Explainable Mapper: Charting LLM Embedding Spaces using Perturbation-Based Explanation, Explanation, and Verification Agents

Xinyuan Yan, Rita Sevastjanova, Sinie van der Ben, Mennatallah El-Assady, and Bei Wang

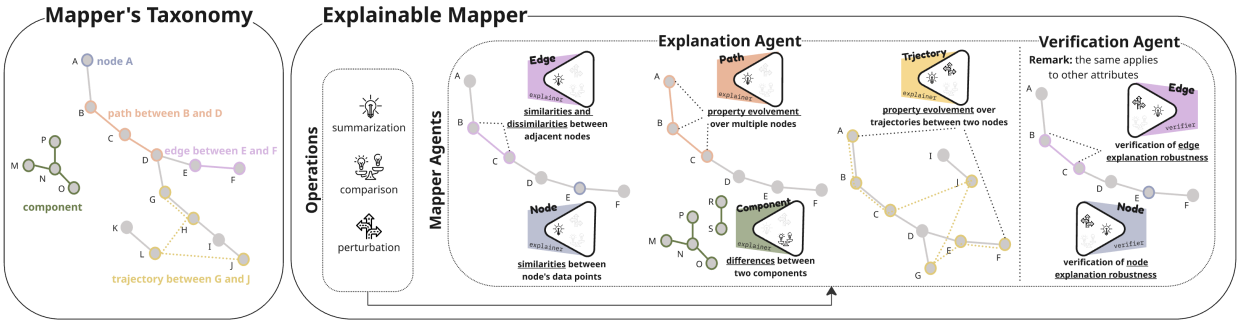

Fig. 1: We use mapper graphs, a common tool in topological data analysis and visualization, to study the topology of LLM embedding spaces. Mapper elements include nodes, edges, paths, components, and trajectories. We present the *Explainable Mapper* workspace and two mapper agents for embedding analysis. These agents use summarization, comparison, and perturbation to generate and test explanations of mapper elements, including cluster linguistics, connectivity, and transitions.

Abstract— Large language models (LLMs) produce high-dimensional embeddings that encode rich linguistic structure. We leverage mapper graphs, a technique from topological data analysis, to examine the organization of these embedding spaces, where nodes represent clusters of embeddings and edges capture their overlap. Although mapper graphs can reveal meaningful patterns, interpreting the linguistic properties they encode remains time-consuming and labor-intensive. We introduce *Explainable Mapper*, a framework for semi-automatic annotation and systematic exploration of mapper graphs that enables fine-grained analysis of embedding spaces. *Explainable Mapper* defines a taxonomy of explorable elements, including nodes, edges, paths, components, and trajectories, and integrates two types of LLM-based agents: an Explanation Agent that generates candidate interpretations of mapper elements, and a Verification Agent that evaluates their robustness using perturbation-based strategies. Together, these agents enable semi-automated and stable interpretation of topological structures. We instantiate *Explainable Mapper* within an interactive visual analytics workspace that combines agent-driven analysis with user-guided exploration, supporting diverse human–AI workflows for scalable and interpretable investigation. We evaluate our approach through case studies on embeddings from multiple transformer models and datasets, as well as an expert user study. Results show that *Explainable Mapper* reproduces established linguistic patterns while uncovering novel insights into LLM embedding spaces. This work extends mapper-based embedding analysis by integrating LLM-assisted explanation generation with perturbation-based verification, enabling scalable and systematic interpretation of high-dimensional embedding spaces.

Index Terms—Topological data analysis, explainable AI, large language models, visual analytics, embedding spaces

1 Introduction

Large language models (LLMs) acquire linguistic properties by training on vast text corpora and demonstrate remarkable performance across a wide range of natural language processing (NLP) and understanding (NLU) tasks. LLMs produce contextual word embeddings as internal representations that encode the semantic and syntactic information used for prediction, and interpreting these embeddings provides insight into how the models understand language and represent meaning [43, 69]. Encoder-model embeddings are particularly prevalent in similarity-driven tasks, which makes it essential to understand the types of information they capture and how this underpins their effectiveness across applications.

A common approach to exploring the embedding space involves dimensionality reduction combined with visualization [20]. Dimensionality reduction maps embeddings into a lower-dimensional space, typically visualized as scatter plots [20] to examine word neighborhoods [4], capture semantic and syntactic similarities [50], track shifts in word meanings [48], and uncover potential biases [40, 47]. These tasks help NLP researchers understand, diagnose, and ultimately refine language models [20]. Researchers have recently leveraged *mapper graphs* [52], a popular tool in topological data analysis and visualization, to investigate the embedding spaces [39, 41]. Unlike dimensionality reduction, which compresses and often distorts high-dimensional structures, mapper graphs preserve multi-scale topological relationships. Specifically, a mapper graph encodes the skeleton of embedding spaces, where each node represents a cluster (a topological neighborhood), and an edge connects overlapping clusters. By constructing a graph representation, mapper graphs reveal clusters, transitions, and connectivity patterns in the embedding space [36, 39, 41, 67, 68].

To interpret the embedding properties represented in a mapper graph, it is essential to analyze its various structural elements. Prior work [39, 41] has shown that elements, such as nodes, paths, and components, carries insights into how embeddings are organized and evolve across contexts. For instance, nodes represent embedding clusters with shared characteristics, and summarizing them may reveal dominant semantic or syntactic features. Paths, representing a sequence of connected nodes, can expose semantic shifts within the embedding space.

