

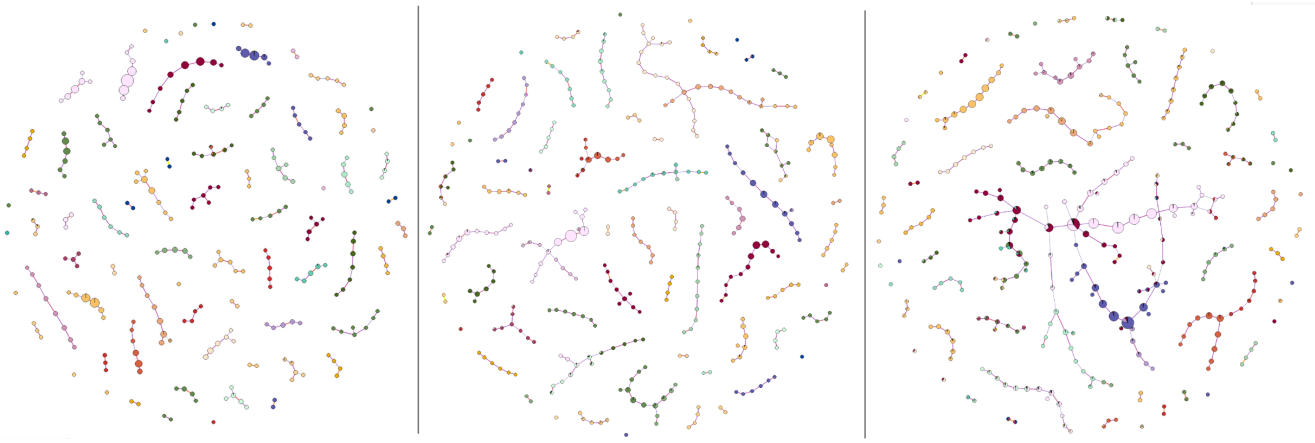


Figure 4. Topological structures of embeddings at the final layers of BERT-base (left), RoBERTa-base (middle), and BERT-Tiny (right), fine-tuned on the Supersense-Role task.

information for training BERT-Tiny to encourage the class separation we observe from the mapper graphs of BERT-base and RoBERTa-base.

Structural differences across batch updates during fine-tuning. During fine-tuning, it is a common practice to monitor various statistics such as loss and accuracy for training or validation examples. Using TopoBERT, a user may explore structural changes across batch updates during fine-tuning. As illustrated in Fig. 5, TopoBERT provides additional qualitative measures for judging the progress of the fine-tuning process based on the evolution of mapper graphs across updates. In particular, we observe significant improvement in node purity from update 5 to update 70, and purity subsequently improves only slightly at update 175. The notion of node purity can lead to additional insights about the model. We will see in Sect. 8 that the average node purity is correlated with model performance on the unseen data.

In other words, the global structure of embeddings appears to stabilize faster (at update 70) than when the model is deemed to have converged (at update 175). We expect that by integrating TopoBERT into monitoring dashboards such as the TensorBoard [100], model designers may derive additional insights about the training and fine-tuning processes.

7.2 Local Behaviors of Embeddings

We now present a few use cases of how TopoBERT can be used to understand the local behaviors of embeddings, especially at intermediate layers.

Identifying subcategories of linguistic phenomena captured by a model. Transformer-based models encode contextual information in their word embeddings, as opposed to static embeddings such as Word2Vec [101]. Using TopoBERT, we can identify different types of linguistic phenomena captured by these contextualized embeddings.

Consider the example in Fig. 6 using the RoBERTa-base model fine-tuned on Supersense-Role. This example shows a chain formed from training examples with the same class label 'Identity'. Examining the examples in the nodes using TopoBERT, we observe variation of a linguistic phenomenon (i.e., fronted clause) captured by the embeddings. Specifically, TopoBERT shows that the mapper graph identifies a chain along which the fronted (Fig. 6a) vs nonfronted usage (Fig. 6b-c) of the word 'as' are separated. This phenomenon is not encoded in the class labels for fine-tuning, or explicitly defined in the original training examples for the base model. Whereas the embeddings of training examples with the same label are shown to be grouped together in the mapper graph, TopoBERT allows a user to hypothesize, investigate, and discover additional structure among them that align with linguistic concepts such as sub-categories that are not explicitly specified in the original task definition.

In addition to identifying specific and frequent linguistic phenomena in the data, this process can lead to better linguistic insight about the class ontology itself, allowing annotation designers to either refine or merge class labels. In this fashion, TopoBERT can be useful, not only for exploring the space of embeddings, but also for understanding the linguistic phenomena at play in the underlying text.

Discovering model confusions in embedding spaces. TopoBERT makes the structures in a high-dimensional embedding space explicit using a mapper graph, which is constructed by grouping embeddings into clusters and preserving the pairwise relations among these clusters with edges. This feature makes TopoBERT distinct from methods such as dimensionality reduction because it enables a user to dive into the local neighborhood of the embedding space and analyze its class composition, as outlined in R1. TopoBERT also makes the relationship between topological neighborhoods explicit through the edges in the mapper graph, allowing structures such as branches, loops, and chains in the embedding space to be identified, which is not possible with dimensionality reduction or clustering techniques. These features can be especially useful for discovering class labels that are frequently confused by the model.

In Fig. 7, we illustrate one such exploration scenario, where we focus on a chain of nodes with low purity. The model appears to group embeddings from three semantically unrelated classes ('Cost', 'Possession', and 'Theme'). Using TopoBERT, a user may further explore the input sentences corresponding to these embeddings and conclude that the model confusion arises due to all sentences discussing monetary concepts.

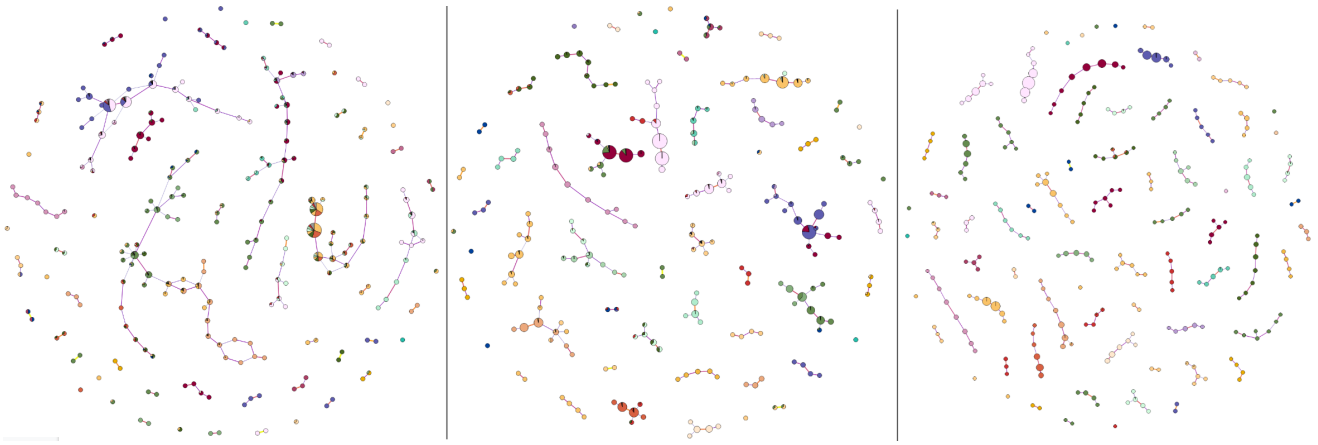


Figure 5. Topological structures of embeddings from the final layer of BERT-base fine-tuned on Supersense-Role task, at updates 5 (left), 70 (middle), and 175 (right).

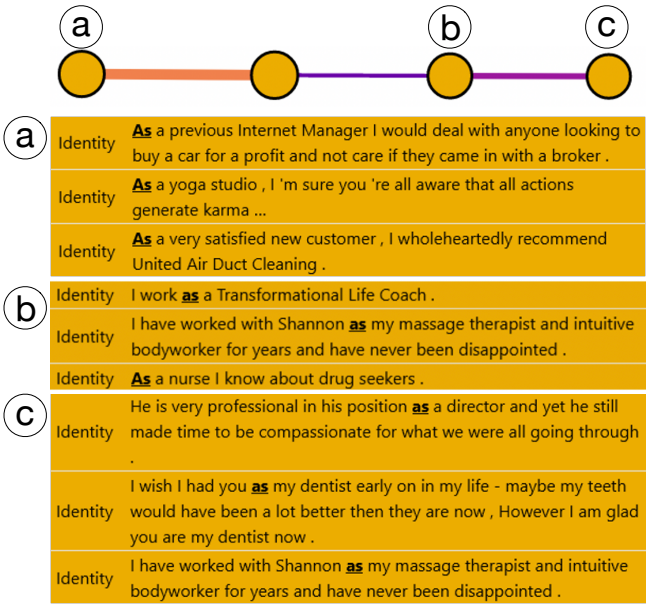


Figure 6. A chain corresponding to the 'Identity' class distinguishes between the fronted (a) and nonfronted usage (b-c) of the word 'as' for embeddings from RoBERTa-base fine-tuned on Supersense-Role.

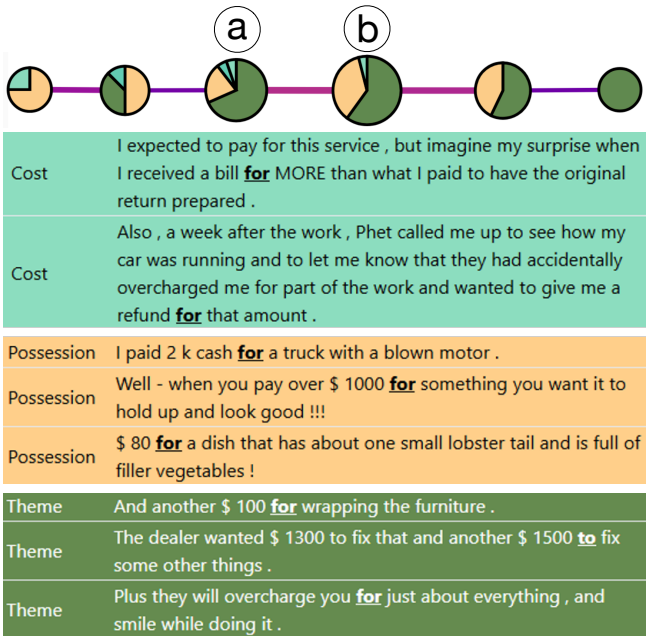


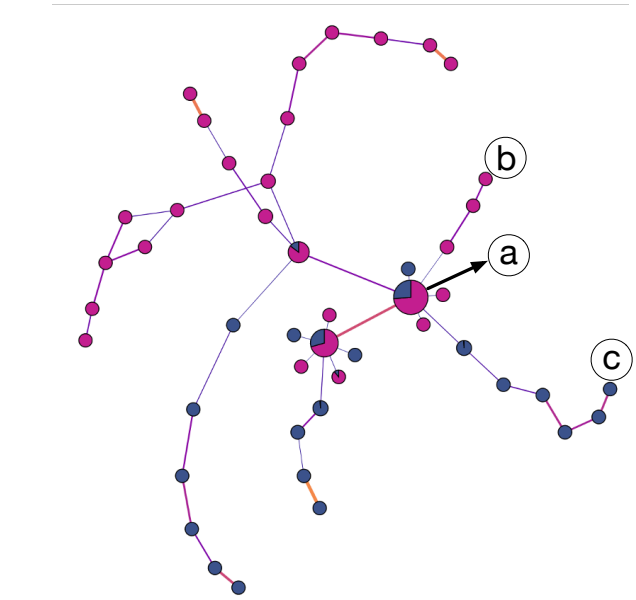
Figure 7. A chain combines embeddings of training examples with labels 'Cost', 'Possession', and 'Theme'. Samples of training examples in nodes (a) and (b) are shown. This chain is from Layer 9 of BERT-base embeddings fine-tuned on Supersense-Role at Update 50.