However, analyzing a mapper graph of word embeddings is challenging due to the vast and complex explorable space. First, a mapper graph often visualizes thousands of tokens simultaneously to provide meaningful insights. As a result, users must manually inspect tokens and

• *Xinyuan Yan and Bei Wang are with the University of Utah. E-mails: xinyuan.yan@utah.edu, beiwang@sci.utah.edu*
• *Rita Sevastjanova, Sinie van der Ben, and Mennatallah El-Assady are with ETH Zürich. E-mails: rita.sevastjanova@inf.ethz.ch, vandersi@ethz.ch, menna.elassady@ai.ethz.ch*

Manuscript received xx xxx. 202x; accepted xx xxx. 202x. Date of Publication xx xxx. 202x; date of current version xx xxx. 202x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.202x.xxxxxxx

their surrounding contexts (i.e., sentences) to summarize and compare different elements of the mapper graph, which demands considerable human effort [39, 41]. Moreover, word embeddings encode multiple linguistic properties simultaneously [43], meaning there is often more than one plausible observation that can characterize a given mapper element. Since the properties encoded in embeddings are not yet fully understood, users frequently generate new hypotheses while exploring the embedding space and strive to verify their validity. This process of hypothesis generation can *diverge*, producing a large set of potential explanations, while the subsequent verification step seeks to *converge* this space to those hypotheses that can be verified.

To assist expert users, i.e., model developers, NLP researchers, and computational linguists, in analyzing the structural elements of the mapper graph and its underlying high-dimensional latent space, we introduce the concept of *mapper agents* that generate and verify explanations for mapper elements (see Section 5.2). In particular, we introduce two types of agents, i.e., an *Explanation Agent* that can be used to create explanation candidates of a selected mapper element and a *Verification Agent* that is used to verify the generated explanations against their robustness. These two agent types perform distinct operations, i.e., *summarization*, *comparison*, and *perturbation*, to support explanation generation and verification. They work together to help model developers and NLP researchers better understand and diagnose model behavior, while enabling computational linguists to explore the linguistic characteristics embedded in their datasets.

In this paper, we define a taxonomy of mapper elements, including nodes, edges, paths, components, and trajectories, and propose corresponding LLM-based instances of the two agent types, i.e., explainers and verifiers, to explain and verify each element. Throughout this paper, we use the term *agent* to refer to a specialized LLM-based component that performs predefined analytical tasks using structured prompting and perturbation analysis, without autonomous planning or independent decision-making. We instantiate this concept within a visual analytics workspace called *Explainable Mapper*, which enables users to effectively explore the embedding space, interact with mapper elements, and produce robust insights into embedding properties. We evaluate our framework on word embeddings generated by five language models across two datasets. Through multiple use cases, we show that it not only reproduces previously reported findings on BERT but also uncovers new insights into the hidden spaces of both BERT and ModernBERT from a topological perspective. An expert study with six participants demonstrates that our workspace supports effective analysis of mapper graphs and provides actionable feedback for future improvements. Our contributions include:

- ***Explainable Mapper* framework.** We introduce *Explainable Mapper*, a framework for semi-automatic annotation and systematic exploration of mapper graphs that summarize the high-dimensional structure of contextual word embeddings. Extending prior mapper-based embedding analysis (e.g., TopoBERT [41]), *Explainable Mapper* provides a taxonomy of explorable mapper elements and combines LLM-assisted explanation generation with perturbation-based robustness verification to enable scalable linguistic and structural interpretation of embedding spaces.
- **LLM-based mapper agents.** We propose two types of LLM-driven agents: (1) an Explanation Agent that generates candidate explanations for mapper elements, and (2) a Verification Agent that tests the robustness of these explanations using perturbation-based techniques.
- **Visual analytics workspace enabling diverse workflows.** We implement *Explainable Mapper* within an interactive visual analytics workspace that integrates LLM-assisted analysis with user-driven exploration to support scalable, interpretable investigation of embedding structures through multiple human-AI workflows.
- **Empirical validation and expert evaluation.** Through case studies on embeddings from five transformer models and two datasets, and an expert user study, we show that *Explainable Mapper* reproduces known linguistic findings and uncovers novel insights into contextual word embedding spaces.

2 Technical Background

We review the technical background underlying mapper graphs—the key ingredients of *Explainable Mapper*.

Given a high-dimensional point cloud $\mathbb{X}$ equipped with a real-valued function $f : \mathbb{X} \to \mathbb{R}$, a (classical) mapper graph [52] summarizes the topological structure of $\mathbb{X}$ induced by the function f. A node in the mapper graph represents a *topological neighborhood*, whereas an edge connects two nodes if their corresponding neighborhoods overlap. In the context of this paper, $\mathbb{X}$ is a set of word embeddings obtained from an LLM. f is also referred to as the *lens*, as it provides a perspective through which we examine the data. Different lens choices, such as density and eccentricity, offer distinct insights [52]. In this work, we adopt the L_2-norm as the lens function. The L_2-norm of a word embedding captures how strongly an LLM is activated by the input token, and has been shown to yield meaningful results when analyzing hidden representations from deep learning models [39, 41].

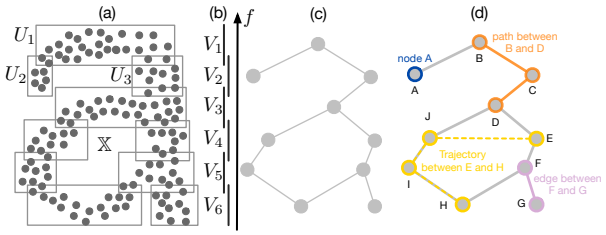

Fig. 2: An example of computing a mapper graph. A point cloud $\mathbb{X}$ in (a) is equipped with a height function f. The cover $\mathcal{U} = \{U_1, U_2, \dots\}$ of $\mathbb{X}$ is induced by a cover $\mathcal{V} = \{V_1, V_2, \dots\}$ of $f(\mathbb{X})$ that contains six intervals with $1/4$ overlap in (b). The 1-dimensional nerve of $\mathcal{U}$ is the mapper graph in (c). Elements derived from the mapper graphs are shown in (d).

For simplicity, we illustrate the construction of a mapper graph from a 2D point cloud $\mathbb{X}$ sampled from a toy dataset in the shape of a letter '*a*'. As shown in Figure 2(a), $\mathbb{X}$ is equipped with a height function f. To obtain a topological summary of $\mathbb{X}$, we start with a finite cover $\mathcal{V} = \{V_1, V_2, \dots\}$ of $f(\mathbb{X}) \subset \mathbb{R}$ formed by a set of partially-overlapping intervals, such that $f(\mathbb{X}) \subseteq \bigcup_j V_j$, as shown in Figure 2(b). In other words, the function value $f(x)$ of each point $x \in \mathbb{X}$ lies within some interval V_j. We then consider the clusters induced by points in $f^{-1}(V_j)$ (for each j) that are enclosed by rectangles. These clusters are denoted as $\mathcal{U} = \{U_1, U_2, \dots\}$. For instance, the points in $f^{-1}(V_1)$ form a single cluster U_1, while those in $f^{-1}(V_2)$ give rise to two clusters, U_2 and U_3, as illustrated in Figure 2(a).

We then construct the 1D *nerve* of $\mathcal{U}$ as the *mapper graph*: each cluster 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.

A mapper graph is constructed with additional parameters in mind. The cover $\mathcal{V}$ is controlled by the number of intervals n and the overlap p. While these values are typically hand-tuned by users, studies have explored automatic parameter selection [6, 7]. We use DBSCAN [13] as the clustering method, which requires two parameters: minPts (minimum points in a neighborhood to form a core point) and ε (maximum radius/distance for points to be considered neighbors). *Explainable Mapper* estimates ε automatically using the elbow method [13, 67] and allows users to set minPts (defaulting to 3). In this work, we apply mapper graphs to explore the LLM embedding spaces. We introduce *Explainable Mapper* to support interpreting a taxonomy of mapper elements, including nodes, edges, paths, components, and trajectories (Section 4). See Figure 2(d) for an illustration.

An alternative to classical mapper is the *ball mapper* [11], which covers $\mathbb{X}$ with balls of radius ε (the same parameter as in DBSCAN), eliminating dependence on the lens f. The 1D nerve of this cover is the *ball mapper graph*. Here, each node represents a *geometric neighborhood* induced by the metric on $\mathbb{X}$, rather than the topological neighborhoods of classical mapper; the same mapper taxonomy applies. In this paper, *Explainable Mapper* supports exploring LLM embedding spaces with both classical and ball mapper graphs, but we focus primarily on classical mapper due to its stronger theoretical foundations and established use in embedding analysis.

3 Related Work

In the following, we review related work on embedding visualization, text perturbation and summarization, and mapper-based visualization of high-dimensional data. Our work also broadly relates to the domain of human–AI collaboration, where interactive workspaces support experts in interpreting embedding spaces with LLM-based agents. Prior research examines this area from multiple angles, including the design space [17], communication dynamics [12], design guidelines [2], and persistent challenges [64]. While we draw on insights from these studies, our primary focus here is on the mechanisms underlying *Explainable Mapper*, which we detail in the remainder of this section.

3.1 Embedding Visualization

Since the introduction of static embeddings such as Word2Vec [31] and GloVe [35], various visualization techniques have been proposed, initially targeting analogy exploration [27] and local neighborhoods [16]. With the advent of contextual language models, properties such as contextual variability become important, prompting the development of visualizations that capture finer-grained semantic differences across contexts. Most current methods examine contextual neighborhoods using dimensionality reduction and scatter plots [4, 47–49, 53].

Huang et al. [20] survey embedding visualization approaches that analyze changes across models [4] and layers [47, 48, 50], investigate encoded linguistic properties and word semantics [3, 48, 50], and examine various forms of bias [47]. Recent work enriches projection-based views with annotations for better interpretability. Sevastjanova et al. [47] use glyphs to highlight dense regions, and KeywordScape [56] uses keyword-based Voronoi maps to depict semantics. Systems such as WizMap [60] and Embedding Atlas [42] enable scalable embedding projections. In contrast, we use mapper graphs to reveal linguistic knowledge in word embedding spaces from a topological perspective.

3.2 Text Perturbation and Summarization

Text perturbation methods improve model robustness and interpretability in NLP by modifying text at linguistic levels ranging from characters to sentences using various techniques [32, 44]. Early methods are largely rule-based [32], while later work shifts toward automated generation using LLMs [14, 24], avoiding the limitations of manual rules. Recent approaches further exploit LLM-based generation: Qiang et al. [38] use LLMs to paraphrase sentences to improve another model's performance, and VarBench [37] mitigates LLM data contamination by dynamically perturbing variables in test cases.