We see similar labels being confused in the Dependency task, shown in Fig. 8. In the central region of the mapper graph, embeddings from two classes ('amod' in fuchsia and 'compound' in blue) are first clustered together in node (a) and then branched into their own regions in nodes (b) and (c), respectively. In this case, these class labels are linguistically close—per the Universal Dependencies guidelines[102], 'amod' denotes an adjective-noun relation, whereas 'compound' essentially represents a noun-noun relation.

These examples could indicate to the user (either a model developer or an annotation designer) that the embeddings are capturing confounding concepts that could interfere with the prediction of the desired class labels. A model developer may take action to prevent such behavior in the model's learned embeddings, whereas an annotation designer may better clarify these examples and make changes to the annotation scheme if required. Studying these model confusions can help the user improve the task definitions. Additionally, these examples can also provide feedback for improving distinction between easily confused classes by taking actions such as sampling or annotating more training examples that correspond to classes found in the impure chains.

7.3 Error Analysis Through Node Attachments

Analyzing errors in a validation set of examples is a key step in the development and refinement of an ML model. Commonly used metrics such as accuracy, precision, and recall provide overall measures of the performance of a model. Additionally, confusion matrices are used to further understand which classes are frequently misclassified. Through the use of node attachments, TopoBERT provides a more detailed view of the embeddings of validation examples where the model is unable to separate the classes from a

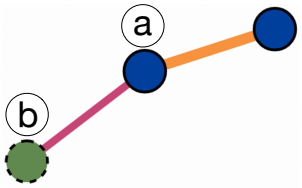


a	amod	It is no surprise that the federal and state governments have classified the National Natural Heritage as a nation-wide **endeavor** of the highest priority and have documented it starting in 2005 in a coalition contract.
	amod	It is no surprise that the federal and state governments have classified the National Natural Heritage as a nation-wide endeavor of the highest **priority** and have documented it starting in 2005 in a coalition contract.
	compound	"While much of the digital transition is unprecedented in the **United** States, the peaceful transition of power is not," Obama special assistant Kori Schulman wrote in a blog post Monday.
	compound	"While much of the digital transition is unprecedented in the United States, the peaceful transition of power is not," Obama special assistant Kori Schulman wrote in a blog post Monday.
b	amod	After all, the internet is not a luxury; it is an essential **tool.**
	amod	As the movie makes clear, the Lovings -- and Richard in particular -- were **reluctant** participants in history.
	amod	Among humans both males and females are ardent singers, and making music is mostly a **communal** activity.
c	compound	First one of **the** Yazidi women started crying, then one of her friends.
	compound	The constituency is in the council area **of** North Kesteven, where 62% of voters backed leaving the EU.
	compound	Businesses had expected to start contracting in July, immediately after the **Brexit** vote, but instead have managed to keep growing steadily.

Figure 8. Connected component from the mapper graph of the BERT-base model fine-tuned on the Dependency task, illustrating that the model's representation groups points with labels 'amod' (blue) and 'compound' (magenta).

topological perspective. It also allows exploration of the examples that the model is likely to confuse, which are candidates for further analysis.

For example, in Fig. 9, an embedding of a validation example with class 'Topic' (green) is attached to a chain of embeddings of training examples with class 'Circumstance'



a	Circumstance	We would highly recommend the El Paradero **on** your next trip to Santa Fe !
	Circumstance	When we expressed our discontent to the manager (that 's right , the manager) , did he say he would return some money , did he say he would give a discount **on** our next visit , did he just say " I 'm sorry " .
	Circumstance	Did n't hurt any that they knew my name by my second visit and greeted me warmly then and **on** my third visit .
	Circumstance	After the way she spoke to me **on** my last visit , I will not be returning !!
b	Topic	Hit or miss **on** the service .

Figure 9. Node attachment of an embedding of a validation example (indicated by the dotted circle boundary) of class 'Topic' (green) to a chain of embeddings of training examples with labels 'Circumstance' (blue).

(blue). The model classifies this validation example as 'Circumstance', confirming that the mapper graph reflects the internal structure of the embeddings. Another interesting aspect of this misclassification is that the target token for all embeddings in Fig. 9 is 'on', which may indicate that the embedding is emphasizing the lexical aspects of the word more than its context.

8 Insight Generation and Validation

The various visual components of TopoBERT allow users to interactively explore the embeddings from the fine-tuning process of transformer-based models. Such an exploration is key to generating insights about the local and global structures of the embeddings, as well as understanding how the fine-tuning process affects the embedding space, as described in the design requirement R3. TopoBERT is built in a modular way that enables qualitative analysis using the frontend visualization as well as subsequent quantitative experiments using the computationally generated data from the backend API.

We first focus on Supersense-Role task using the BERT-base model, and then provide evidence of generalizing these insights to other models and tasks. We also generalize these insights for a number of model-task pairs in the supplementary material. Whereas Sect. 7 focuses on use cases of exploring contextualized word embeddings for linguistic phenomena for a specific model-task pair, this section focuses on studying general principles regarding the evolution of embeddings during fine-tuning across different model-task pairs.

8.1 Organization and Evolution of Embeddings During Fine-Tuning

In this section, we present multiple insights generated using TopoBERT and perform a follow-up analysis to validate them. We focus on BERT-base fined-tuned on Supersensense-Role. However, these insights are generalizable to other model-task pairs, as shown in Sect. 8.2.

Insight 1. *Fine-tuning changes the topological structures of embeddings in higher layers more than in lower layers.*

Kovaleva et al.[103] compared the cosine similarity of the attention layer's weight before and after fine-tuning, and observed that task-specific fine-tuning of transformer-based models leads to more changes in the higher layers of the model. Using TopoBERT, we can interactively observe the topological changes in the embeddings of training examples across all batch updates of the fine-tuning process.

As observed in the top of Fig. 10 (b) and (c), at the beginning of the fine-tuning process, the mapper graphs at layers 9 and 12 contain one large chain with mostly impure nodes and a number of single node islands. Such an observation indicates that before fine-tuning, the data points (i.e., embeddings of training examples) are scattered across the embedding space without much structure to them. On the other hand, embeddings obtained after fine-tuning for layers 9 and 12 show a number of disconnected chains of pure nodes, where points with the same label are grouped closer together in the embedding space. In comparison, we do not observe a qualitative difference in the mapper graphs before and after fine-tuning for layer 1, see Fig. 10 (a).

We also quantify the amount of change in various layers by computing the distance between the mapper graph at each batch update with respect to the mapper graphs before fine-tuning (at batch update 0), using their induced hypergraphs (see Sect. 3.5 for details). Fig. 11 plots the distances for layers 1, 4, 9, and 12. We observe, first, that the distance between the mapper graphs changes rapidly at the beginning of the fine-tuning process and then plateaus. This observation suggests that the embedding space changes the most during the initial fine-tuning batch updates, which is consistent with findings from Zhou et al.[10]. Second, the magnitude of change in the topological structure is greater in later layers (e.g., layers 9 and 12) than in earlier ones (e.g., layers 1 and 4).

Insight 2. *During fine-tuning, the topological neighborhood purity changes more for the higher layers than for the lower layers.*

TopoBERT provides an easy way to inspect neighborhood purities by using pie-chart glyphs for the nodes. Nodes with higher purities have a single or a small number of slices. From visual inspection of the mapper graphs, we observe that the node purities (see Sect. 3.2 for details) change more for the higher layers than for the lower layers. Specifically, the mapper graph nodes of the higher layers are purer than those in the lower layers; see the bottom of Fig. 10 (b) and (c) for examples.

We also verify this insight quantitatively by plotting the kernel density estimate of the distribution of node purities for layers 1, 9, and 12 across batches in Fig. 12. We see a clear shift in the distribution towards higher purity values for layer 9 and 12 as the fine-tuning progresses. In particular, the distribution of node purities for layer 1 does not change much, but the distribution of node purities for layers 9 and 12 concentrate around the value 1 at the end of the fine-tuning process. This observation indicates that as the fine-tuning progresses, more neighborhoods obtain higher purity in terms of their class label composition.

We also quantify the shift in node purities by computing the earth mover's distance of the node purity distribution at each batch update with respect to the mapper graph before fine-tuning (at update 0), see Fig. 13. We observe that the distance remains roughly constant for layer 1, whereas it increases over batch updates for layers 9 and 12.

Insight 3. *The average topological neighborhood purity is correlated with model performance on unseen data.*

From Insight 1, we observe that the mapper graphs of the embeddings from training examples in later layers have higher overall node purity and better label separation during the fine-tuning process. Using TopoBERT, we also observe that a large number of nodes become purer at the beginning of fine-tuning. We conjecture that the purity of mapper nodes may be related to how well the model is able to differentiate between different class labels.

To validate this insight, we plot the average node purities of the mapper graphs (computed on the embeddings of training examples) at various layers along with the accuracy of the validation examples, as shown in Fig. 14 and Fig. 15 left. From the plots, we observe that the average purity for layers 9 and 12 follows the same trend as the accuracy of the validation examples, whereas the node purities for layer 1 remain roughly constant (also observed in Insight 2). This observation indicates a possible correlation between the node purity and model accuracy.

To validate this insight quantitatively, we compute the Pearson correlation coefficient (PCC) between the average purities and the validation accuracy for each layer of BERT-base model in Fig. 15 (left). The highest correlation of 0.930 appears in layer 9, which suggests that layer 9 is capturing structures that may have the best predictive power for the Supersense-Role task that the model is fine-tuned on. Previous studies[75,92] confirm that that this is indeed the case.

Insight 4. *The purity of the neighborhood a validation point (i.e., the embedding of a validation example) attaches to can be used to predict the correctness of the model for that point.*

We are interested in the embeddings of training examples (i.e., training points) as well as the embeddings of validation examples (i.e., validation points). TopoBERT visualizes the attachment of validation points onto the mapper graph of training points, as described in Sect. 3.4. Roughly speaking, for each validation point x, we compute its nearest neighbor y among the training points. We further observe the purity of the node that y belongs to. If y belongs to more than one node in the mapper graph, we compute the average node purity. We then use this average node purity to predict whether the model would correctly classify a validation point or not. Note that this binary classification task is different from the actual task of the model in predicting the class label.