Perturbation methods are also widely used to assess the stability of explanations and interpretability mechanisms [18]. Marjanovic et al. [28] evaluate outputs and explanations of various LLMs under input perturbations and show that robustness varies by model, explanation method, and perturbation type. Instead of applying random changes, SyntaxShap [1] perturbs text by removing words while preserving grammatical coherence. The authors also use perturbations to evaluate explanations by comparing attribution scores before and after perturbation; robust explanations yield minimal score changes.

In our study, we use both perturbations and LLM-generated summaries to help experts extract information from embedding vectors. Prior work shows that LLMs produce summaries comparable to human-written ones. Large-scale benchmarks [66] demonstrate that instruction-tuned LLMs outperform traditional abstractive summarization models, particularly in zero-shot and few-shot settings. Summarization can incorporate human-like structured reasoning into LLM generation, as shown by Wang et al. [59], who introduce an element-aware summarization method in which LLMs use a structured Chain-of-Thought approach to generate summaries. Although summary generation is automated, human experts design the evaluation framework and assess alignment between generated summaries and reference texts, bridging fully automated summarization and human-guided refinement.

Mozannar et al. [33] use LLMs to describe image or sentence clusters by contrasting in-cluster and out-of-cluster examples in a human-AI collaborative setting. While our work also involves human-AI collaboration for text summarization, it takes a different approach based on the interplay between the mapper workspace and mapper agents.

3.3 Mapper-Based Visualization of High-Dimensional Data

In this paper, we use mapper graphs [52] to explore the spaces of contextual word embeddings. This technique is widely applied across data science domains, including cancer research [29], gene expression analysis [23], and genomic profiling [9]; see [34] for an overview. Mapper graphs are used for graph skeletonization [45] and, in deep learning, to capture the topology of embeddings from image classifiers and LLMs [39, 41, 62], track layer evolution [36] and during fine-tuning [41], and characterize adversarial attacks [68]. TopoBERT [41] also applies mapper to word embeddings but relies on time-consuming and difficult-to-scale expert interpretation. *Explainable Mapper* extends this work by integrating LLM-assisted explanation generation, perturbation-based robustness assessment, and an interactive visual analytics workflow.

4 Mapper Taxonomy and Task Analysis

Boggust et al. [4] interviewed 13 expert users (data analysts, model developers, and computational linguists) and found that they often explore embeddings for model interpretation. Model-driven users compare embedding spaces to understand model behavior, while data-driven users inspect embeddings to uncover characteristics and relationships in the data. Both groups ultimately use embeddings to **generate and verify hypotheses** about specific subjects. We study the LLM embedding space topologically using mapper. Following Boggust et al. [4], we address model developers, NLP experts, and computational linguists. Our aim is to help them explore and interpret mapper graphs with *mapper agents* to gain deeper insight into the word embedding spaces.

To interpret mapper graphs, prior work has analyzed structural elements such as nodes and paths to generate and verify data-driven hypotheses. To better organize this exploratory space, we review ML studies using mapper graphs on embeddings from language models [39, 41], vision models [36, 39, 41, 67, 68], and local interpretations [54]. We summarize the common elements in Table 1 and define a taxonomy of five mapper elements to explain: nodes, components, paths, edges, and trajectories (illustrated in Figure 2(d)). The first three follow prior work; edges and trajectories are newly introduced. We exclude loops, since prior work [6] shows small cycles in mapper graphs can be artifacts that mislead users. Next, we describe user tasks for each element and their implications for interpreting contextual word embedding spaces.

Table 1: Mapper graph elements examined in prior work.

Map Elements	Node	Path	Loop	Component	Edge
TopoAct [39]	✓	✓	✓	✗	✗
Zhou et al. [68]	✓	✗	✗	✗	✗
TopoBERT [41]	✓	✓	✗	✓	✗
Purvine et al. [36]	✗	✓	✗	✗	✗
MOUNTAINEER [54]	✓	✓	✓	✗	✗
Mapper Interactive [67]	✓	✓	✗	✓	✗

T1. Investigating Node characteristics. A node represents a cluster of contextual word embeddings within the same topological neighborhood. Analyzing a node reveals common semantic or syntactic properties of the clustered tokens and their contexts, a common approach for studying shared characteristics of embedding neighborhoods [4, 39, 50].

T2. Investigating Path characteristics. A path between a source and target node is the shortest path connecting them in the mapper graph. Explaining a path shows how encoded linguistic properties change across connected clusters. For example, TopoAct [39] revealed a branch of water-related words shifting from a marine context (e.g., '*sea*') to a culinary one (e.g., olive '*oil*' in a pan).

T3. Investigating Component characteristics. A component in a mapper graph is a maximal set of nodes in which each pair of nodes is reachable from one another via edges. Disconnected components may correspond to distinct semantic topics within the contextual word embedding space that are more abstract than the node-level characteristics. For instance, prior work with TopoBERT [41] observed different components that capture distinct semantic information.

T4. Investigating Edge characteristics. An edge in a mapper graph connects two neighboring nodes and represents overlapping data points

between these two nodes. Analyzing the similarities and differences between neighboring nodes allows the users to understand how the encoded linguistic properties transition from one node to another. These differences may capture subtle linguistic variations, such as the use of specific words and their placement within a sentence (e.g., the token '*for*' used at the beginning vs. in the middle of a sentence).

T5. Investigating Trajectory characteristics. Understanding connections between any two nodes, even if not directly linked in the mapper graph, can reveal latent transitions or semantic pathways in the embedding space. We define a 1-token perturbation as an embedding produced by altering a single token in the sentence while keeping the focus token; this new embedding is called a *perturbed embedding*. A trajectory between an embedding in the source node and one in the target node is defined by a sequence of 1-token perturbations in embedding space and visualized by projecting it onto the nearest nodes in the mapper graph (see Section 6). Exploring such trajectories helps reveal how concepts gradually evolve through intermediate regions, and their projections onto the mapper graph trace exploratory routes among topological neighborhoods of the space.

The above list does not exhaust all mapper elements, but focuses on the most common ones. Other structures, such as star-shaped subgraphs explored by TopoAct [39], can also reveal meaningful insights. Notably, our framework is extensible to accommodate additional elements.

5 Mapper Agents

Interpreting a mapper graph over an embedding space is time-consuming and cognitively demanding, requiring users to inspect many words and sentences while hypothesizing and verifying various encoded linguistic properties. Computational assistance is therefore essential for revealing embedding properties. We identify three main **operations** to support the user tasks from Section 4: *summarization*, *comparison*, and *perturbation*. To apply these to mapper elements, we introduce two mapper agents: an **Explanation Agent** and a **Verification Agent**.

5.1 Operations

Summarization. This operation generates an understanding of a mapper element, such as the contextual embeddings of a node, edge, path, or component. For example, for a node *summarization* operation, this entails the *summarization* of common characteristics and linguistic properties of the node's embeddings.

Comparison. This operation identifies similarities and differences between two mapper elements. Unlike the *summarization* operation, which takes a single element (*uni-variate* input), *comparison* takes two elements of the same type (*bi-variate* input). Its output describes how the two elements are similar and how they differ. For example, comparing a pair of nodes yields similarities and differences between contextual embeddings in their respective neighborhoods.

Perturbation. The perturbation operation makes minor changes to a sentence while preserving the original token; the resulting token embedding is a *perturbed embedding*. Perturbation has two purposes. First, it validates results from summarization and comparison. For node summarization, each instance is perturbed, and the perturbed embeddings are checked to ensure they stay within the same node. The similarity between summaries of original and perturbed instances measures summary robustness. This validation also applies to edges, paths, and connected components. Second, perturbation enables exploration between nodes. A trajectory from a source node to a target node can be explored via a sequence of minor perturbations that gradually transform a source sentence into a target sentence (see Section 4).

5.2 Agent Types

Our goal is to help users generate and verify hypotheses about word embedding properties, i.e., to identify possible reasons for the mapper structure. We first *diverge* by generating many candidate explanations, then *converge* on those that are robust and likely to faithfully explain the embedding space. Our approach, shown in Figure 3, uses an encoder–decoder model with two agents: an *Explanation Agent* for hypothesis generation and a *Verification Agent* for hypothesis and explanation verification.

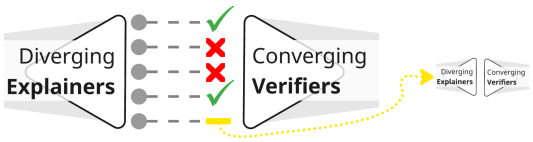

Fig. 3: We use the analogy of a decoder-encoder model, where the main purpose of explainers is divergence (creation of hypotheses) and of verifiers is convergence (confirmation of hypotheses). Hypotheses can also be refined, leading to new explanations and verifications.

5.2.1 Explanation Agents

The Explanation Agent class is a set of dedicated *explainers* that use *summarization*, *comparison*, and *perturbation* to explain properties of nodes, edges, paths, components, and trajectories (T1–T5) in a mapper graph. It supports hypothesis generation by producing diverse explanations, in particular, by identifying potential embedding properties that shape mapper element structures. The Explanation Agent can be implemented using precomputed features and rule-based methods, as demonstrated in prior work [48], which leverages semantic-role statistics and token frequencies. However, such approaches are inherently constrained by predefined rules and limited expressiveness [30], reducing their effectiveness for the open-ended interpretation of diverse mapper elements and analysis tasks.

To overcome these limitations, we use LLMs to generate natural-language explanation candidates, leveraging their semantic flexibility [33, 63] and extensive linguistic knowledge [57]. We implement explainers as task-specific prompts. For instance, to summarize a node, the LLM receives its instances (tokens and sentences) and identifies their common features. Full prompts are provided in the supplement.

The **Node Explainer** operates on individual nodes to support user task T1 in Section 4. The *summarization* operation analyzes shared linguistic properties of a node's data points to reveal its embedding neighborhood, while the *comparison* operation does the same for two nodes to highlight differences between their neighborhoods.

An **Edge Explainer** takes two adjacent, edge-connected nodes as input, where the edge represents shared word embeddings. It supports user task T2. The *summarization* operation examines shared and unique words to show how one node (embedding neighborhood) transitions into the other, while the *comparison* operation highlights differences in their transition patterns.

Unlike the Edge Explainer, the **Path Explainer** operates on two nodes that need not share overlapping embeddings. It takes the shortest path between two non-adjacent nodes as input and analyzes the data points in each node along that path. The *summarization* operation tracks how word usage patterns shift, develop, or stabilize along the path. The *comparison* operation contrasts these evolution patterns across different paths. Both outputs support task T3.