Fig. 16 shows the precision and recall of this binary classifier. The threshold value for the binary classification is estimated using cross-validation, i.e., by splitting the purity values into two sets, one for estimating the threshold and one for testing the prediction of the classifier using the estimated threshold. We compare it against a baseline classifier that always predicts a validation point to be correctly classified. Since the data is imbalanced, at each batch update, nearly 80% of the validation points are correctly classified. We observe that the simple binary classification is able to consistently get higher precision than the baseline. In other

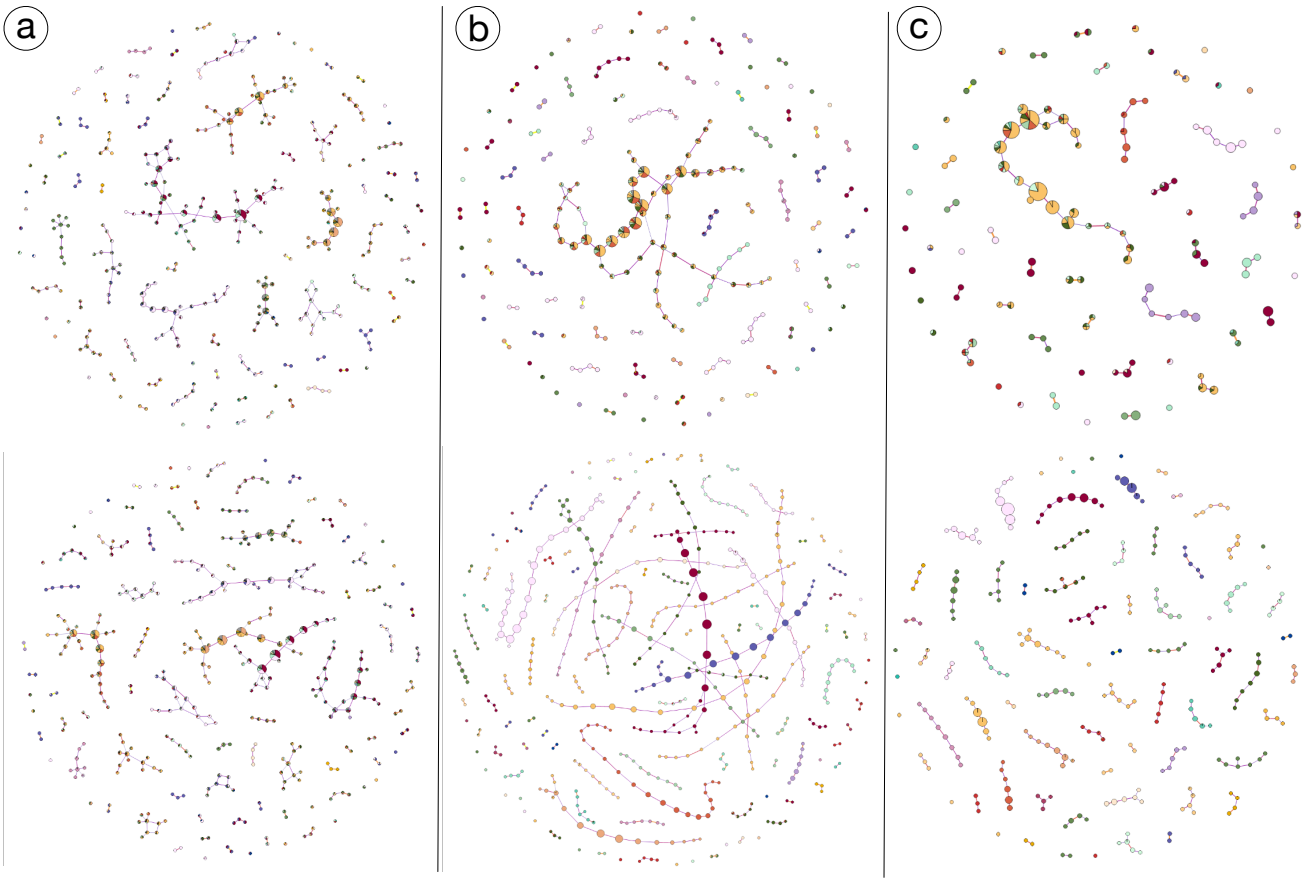


Figure 10. Top: mapper graphs of the embeddings at update 0 before fine-tuning. Bottom: mapper graphs at update 176 after fine-tuning. From left to right: layers 1 (a), 9 (b), and 12 (c), respectively. BERT-base fined tuned on Supersense-Role.

Figure 11. Distance between a mapper graph at each batch update with respect to the mapper graph before fine-tuning, at layers 1, 4, 9, and 12.

words, the attachment node purity has high predictive power for the validation point. The lower recall indicates, however, that this measure misses some validation points that are correctly classified.

Insight 5. *During the fine-tuning process, points of the same label move closer, whereas points of different labels move further away from one another in the embedding space.*

Based on existing studies by Zhou et al.[10,75], we know that the fine-tuning process transforms the embedding space geometrically such that points of the same label move closer and points of different labels move further away from one another. TopoBERT facilitates the validation of this insight from a topological perspective, by generating a mapper graph of the embedding space at every batch update. As shown in Fig. 10 (b) and (c), the mapper graphs before fine-tuning consist of a set of nodes with mixed class labels and poor separation. After fine-tuning, the mapper graphs contain clearly separated chains with pure class labels.

We further corroborate this insight by plotting the t-SNE projection of the embedding space before and after fine-tuning in Fig. 17 and comparing it against the mapper graphs of the same embedding space. The t-SNE projection on the top of Fig. 17 shows that after fine-tuning, the embeddings cluster more tightly with better class separation. The main strength of the mapper graphs is that they better capture the intercluster relationships among points of the same label, and the intracluster relationships among points of different labels, as shown in Fig. 17 bottom.

8.2 Generalization to Other Models and Tasks

We present most of the above insights using word embeddings generated by the BERT-base model fine-tuned on Supersense-Role task. We present evidence in this section that these insights generalize to other model-task pairs.

Starting from the BERT-base model, we show that Insight 2 generalizes from one task to another. As shown in Fig. 18, we observe the same trend in the node purity for embeddings from BERT-base fine-tuned on the Dependency task. Specifically, we compute the node purity distribution for layers 1, 9, and 12 and observe that purities for higher layers change much more than those for lower layers.

Similarly, the insight that average topological purity is correlated with model performance (Insight 3) holds for

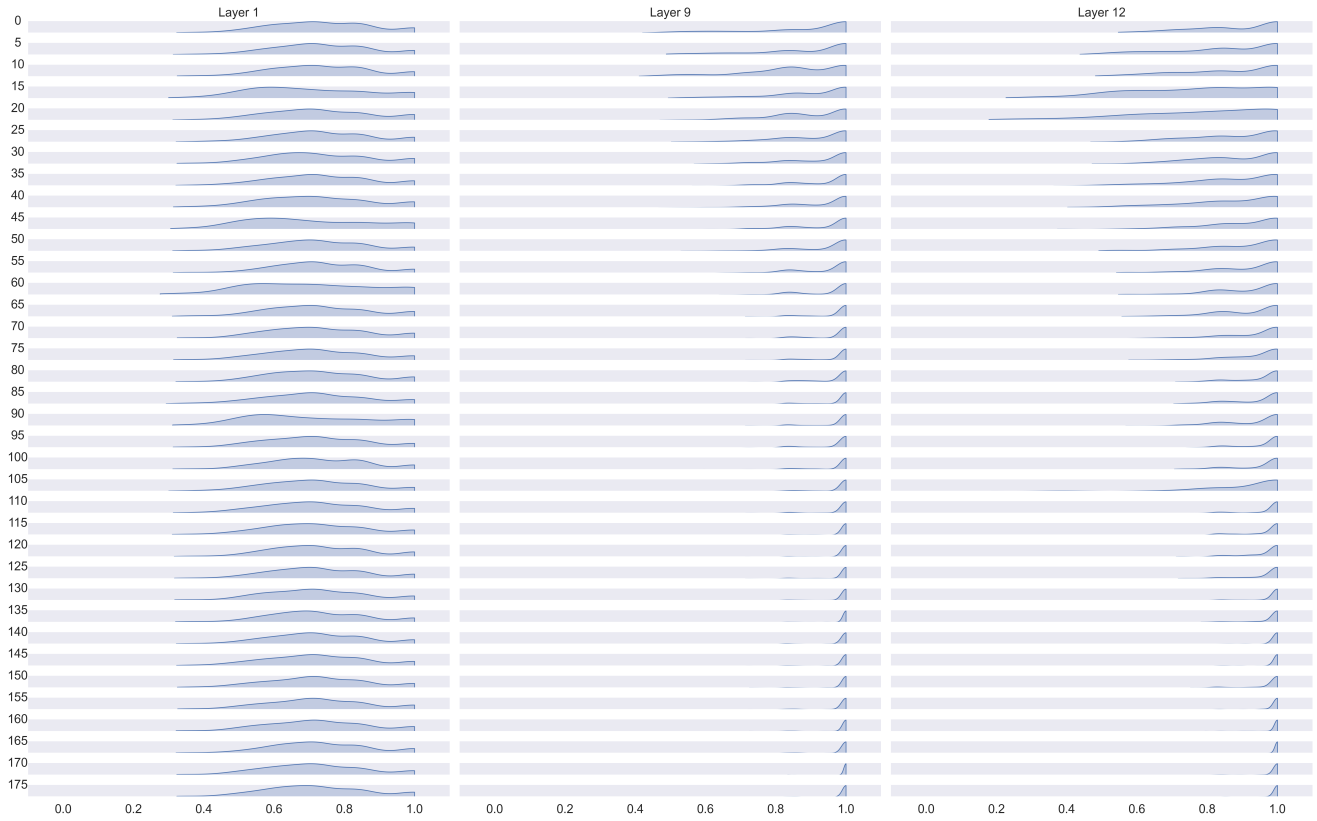


Figure 12. Distribution of node purities over batch updates (along the y-axis) for layers 1, 9, and 12. BERT-base fine-tuned on Supersense-Role.

Figure 13. The earth mover's distance of the node purity distribution at each batch update with respect to the mapper graph before fine-tuning (at update 0).

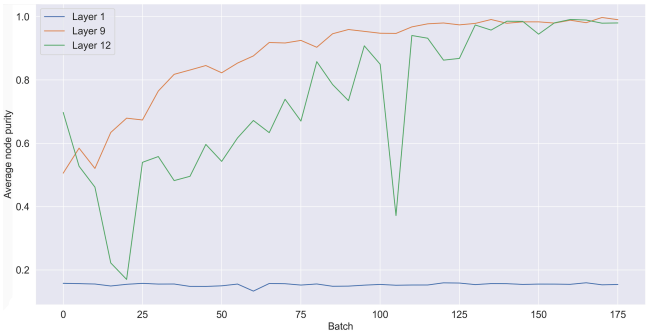
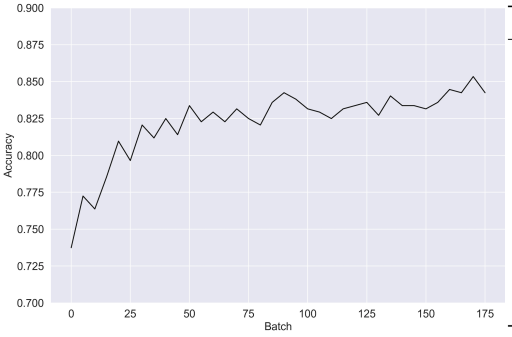


Figure 14. The average node purities of the mapper graphs (constructed from embeddings of training examples) from layers 1, 9, and 12 across batch updates. BERT-base fine-tuned on Supersense-Role.



Layer	PCC
1	0.139
2	0.115
3	0.047
4	0.363
5	0.369
6	0.642
7	0.806
8	0.884
9	**0.930**
10	0.770
11	0.750
12	0.554

Figure 15. Left: the accuracy of validation examples across batch updates. Right: the Pearson correlation coefficients (PCC) between average node purity and the accuracy of validation examples. BERT-base fine-tuned on Supersense-Role.

embeddings from RoBERTa-base model fine-tuned on the Supersense-Role dataset as well. In particular, Fig. 19 shows that the average node purity of higher layers of RoBERTa-base is more correlated with the validation point accuracy than the lower layers. We provide additional evidence of the generalization of the insights in the supplementary material.

9 Expert Evaluation

We conducted a 60-minute tutorial using TopoBERT, followed by 60- to 150- minute semistructured interviews with five domain experts (E1-E5), all of whom have 3 to 7

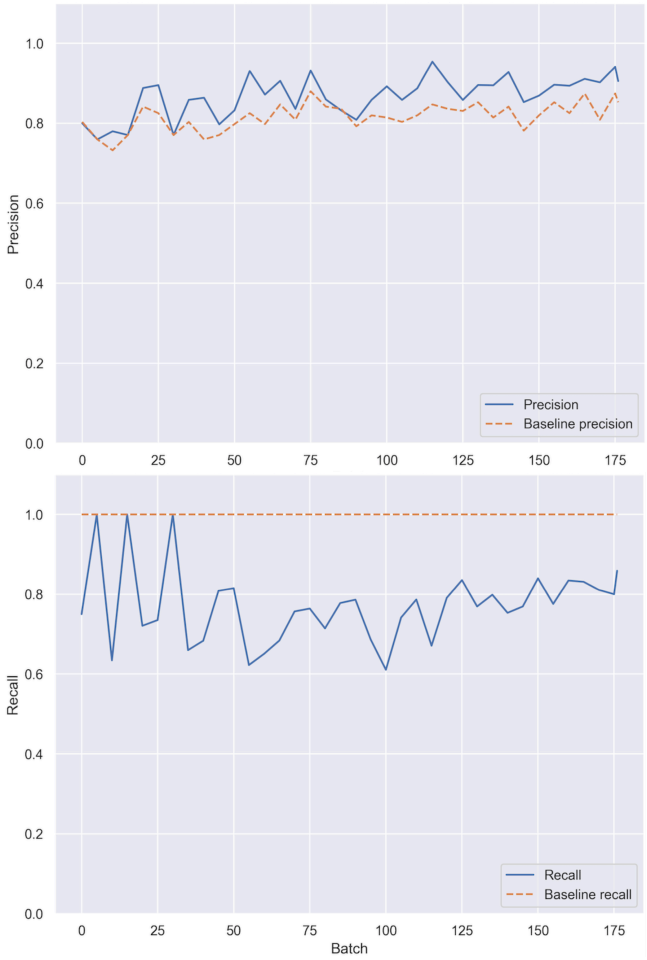


Figure 16. Precision (top) and recall (bottom) curves using a binary classifier based on purities of the attachment nodes of the validation points. Layer 9 of BERT-base fine-tuned on Supersense-Role.

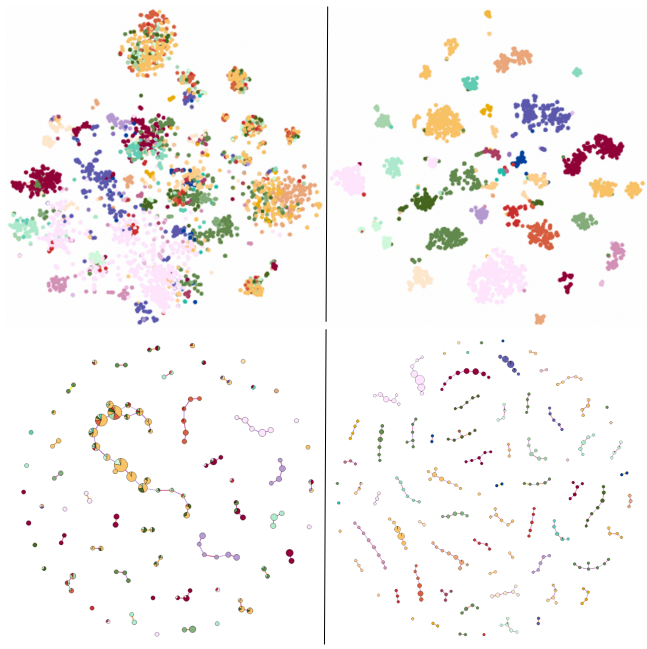


Figure 17. t-SNE projections (top) and mapper graphs (bottom) of the word embeddings from layer 12 at batch update 0 (left) and 176 (right). BERT-base fine-tuned on Supersense-Role.

years of research experiences in NLP. E1, E2, E3, and E4 are 3rd, 4th, 4th and 5th year Ph.D. candidates in Computer Science, and E5 is a Computer Science Ph.D. working in the industry. All of them have published research papers in NLP, and regularly use contextualized embeddings in their work. During the tutorial, we first introduced the three NLP tasks, and their corresponding BERT-type models that were fine-tuned. To establish a knowledge baseline, we explained that checkpoints were saved during fine-tuning to generate embeddings for all layers: for the supersense tasks, each embedding corresponds to a single-token preposition, and for the dependency task, we concatenated contextualized embedding of two tokens and treated the concatenation as the representation of the pair. We then introduced the notion of a mapper graph, demonstrated the visual interface of TopoBERT and its various features, and showcased its exploratory capacities with several case studies. We then solicited their feedback on the utility, usability, and potential improvements of TopoBERT both verbally and in writing. We also collected their comments on the tutorial itself, in particular, on the best way to introduce topological concepts to NLP experts.

In terms of utility, all participants expressed that studying the topology of non-contextualized and contextualized word embeddings is something new to them, and they appreciated the use of visualization to explore the space of embeddings. E1 appreciated the ability to explore and compare structures of embeddings across different layers of the model. E2 stated that the tool gave a direct way to observe how embeddings behave during fine-tuning, and it allowed close investigation of such embeddings. He also appreciated that TopoBERT can be used to explore the models before fine-tuning. He was particularly interested in exploring the linguistic phenomena captured by the pure chains. He also expressed interest in exploring, under the same parameter settings, why longer and purer chains were forming for the dependency task after fine-tuning. E3 wanted to use TopoBERT to study the differences between models. During the tutorial, he placed two instances of TopoBERT side by side to compare embeddings from BERT-base before fine-tuning and BERT-Tiny after fine-tuning. He hypothesized that comparing node purities from these models would help him study their generalizability and distillation of BERT-like models. During the tutorials, E2 and E3 had a debate about the correlation between purities and lengths of chains with model robustness and began to formulate their individual hypotheses.

In terms of usability, all participants found the interface to be easy to use with a short live demo. In particular, an introduction of the mapper graph algorithm followed by a brief Q&A addressed their initial concern on the topological construct. E1 found the interface to be “beautiful”, and thought it was “cool to be able to select particular substructures from the mapper graph, drag and reorder them” for detailed investigations. E5 was impressed by the visualization of embeddings and expressed interest in exporting the visualizations as images for use in research papers. E4 liked the ability to “highlight by class labels”. He also stated that TopoBERT “could be an exceptionally useful tools to view and find annotation errors” and “to choose the best examples for an active learning paradigm.” E2 appreciated that a sufficient amount of NLP relevant

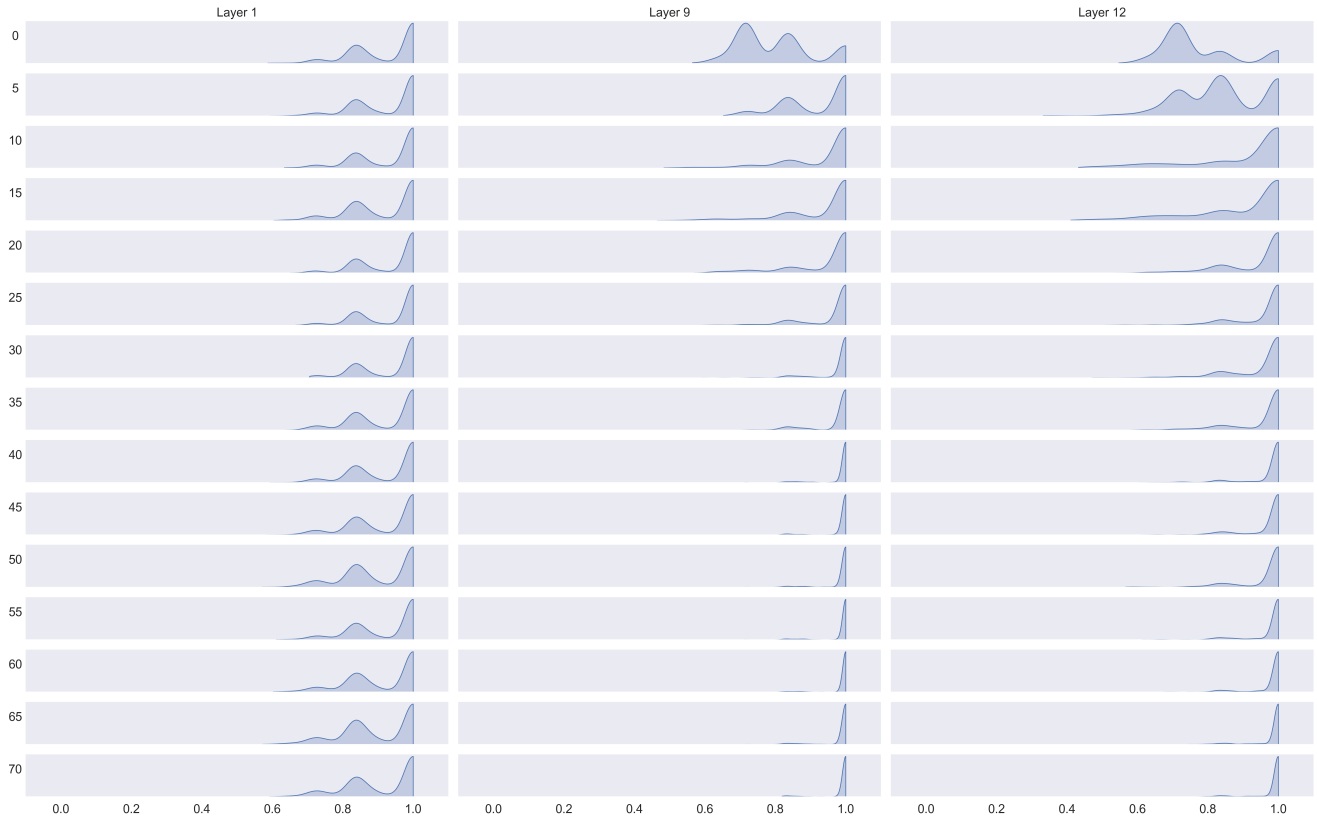


Figure 18. Distribution of node purities for layers 1, 9, and 12. BERT-base fine-tuned on the Dependency task.