The **Component Explainer** operates on connected components to support user task T4. Its *summarization* operation analyzes points within a component to identify shared linguistic characteristics and higher-level patterns beyond individual nodes. Its *comparison* operation reveals similarities and differences between two components.

The **Trajectory Explainer** operates on any two mapper nodes to reveal their potential connections. It applies a *perturbation* operation to generate a sequence of small (e.g., 1-token) modifications that gradually transform a sentence from one mapper node into a sentence belonging to the other. We prompt an LLM to generate such trajectories following Choi et al. [10]. The trajectories are not unique: different perturbation orders can produce alternative trajectories between mapper nodes, each representing a plausible transition hypothesis. The trajectory is then visualized by linking the sequence of perturbed embeddings, each of which is assigned to its nearest mapper node or edge. See Section 6.2.4 for details and Figure 7 for an illustration.

5.2.2 Verification Agents

For each candidate explanation, we employ a Verification Agent to assess its robustness under perturbations. Although robustness does not guarantee the correctness or reliability of an explanation, it helps identify explanations that remain stable under small input variations,

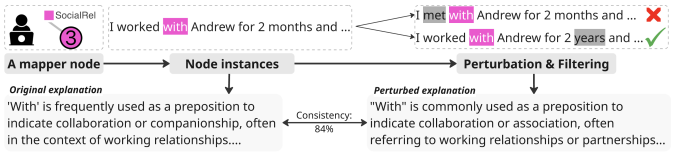

Fig. 4: Node explainer and verifier workflow. A user-selected mapper node contains three *SocialRel* instances. For each instance, context words are perturbed (gray) while preserving the focus word (pink). Perturbations that remain in the same node are retained. Explanations for the original and retained perturbed instances are compared using a consistency score based on sentence embedding cosine similarity.

providing evidence to support human interpretation. Since the Explanation Agent relies on LLM prompting, this verification step serves to mitigate the risk of hallucinations. Similar to the Explanation Agent, the Verification Agent comprises three operations—*summarization*, *comparison*, and *perturbation*—and the same tailored LLM prompts (given in the Supplement), but with different data inputs.

Node, Edge, Path, Component Verifier. A Verification Agent uses the *perturbation* operation to check outputs from Explanation Agents. The Verification Agent's input depends on the Explainer's scope (e.g., a node and its summary for the Node Verifier, an edge and its summary for the Edge Verifier).

We perturb each data point by changing one token while keeping the target token fixed. By default, an LLM generates five perturbations per data point using the supplemental prompt. We then choose a perturbed sentence whose perturbed-token embedding remains in the same node(s), measured by its average distance to all original token embeddings in that node. If this distance is below the node's average pairwise distance, we keep the perturbed sentence. Finally, we generate a new explanation for the perturbed examples using the same prompt as for the original Explainer.

We measure explanation robustness via similarity between original and perturbed explanations. We use cosine similarity over MiniLM sentence embeddings for their popularity and efficiency [58]. Our framework is metric-agnostic, allowing this measure to be readily replaced with alternative text-level semantic similarity metrics [8], such as embedding-based scores (e.g., BERTScore [65]) or methods that emphasize local differences [51]. The resulting score provides a coarse measure of explanation stability and is intended to support, rather than replace, human judgment of explanation faithfulness. Figure 4 illustrates the node explainer and verifier workflow. For a Node Verifier using *summarization*, the agent receives a node and its summary. We perturb the node's sentences (1-token substitutions and rephrasing), keep those that remain in the same neighborhood, and generate a new summary. The similarity between the original and perturbed summaries measures explanation robustness.

Trajectory Verifier. Because the Trajectory Explainer relies on perturbations to connect two mapper nodes, its robustness cannot be readily verified automatically. Instead, verification relies on user inspection of the intermediate sentences and their mapper-node assignments, supported by visualizations that highlight sentence-level changes and link them to points along the trajectory (see Section 6.2.4).

6 Mapper Workspace

To help users generate and verify hypotheses about embedding properties with the agents from Section 5, we present *Explainable Mapper*.

6.1 Design Requirements

Explainable Mapper's requirements build on three prior systems, i.e., Embedding Comparator [4], TopoBERT [41], and Mapper Interactive [67], which derived design goals from expert interviews and systematic literature reviews on embedding spaces and mapper graphs. From these studies, we distill six requirements for *Explainable Mapper* workspace, combining established goals with agent-specific needs.

R1. Visualize and compute the mapper graph in a flexible way. For effective exploration, *Mapper Interactive* offers dynamic layouts and customizable visual encodings (e.g., node size, color, glyph) and lets users adjust mapper parameters to interactively recompute the graph.

R2. Display a projection plot as an overview reference. *Embedding Comparator* recommends projection views because they are familiar to most expert users, whereas *TopoBERT* notes that experts are typically less familiar with mapper graphs. Therefore, a projection view should be displayed to guide users in interpreting mapper graphs.

R3. Facilitate rapid identification of regions of interest. Given the vast exploration space, *Embedding Comparator* surfaces meaningful patterns. *TopoBERT* supports region identification through graph structure, node glyphs, and keyword search. We extend this with projection views to guide mapper inspection and annotate graphs with *mapper agents*' explanations to help users quickly find elements.