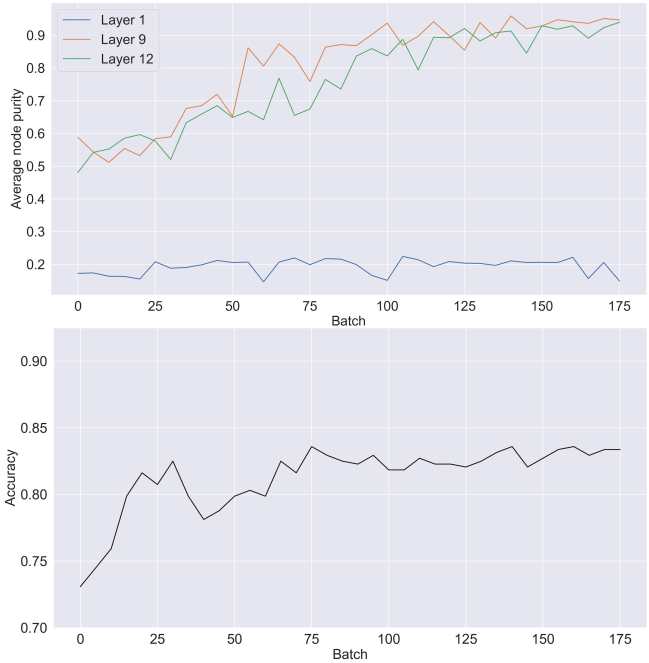


Figure 19. Top: average node purities for layers 1, 9, and 12 across batch updates. Bottom: accuracy of validation points across batch updates. RoBERTa-base model fine-tuned on Supersense-Role task.

information is already included in TopoBERT, in particular, tokens' metadata (labels and sentences). E3 pointed out that PCA is a weak baseline visualization of embeddings due to its occlusions and suggested adding t-SNE as an alternative; although he also acknowledged the scalability issue of t-SNE for real-time computations.

In addition, we asked the participants how they would like to use TopoBERT to assist in their research in the future (if at all). E5 stated that he would like to use TopoBERT for more complex NLP tasks beyond the ones currently in the tool, such as common-sense reasoning. E3 suggested that the mapper graphs of embeddings from the last layers are useful for model diagnostics. Specifically, he would like to "find ambiguous examples and hard examples (e.g., data points from impure branching nodes), train on them" for model improvements, and "annotate less data to get better performance". Furthermore, he hypothesized that he would be able to use purity as an indicator for an early-layer exiting strategy during inference tasks. E2 was interested in exploring small and isolated chains and conjectured that they might contain obscure examples of interest. E4 would like to see TopoBERT applied to an NLP task with fewer labels. For example, he would be curious to know if a 3-class NLI (Natural Language Inference) classifier would form extremely long chains or innumerable tiny islands.

In terms of future improvements, most participants suggested the extension of support for GPT-type models. They believed that TopoBERT is also applicable to a number of NLP classification tasks, such as topic modeling, sentiment analysis, product classification, and name entity tagging. E2 would like to see TopoBERT extended to study multilingual BERT, in particular, comparing English against low-resource languages. E3 would like to compare models side by side (instead of using two instances of TopoBERT simultaneously). He would also like to see additional statistics displayed with each mapper graph: the distribution of node purities, number and lengths of branches, etc. E5 would like to study the evolution of a

single token during the fine-tuning process more easily with precomputed animations. Studying a single-token evolution currently requires manually searching and highlighting the token in the mapper graph at each batch update. He would also like to have an interface to add new token embeddings and update the underlying mapper graph. All participants would like to have enhanced search capabilities beyond user-specified tokens, including searching with labels, token-label pairs, and a drop-down token list. E3 would like to search and filter by node size and node purity as he would like to use TopoBERT to discover data points associated with model confusion for retraining purposes. E2 would like to be able to save intermediate results to form an exploration sequence to be visited later. All participants would like to see a user manual associated with the interface, which is under development.

In terms of the tutorial itself, all participants found that the introduction to topological concepts such as mapper graphs and the mapper algorithm to be appropriate and sufficient for an NLP audience. E4 stated that "the topology section was clear and easy to digest." E2 and E3 were particularly interested in understanding the parameters (the number of intervals n and the amount of overlap p) and their impact on the mapper graphs. E3 was less interested in exploring the different lens functions beyond the default L_2-norm and stated that it was less relevant to the NLP experts. He also pointed out that it is important to differentiate the hierarchical clustering of embeddings from the mapper graph to avoid confusion. E2 and E3 were very excited about the tool's potential and suggested that TopoBERT should be shown as a system demonstration in NLP venues such as ACL and EMNLP to reach a large NLP audience.

10 Conclusion and Future Work

This paper presents TopoBERT, a new tool to examine contextualized word embeddings from a transformer-based model fine-tuned on linguistics tasks. TopoBERT is the first tool (to our knowledge) that employs topological data analysis to interactively probe contextualized embeddings during fine-tuning. Its interface allows users to perform exploratory analysis of global and local structures in embedding spaces. We provide various use cases for the tool that can help bridge the gap between model architects who design statistical NLP models and experts in computational linguistics who design annotation schemes. The modular design of the computational and visualization components of the system facilitates insight generation and validation of various aspects of the fine-tuning process.

In this work, we mainly focus on neural network architectures from the transformer family on a set of semantic tasks using token-based word embeddings. The modular design of TopoBERT means that it is agnostic to the provenance of the embeddings, the specific linguistic tasks being studied, and the objects that are embedded or the language under investigation. TopoBERT can be easily extended to work with other non transformer-based models, such as the LSTM-based ELMo[104] model. Similarly, TopoBERT can easily be extended to analyze embeddings of other objects such as token spans or entire sentences. Whereas the examples presented in this work are English-specific, TopoBERT can also be used to probe and understand the increasingly prevalent non-English and multilingual embeddings (such as XLM-RoBERTa[105]) for their lexical, syntactic, and semantic regularities. Finally, we currently restrict our analysis to models fine-tuned on specific tasks, exploring cross-task performance of models using TopoBERT would be an interesting direction to explore.

Acknowledgements

This work is supported in part by NSF awards DMS 2134223, IIS 2205418, IIS 1513616, IIS 1910733, IIS 1822877, CNS 1801446, III 2007398, and CCF 2217154, and by a generous gift from Verisk Inc.

References

1. Devlin J, Chang MW, Lee K et al. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pp. 4171–4186.
2. Liu Y, Ott M, Goyal N et al. RoBERTa: A robustly optimized BERT pretraining approach. arXiv preprint arXiv:1907.11692, 2019.
3. Vaswani A, Shazeer N, Parmar N et al. Attention is all you need. *Advances in Neural Information Processing Systems* 2017; 30: 5998–6008.
4. Howard J and Ruder S. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, pp. 328–339.
5. Belinkov Y. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics* 2022; 48(1): 207–219.
6. Tenney I, Xia P, Chen B et al. What do you learn from context? probing for sentence structure in contextualized word representations. In *International Conference on Learning Representations (ICLR) 2019*.
7. Immer A, Torroba Hennigen L, Fortuin V et al. Probing as quantifying inductive bias. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Dublin, Ireland: Association for Computational Linguistics, pp. 1839–1851.
8. Lasri K, Pimentel T, Lenci A et al. Probing for the usage of grammatical number. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Dublin, Ireland: Association for Computational Linguistics, pp. 8818–8831.
9. Aghazadeh E, Fayyaz M and Yaghoobzadeh Y. Metaphors in pre-trained language models: Probing and generalization across datasets and languages. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Dublin, Ireland: Association for Computational Linguistics, pp. 2037–2050.
10. Zhou Y and Srikumar V. A closer look at how fine-tuning changes BERT. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Dublin, Ireland: Association for Computational Linguistics, pp. 1046–1061.

11. Singh G, Mémoli F and Carlsson GE. Topological methods for the analysis of high dimensional data sets and 3D object recognition. *Eurographics Symposium on Point-Based Graphics* 2007; : 91–100.
12. Hofer C, Kwitt R, Niethammer M et al. Deep learning with topological signatures. *Advances in Neural Information Processing Systems* 2017; 30: 1634–1644.
13. Rathore A, Chalapathi N, Palande S et al. TopoAct: Exploring the shape of activations in deep learning. *Computer Graphics Forum* 2021; 40(1): 382–397.
14. Gabrielsson RB and Carlsson GE. A look at the topology of convolutional neural networks. arXiv preprint arXiv:1810.03234, 2018.
15. Clough JR, Öksüz I, Byrne N et al. A topological loss function for deep-learning based image segmentation using persistent homology. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 2020; 44: 8766–8778.
16. Hu X, Li F, Samaras D et al. Topology-preserving deep image segmentation. *Advances in Neural Information Processing Systems* 2019; 32: 5657–5668.
17. Chen C, Ni X, Bai Q et al. A topological regularizer for classifiers via persistent homology. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics*, *Proceedings of Machine Learning Research*, volume 89. PMLR, pp. 2573–2582.
18. Rieck B, Togninalli M, Bock C et al. Neural persistence: A complexity measure for deep neural networks using algebraic topology. arXiv preprint arXiv:1812.09764, 2018.
19. Corneanu CA, Escalera S and Martinez AM. Computing the testing error without a testing set. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 2677–2685.
20. Naitzat G, Zhitnikov A and Lim LH. Topology of deep neural networks. *Journal of Machine Learning Research* 2020; 21(184): 1–40.
21. Barannikov S, Sotnikov G, Trofimov I et al. Topological obstructions in neural networks learning. arXiv preprint arXiv:2012.15834, 2020.
22. Doshi P and Zadrozny W. Movie genre detection using topological data analysis. In *International Conference on Statistical Language and Speech Processing*. Springer, pp. 117–128.
23. Savle K, Zadrozny W and Lee M. Topological data analysis for discourse semantics? In *Proceedings of the 13th International Conference on Computational Semantics-Student Papers*. pp. 34–43.
24. Guan H, Tang W, Krim H et al. A topological collapse for document summarization. In *IEEE 17th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. pp. 1–5.
25. Das S, Haque SA, Tanveer M et al. Persistence homology of TEDtalk: Do sentence embeddings have a topological shape? arXiv preprint arXiv:2103.14131, 2021.
26. Kushnareva L, Piontkovski D and Piontkovskaya I. Betti numbers of attention graphs is all you really need. arXiv preprint arXiv:2207.01903, 2022.
27. Kushnareva L, Cherniavskii D, Mikhailov V et al. Artificial text detection via examining the topology of attention maps. arXiv preprint arXiv:2109.04825, 2021.
28. Cherniavskii D, Tulchinskii E, Mikhailov V et al. Acceptability judgements via examining the topology of attention maps. arXiv preprint arXiv:2205.09630, 2022.
29. Perez I and Reinauer R. The topological BERT: Transforming attention into topology for natural language processing. arXiv preprint arXiv:2206.15195, 2022.
30. Ribeiro MT, Singh S and Guestrin C. "Why should I trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. pp. 1135–1144.
31. Liu M, Shi J, Li Z et al. Towards better analysis of deep convolutional neural networks. *IEEE Transactions on Visualization and Computer Graphics* 2016; 23(1): 91–100.
32. Pezzotti N, Höllt T, Van Gemert J et al. DeepEyes: Progressive visual analytics for designing deep neural networks. *IEEE Transactions on Visualization and Computer Graphics* 2017; 24(1): 98–108.
33. Wongsuphasawat K, Smilkov D, Wexler J et al. Visualizing dataflow graphs ofdeep learning models in TensorFlow. *IEEE Transactions on Visualization and Computer Graphics* 2017; 24(1): 1–12.
34. Choo J and Liu S. Visual analytics for explainable deep learning. *IEEE Transactions on Visualization and Computer Graphics* 2018; 38(4): 84–92.
35. Liu S, Wang X, Liu M et al. Towards better analysis of machine learning models: A visual analytics perspective. *Visual Informatics* 2017; 1(1): 48–56.
36. Wang ZJ, Turko R, Shaikh O et al. CNN explainer: Learning convolutional neural networks with interactive visualization. *IEEE Transactions on Visualization and Computer Graphics* 2021; 27(2): 1396–1406.
37. Arrieta AB, Díaz-Rodríguez N, Del Ser J et al. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion* 2020; 58: 82–115.
38. Wexler J, Pushkarna M, Bolukbasi T et al. The What-If Tool: Interactive probing of machine learning models. *IEEE Transactions on Visualization and Computer Graphics* 2020; 26: 56–65.
39. Chen C, Yuan J, Lu Y et al. OoDAnalyzer: Interactive analysis of out-of-distribution samples. *IEEE Transactions on Visualization and Computer Graphics* 2021; 27(1): 3335–3349.
40. Yang W, Li Z, Liu M et al. Diagnosing concept drift with visual analytics. In *IEEE Conference on Visual Analytics Science and Technology (VAST)*. pp. 12–23.
41. Krause J, Perer A and Bertini E. INFUSE: interactive feature selection for predictive modeling of high dimensional data. *IEEE Transactions on Visualization and Computer Graphics* 2014; 20(12): 1614–1623.
42. May T, Bannach A, Davey J et al. Guiding feature subset selection with an interactive visualization. In *IEEE Conference on Visual Analytics Science and Technology (VAST)*. pp. 111–120.
43. Springenberg JT, Dosovitskiy A, Brox T et al. Striving for simplicity: The all convolutional Net. In *Workshop Track Proceedings of the 3rd International Conference on Learning Representations (ICLR)*. San Diego, CA, USA.
44. Rauber PE, Fadel SG, Falcao AX et al. Visualizing the hidden activity of artificial neural networks. *IEEE Transactions on Visualization and Computer Graphics* 2016; 23(1): 101–110.

45. Wang J, Gou L, Yang H et al. GANViz: A visual analytics approach to understand the adversarial game. *IEEE Transactions on Visualization and Computer Graphics* 2018; 24(6): 1905–1917.
46. Kahng M, Andrews PY, Kalro A et al. ActiVis: Visual exploration of industry-scale deep neural network models. *IEEE Transactions on Visualization and Computer Graphics* 2018; 24: 88–97.
47. Hohman F, Park H, Robinson C et al. Summit: Scaling deep learning interpretability by visualizing activation and attribution summarizations. *IEEE Transactions on Visualization and Computer Graphics* 2020; 26(1): 1096–1106.
48. Amershi S, Chickering M, Drucker SM et al. ModelTracker: Redesigning performance analysis tools for machine learning. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. pp. 337–346.
49. Ren D, Amershi S, Lee B et al. Squares: Supporting interactive performance analysis for multiclass classifiers. *IEEE Transactions on Visualization and Computer Graphics* 2016; 23(1): 61–70.
50. Zhang J, Wang Y, Molino P et al. Manifold: A model-agnostic framework for interpretation and diagnosis of machine learning models. *IEEE Transactions on Visualization and Computer Graphics* 2019; 25: 364–373.
51. Wang Q, Alexander W, Pegg J et al. HypoML: Visual analysis for hypothesis-based evaluation of machine learning models. *IEEE Transactions on Visualization and Computer Graphics* 2021; 27: 1417–1426.
52. Garcea F, Famouri S, Valentino D et al. iNNvestigate-GUI - explaining neural networks through an interactive visualization tool. In *Artificial Neural Networks in Pattern Recognition*. pp. 291–303.
53. Yuan J, Chen C, Yang W et al. A survey of visual analytics techniques for machine learning. *Computational Visual Media* 2021; 7(1): 3–36.
54. Chatzimparmpas A, Martins RM, Jusufi I et al. A survey of surveys on the use of visualization for interpreting machine learning models. *Information Visualization* 2020; 19: 207–233.
55. Islam MR, Akter S, Ratan MR et al. Deep visual analytics (DVA): applications, challenges and future directions. *Human-Centric Intelligent Systems* 2021; 1(1-2): 3–17.
56. Hohman F, Kahng M, Pienta RS et al. Visual analytics in deep learning: An interrogative survey for the next frontiers. *IEEE Transactions on Visualization and Computer Graphics* 2019; 25: 2674–2693.
57. Liu S, Bremer PT, Thiagarajan JJ et al. Visual exploration of semantic relationships in neural word embeddings. *IEEE Transactions on Visualization and Computer Graphics* 2018; 24(1): 553–562.
58. Spinner T, Schlegel U, Schäfer H et al. explAIner: A visual analytics framework for interactive and explainable machine learning. *IEEE Transactions on Visualization and Computer Graphics* 2019; 26(1): 1064–1074.
59. Ribeiro MT, Singh S and Guestrin C. Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.
60. Liu S, Li Z, Li T et al. NLIZE: A perturbation-driven visual interrogation tool for analyzing and interpreting natural language inference models. *IEEE Transactions on Visualization and Computer Graphics* 2019; 25(1): 651–660.
61. Chan GYY, Bertini E, Nonato LG et al. Melody: Generating and visualizing machine learning model summary to understand data and classifiers together. arXiv preprint arXiv:2007.10614, 2020.
62. Ming Y, Cao S, Zhang R et al. Understanding hidden memories of recurrent neural networks. In *IEEE Conference on Visual Analytics Science and Technology (VAST)*. pp. 13–24.
63. Strobelt H, Gehrmann S, Pfister H et al. LSTMVis: A tool for visual analysis of hidden state dynamics in recurrent neural networks. *IEEE transactions on visualization and computer graphics* 2017; 24(1): 667–676.
64. Berger M. Visually analyzing contextualized embeddings. In *IEEE Visualization Conference (VIS)*. pp. 276–280.
65. Ji X, Shen HW, Ritter A et al. Visual exploration of neural document embedding in information retrieval: Semantics and feature selection. *IEEE Transactions on Visualization and Computer Graphics* 2019; 25: 2181–2192.
66. Vig J. A multiscale visualization of attention in the transformer model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Florence, Italy: Association for Computational Linguistics, pp. 37–42.
67. Pearson K. LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science* 1901; 2(11): 559–572.
68. Van der Maaten L and Hinton G. Visualizing data using t-SNE. *Journal of Machine Learning Research* 2008; 9(11).
69. McInnes L, Healy J and Melville J. UMAP: Uniform manifold approximation and projection for dimension reduction. arXiv preprint arXiv:1802.03426, 2018.
70. Liu S, Maljovec D, Wang B et al. Visualizing high-dimensional data: Advances in the past decade. *IEEE Transactions on Visualization and Computer Graphics* 2017; 23(3): 1249–1268.
71. Rogers A, Kovaleva O and Rumshisky A. A primer in BERTology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics* 2021; 8: 842–866.
72. Whitney WF, Song MJ, Brandfonbrener D et al. Evaluating representations by the complexity of learning low-loss predictors. In *Neural Compression: From Information Theory to Applications, Workshop at ICLR*.
73. Voita E and Titov I. Information-theoretic probing with minimum description length. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pp. 183–196.
74. Limisiewicz T and Mareček D. Introducing orthogonal constraint in structural probes. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, pp. 428–442.
75. Zhou Y and Srikumar V. DirectProbe: Studying representations without classifiers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pp. 5070–5083.

76. Karidi T, Zhou Y, Schneider N et al. Putting words in BERT's mouth: Navigating contextualized vector spaces with pseudowords. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pp. 10300–10313.
77. Hewitt J and Manning CD. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, pp. 4129–4138.
78. Ethayarajh K. How contextual are contextualized word representations? Comparing the geometry of BERT, ELMo, and GPT-2 embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, pp. 55–65.
79. Peters ME, Ruder S and Smith NA. To tune or not to tune? adapting pretrained representations to diverse tasks. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*. Association for Computational Linguistics, pp. 7–14.
80. Merchant A, Rahimtoroghi E, Pavlick E et al. What happens to BERT embeddings during fine-tuning? In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*. Association for Computational Linguistics, pp. 33–44.
81. Mosbach M, Khokhlova A, Hedderich MA et al. On the interplay between fine-tuning and sentence-level probing for linguistic knowledge in pre-trained transformers. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*. Association for Computational Linguistics, pp. 68–82.
82. Hao Y, Dong L, Wei F et al. Investigating learning dynamics of BERT fine-tuning. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*. Suzhou, China: Association for Computational Linguistics, pp. 87–92.
83. Aleksandroff PS. Über den allgemeinen dimensionsbegriff und seine beziehungen zur elementaren geometrischen anschauung. *Mathematische Annalen* 1928; 98(1): 617–635.
84. Biasotti S, Giorgi D, Spagnuolo M et al. Reeb graphs for shape analysis and applications. *Theoretical Computer Science* 2008; 392: 5–22.
85. Biasotti S, Marini S, Mortara M et al. An overview on properties and efficacy of topological skeletons in shape modelling. In *Shape Modeling International*.
86. Carrière M, Michel B and Oudot S. Statistical analysis and parameter selection for mapper. *Journal of Machine Learning Research* 2018; 19(12): 1–39.
87. Chalapathi N, Zhou Y and Wang B. Adaptive covers for mapper graphs using information criteria. In *IEEE International Conference on Big Data (Big Data)*.
88. Ester M, Kriegel HP, Sander J et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*. pp. 226–231.
89. Zhou Y, Chalapathi N, Rathore A et al. Mapper Interactive: A scalable, extendable, and interactive toolbox for the visual exploration of high-dimensional data. In *Proceedings of the IEEE 14th Pacific Visualization Symposium (PacificVis)*. pp. 101–110.
90. Purvine E, Brown D, Jefferson B et al. Experimental observations of the topology of convolutional neural network activations. In *Proceedings of the 37th AAAI Conference on Artificial Intelligence (AAAI)*.
91. Chowdhury S, Needham T, Semrad E et al. Hypergraph co-optimal transport: Metric and categorical properties. arXiv preprint arXiv:2112.03904, 2021.
92. Liu NF, Gardner M, Belinkov Y et al. Linguistic knowledge and transferability of contextual representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, pp. 1073–1094.
93. Nivre J, de Marneffe MC, Ginter F et al. Universal Dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC)*. Portorož, Slovenia: European Language Resources Association (ELRA), pp. 1659–1666.
94. Turc I, Chang MW, Lee K et al. Well-read students learn better: On the importance of pre-training compact models. arXiv preprint arXiv:1908.08962, 2019.
95. Wolf T, Debut L, Sanh V et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, pp. 38–45.
96. Loshchilov I and Hutter F. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*.
97. Lloyd S. Least squares quantization in PCM. *IEEE Transactions on Information Theory* 1982; 28(2): 129–137.
98. Ester M, Kriegel HP, Sander J et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, volume 96. pp. 226–231.
99. Shneiderman B. The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of IEEE Symposium on Visual Languages*. pp. 336–343.
100. Abadi M, Agarwal A, Barham P et al. TensorFlow: Large-scale machine learning on heterogeneous systems. https://www.tensorflow.org/, 2015.
101. Mikolov T, Sutskever I, Chen K et al. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, volume 26. pp. 3111–3119.
102. de Marneffe MC, Dozat T, Silveira N et al. Universal Stanford dependencies: A cross-linguistic typology. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC)*. Reykjavik, Iceland: European Language Resources Association (ELRA), pp. 4585–4592.
103. Kovaleva O, Romanov A, Rogers A et al. Revealing the dark secrets of BERT. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China:

Association for Computational Linguistics, pp. 4365–4374.