R4. Enable interactive selection and comparison of elements. *Mapper Interactive* supports selecting mapper elements of nodes, paths, and components. To better support user tasks (Section 4), this functionality should be extended to cover all elements in the mapper taxonomy and enable element-to-element comparisons.

R5. Support iterative explanation and verification of mapper elements. The system should display explanations and verification information for mapper elements generated by *mapper agents*, helping users interpret these elements. In addition, users should be able to use *mapper agents* to iteratively generate and verify new explanations.

R6. Present metadata for selected mapper elements. *TopoBERT* shows that experts require detailed metadata of embeddings within selected mapper elements to aid interpretation, such as word label distributions and highlighted words in their original sentences. Our tool should also show perturbation instances generated by *mapper agents*.

6.2 Workspace Interface

As shown in Figure 5, the *Explainable Mapper* workspace supports the tasks in Section 4 and meets the requirements in Section 6.1. It enables iterative embedding exploration by combining multiple agents to generate insights. Users begin with an embedding overview via the Mapper Graph (**R1**) and Projection views (**R2**), then identify mapper elements of interest using cues such as precomputed annotations (**R3**), select elements (**R4**), create explanations and verify their robustness (**T1-T5**, **R5**), and inspect detailed metadata (**R6**).

Data Preprocessing. The data preprocessing steps involve extracting token embeddings layer-by-layer from a (encoder or autoregressive) language model on a sufficiently large text corpus (e.g., 12 layers of a BERT-base model) and associating each token with various linguistic labels, such as its semantic role and part-of-speech (POS) tag.

6.2.1 Control Panel and Settings Panel

In the *control panel*, users set the parameters for the mapper algorithm introduced in Section 2. They can create either the *classical mapper* or the *ball mapper*. For the classical mapper, users specify DBSCAN parameters (*minPts* and ε) and mapper parameters (*cover number* and *cover overlap*). For the ball mapper, only ε is required. By default, we compute the mapper graph using the parameters from Section 2 on embeddings from a single layer at a time.

In the *settings panel*, users specify visual encodings for the mapper graph (e.g., which attribute controls node color/size or edge width; see Section 6.2.2). For the default mapper, we precompute explanations, representative keywords, and consistency scores for each component and node. Explanations for other mapper operations are generated on demand during user interaction (see the Supplement). Users can select these keywords and scores as visual annotations in this panel, and filter specific tokens for further exploration.

6.2.2 Mapper Graph

The mapper visualization follows *Mapper Interactive* [67], a toolbox for exploring mapper graphs. Nodes represent embedding clusters, and edges indicate overlaps in their underlying data points. Nodes are rendered either as circles (encoding clusters based on the L_2 norm) or as pie charts, where slices reflect linguistic properties (e.g., semantic roles). Node size can scale with the number of contained points or remain fixed, while edge width can encode the Jaccard similarity or be shown uniformly. The default layout is force-directed, but we also provide an anchored layout that positions nodes at the centroids of their

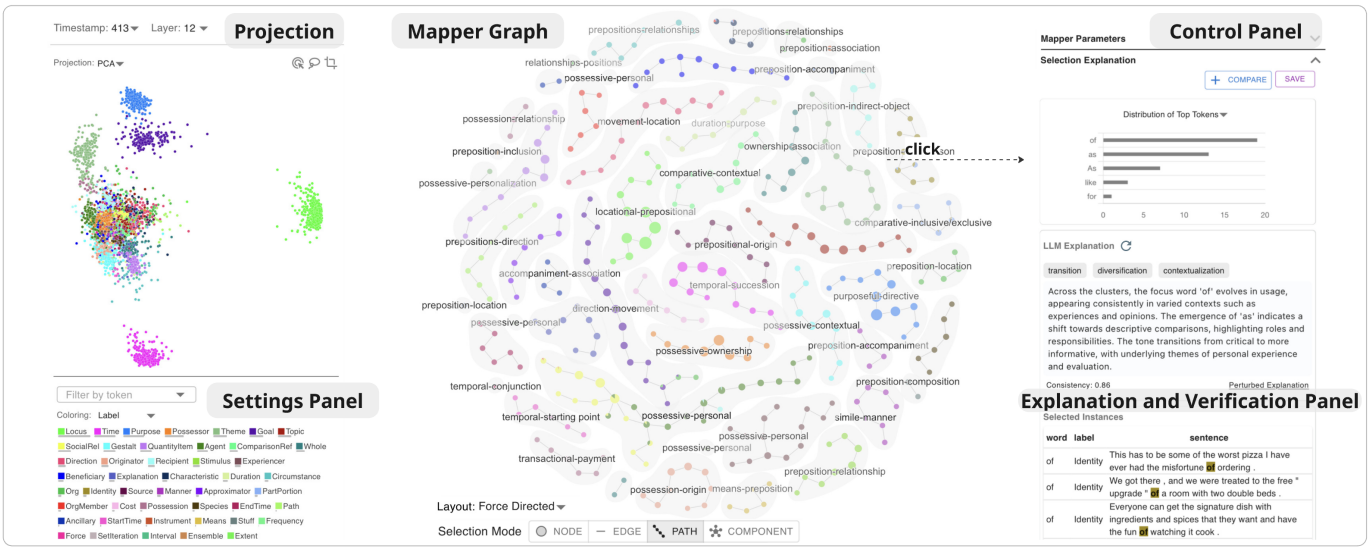

Fig. 5: The *Explainable Mapper* workspace consists of two main visualizations, i.e., the *Mapper Graph* and *Projection* visualized as a scatter plot. Properties associated with the mapper graph and its underlying embeddings are displayed in the *Settings Panel*. The user can specify mapper parameters in the *Control Panel*. To explore the embedding properties, the users can select the mapper elements for explanation and verification.

points in a 2D projection (e.g., PCA). These layouts support different analysis goals: the force-directed view highlights topological structure but ignores geometric positioning, whereas the anchored view preserves relative positions but may introduce overlap in dense regions.