104. Peters ME, Neumann M, Iyyer M et al. Deep contextualized word representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1 Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, pp. 2227–2237.
105. Conneau A, Khandelwal K, Goyal N et al. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. pp. 8440–8451.

11 Supplementary Materials

We give additional examples of generalizing insights from Sect. 8 to other model-task pairs.

While Insight 1 is applicable to BERT-base fine-tuned on the Supersense-Role task, it is also observed for BERT-base fine-tuned on the Dependency task, as shown in Fig. 20. Specifically, we observe that after fine-tuning, the embeddings for layer 1 remain largely unchanged, while the embeddings for layers 9 and 12 contain distinct chains of points with the same class labels.

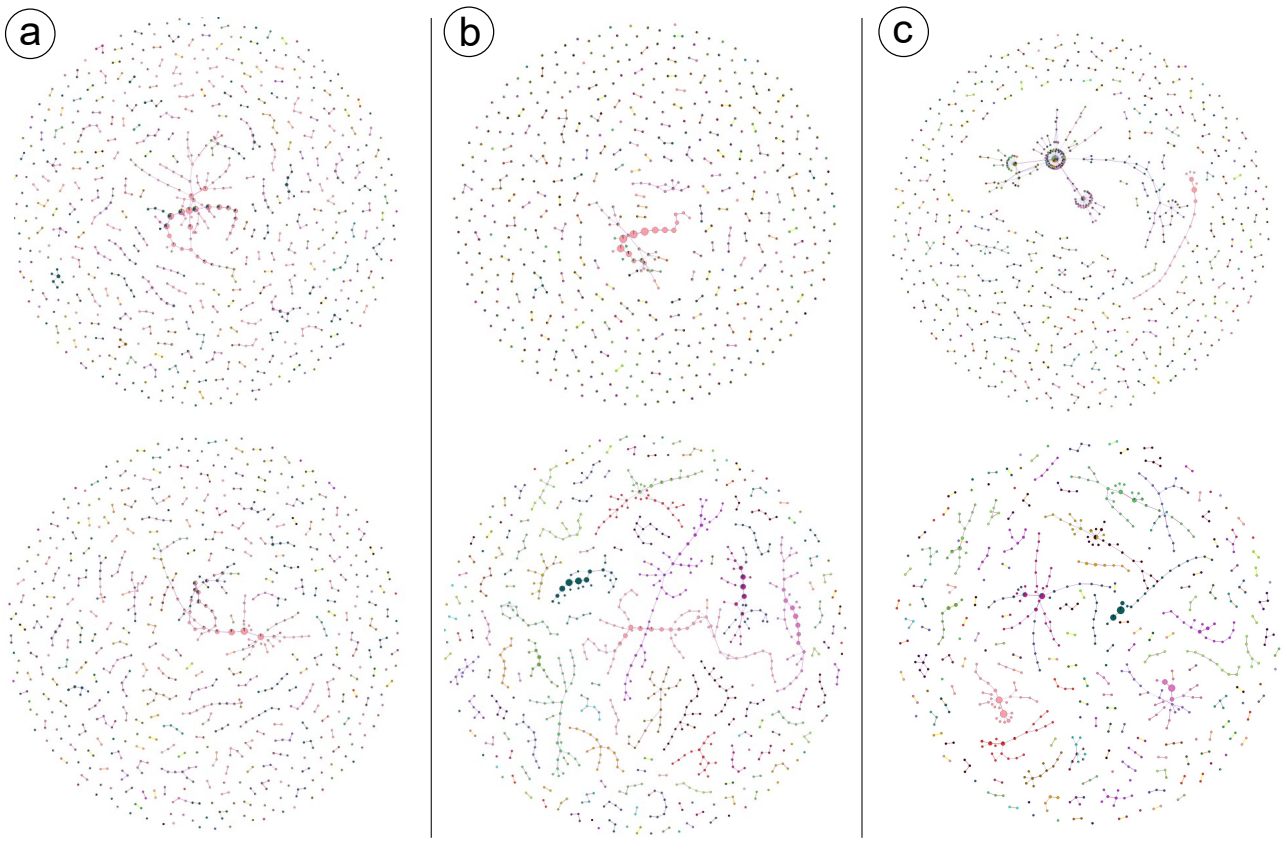


Figure 20. Mapper graphs before (top, batch update 0) and after (bottom, batch update 176) fine-tuning, for layers 1 (a), 9 (b), and 12 (c). BERT-base fined-tuned on the Dependency task.

We also plot the distances between mapper graphs at each batch update and the mapper graph before fine-tuning, for BERT-base fined-tuned on the Dependency task. This is shown in Fig. 21. Again, we observe that the distance changes rapidly at the beginning of the fine-tuning process.

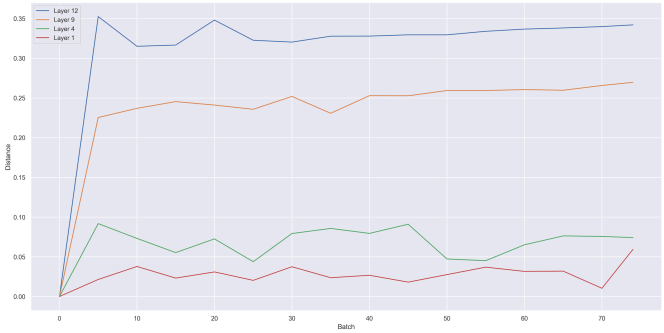


Figure 21. Distance between the mapper graph at each batch update with respect to the mapper graph before fine-tuning, for layers 1, 4, 9, and 12. BERT-base fined-tuned on the Dependency task.

In terms of Insight 2, we can observe that the node purity (for nodes in higher layers) increases for tasks beyond the Supersense-Role. We quantify this observation by plotting the kernel density estimate of the node purities for layers 1, 9, and 12, for embeddings from BERT-base fine-tuned on the Dependency task, and RoBERTa-base fine-tuned on the Supersense-Role task; this is shown in Fig. 22. We also quantify the shift in purities by computing the earth mover's distance, as shown in Fig. 23.

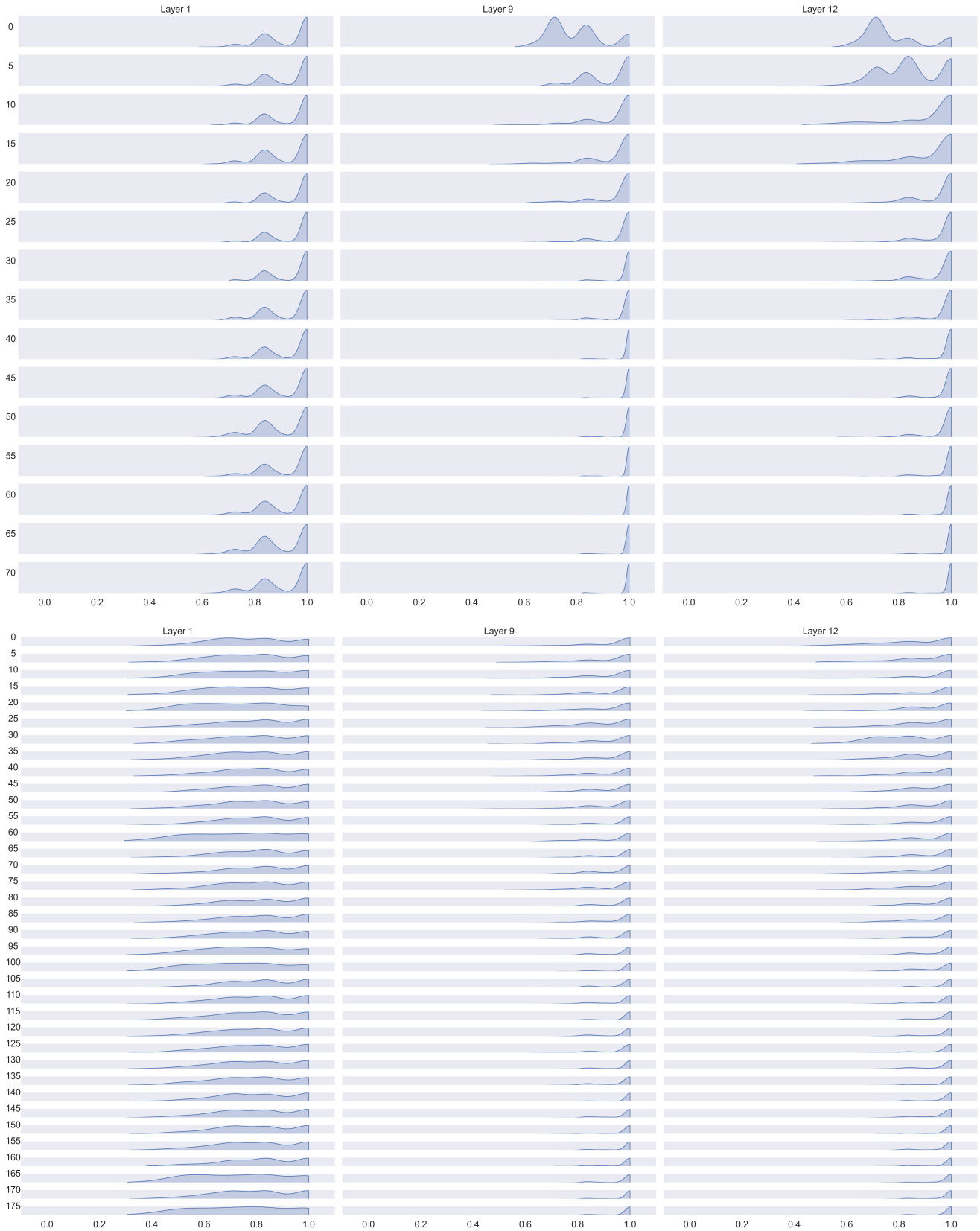


Figure 22. Node purity distributions for layers 1, 9, and 12. Top: BERT-base fined-tuned on the Dependency task. Bottom: RoBERTa-base fined-tuned on the Supersense-Role task.

In terms of Insight 3, we observe similar trends in correlating the node purity with the model performance on the unseen data, for BERT-base fine-tuned on the Dependency task, as shown in Fig. 24. Specifically, the correlation between the node purity and the model performance is higher for layers 9 and 13 than layer 1.

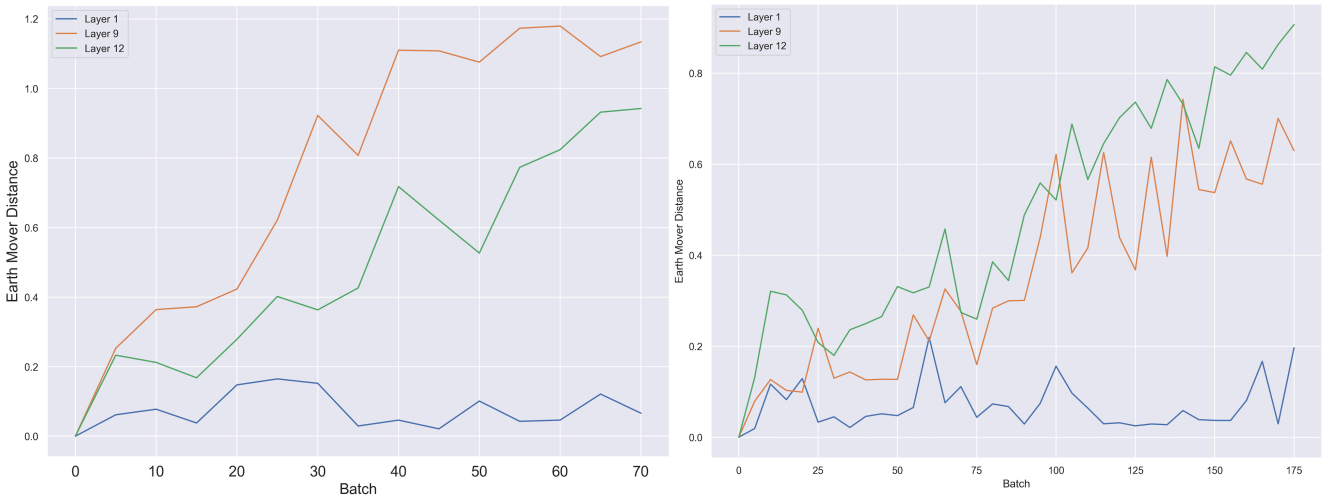


Figure 23. The earth mover's distance of the node purity distribution at each batch update with respect to the mapper graph before fine-tuning. Left: BERT-base fined-tuned on the Dependency task. Right: RoBERTa-base fined-tuned on the Supersense-Role task.

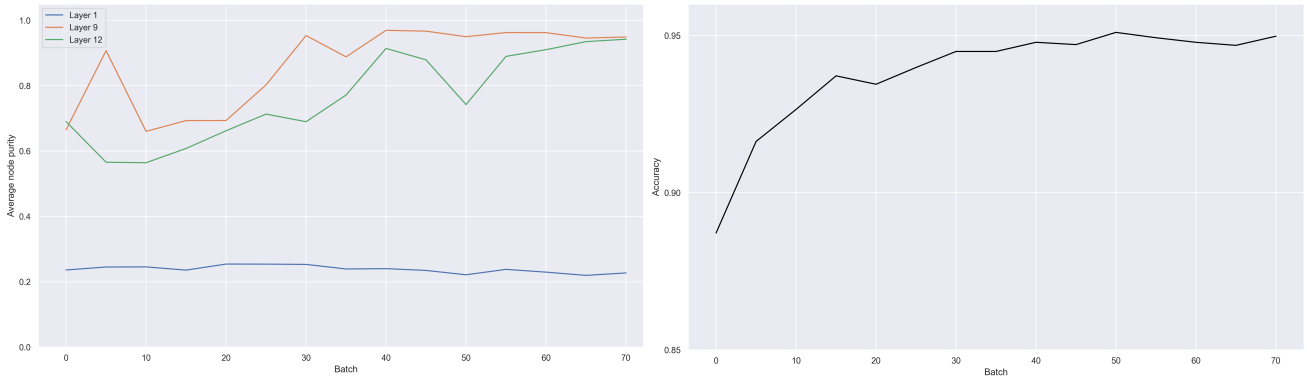


Figure 24. Left: Average node purities of embeddings from layers 1, 9, and 12 across batch updates. Right: Accuracy of the validation points across batch updates. BERT-base fine-tuned on the Dependency task.

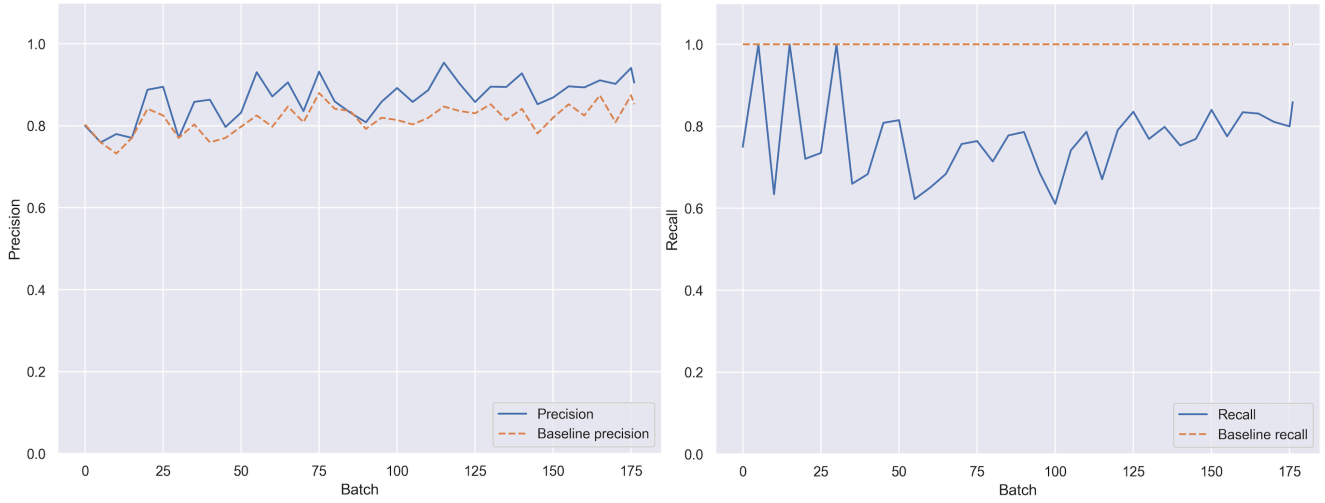


Figure 25. Precision (left) and recall (right) curves using a binary classifier based on the purity of the attachment nodes. RoBERTa-base fined tuned on the Supersense-Role task.

In terms of Insight 4, we replicate our experiments for RoBERTa-base fine-tuned on the Supersense-Role task. As shown in Fig. 25, the purity of the node a validation point attaches to can be used to predict the correctness of the model for that point.

Finally, we discuss dimensionality reduction for comparing word embeddings. We apply different dimensionality reduction techniques, PCA, t-SNE, and UMAP (Fig. 26) to demonstrate that traditional dimensionality reduction methods do not capture detailed intercluster and intracluster relations among the embeddings, comparing with the mapper graphs.

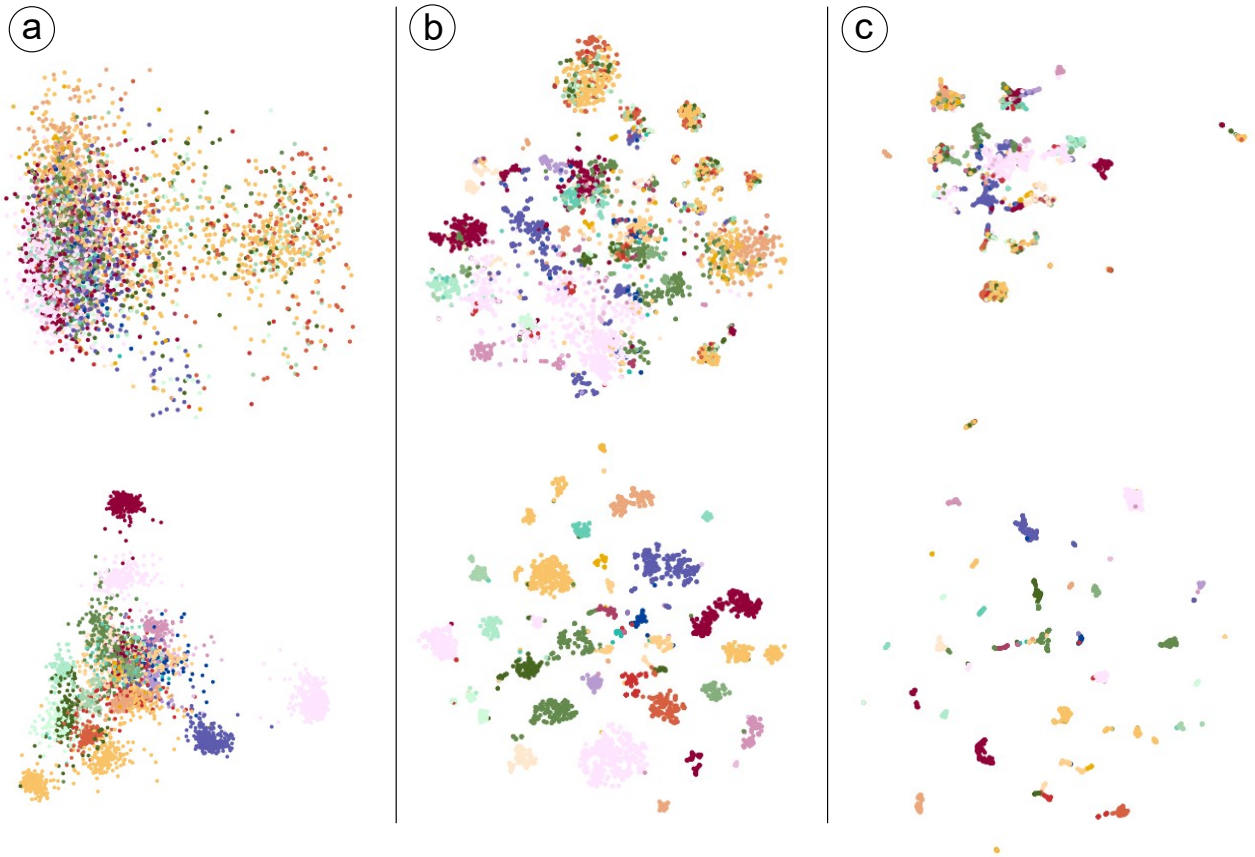


Figure 26. Dimensionality reduction with PCA (a), t-SNE (b), and UMAP (c) on the embeddings before (top) and after (bottom) fine-tuning. BERT-base fine-tuned on the Supersense-Role task.