Interaction Methods. Similar to *Mapper Interactive*, the workspace supports zooming, dragging, and panning. Users can enable keyword-based annotations in the settings panel. Precomputed explanation keywords are attached to mapper nodes or components, with opacity reflecting verification scores (higher scores yield higher opacity). To reduce overlap, higher-scoring keywords are prioritized while overlapping ones are hidden. When zoomed in, annotations update to match the visible nodes, revealing additional keywords with increasing detail.

The selection panel beneath the mapper graph lets users select mapper elements (nodes, edges, paths, components). Users can also select elements directly in the graph view. Once selected, the structures are highlighted and node size is shown numerically. Users can apply an Explainer to summarize an element or compare explanations for two elements of the same type. They can also generate a trajectory explanation by selecting a sentence from the source and target nodes and specifying the number of 1-token perturbations connecting them.

6.2.3 Projection

We complement the mapper graph with a scatterplot of embeddings projected using a user-specified method (e.g., PCA, MDS, t-SNE, UMAP). Embeddings appear as dots colored by a selected linguistic property (semantic role, POS tag, or L_2-norm). Selecting clusters in the projection reveals their locations in the mapper graph and supports anchored graph layout. Users can select points via clicking, lassoing, or brushing. Linked views ensure that selections in either view highlight the corresponding regions in the other view.

6.2.4 Explanation and Verification Panel

When the user selects one or two mapper elements from the selection panel, a bar chart shows token or label frequencies in those elements, and the tokens with their sentences appear in a table in the *explanation and verification panel*. After an explanation is generated, it is automatically verified by the mapper agents introduced in Section 5.

Explanation Design. Each explanation consists of a natural language description and three descriptive keywords that summarize the selected element(s). The user can switch the Explainer from summarization to comparison by clicking the *compare* button on the panel and selecting a second element of the same type in the mapper graph as a reference. After the second element is selected, the metadata for both elements is updated in the bar chart and table, and a new comparison explanation is computed and shown in that panel.

Verification Design. An explanation's robustness to perturbations is given by a consistency score shown beneath it, computed as the cosine similarity between sentence embeddings of the original and perturbed explanations. Clicking the *perturbed explanation* button reveals the explanation generated for the perturbed examples, which are listed in the table below for inspection. Clicking the *refresh* button in this panel generates new hypothetical explanations for the selected element or comparison. For each new explanation, a perturbed explanation and updated consistency score are automatically computed. This lets users iteratively select explanations with high robustness scores as candidates for further inspection.

When a Trajectory Explainer is applied, the system projects the trajectory onto the nearest mapper graph nodes or edges. For each perturbed example, it computes the embedding, identifies the nearest data point within the ε threshold and the same L_2-norm interval (see Section 2), and assigns the example to the corresponding node or edge. The trajectory then plots these assignments in sequence and connecting them. Selecting a perturbed example highlights its corresponding trajectory assignment in red. To facilitate inspection, an LLM also identifies and highlights the key changes between consecutive perturbations (see Figure 7). Users can further refine the trajectory by inserting or deleting perturbed examples or by regenerating an alternative trajectory; see the supplement for additional use cases.

7 Case Studies

We evaluate our methodology by examining how the Explanation and Verification Agents in the *Explainable Mapper* workspace support generating robust insights into embedding properties. Through case studies, we show that our framework recovers prior findings on how embedding properties are encoded across BERT-based layers. Beyond replicating known results, *Explainable Mapper* provides more nuanced, topology-driven insights, revealing structural patterns via mapper elements and suggesting new interpretive directions.

7.1 Prior Insights on BERT's Embeddings

Prior work has shown that BERT's early layers encode surface features such as lexical similarity [50], sentence length, and word presence [22]. Early to middle layers capture syntactic properties, particularly POS tags [26, 55]. Semantic information is encoded throughout the model [55], while later layers are more task-specific [26]. Whereas prior work mostly uses NLP-specific probes and geometric analyses, mapper offers a topological view of the embedding space. For instance, TopoBERT [41] revealed patterns in a fine-tuned BERT, such as shifts from fronted to non-fronted token usage along a path and clustering of

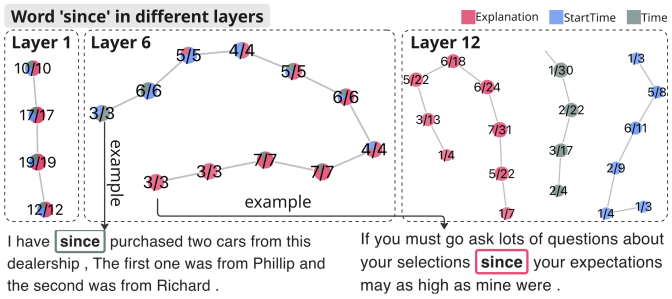

Fig. 6: The word *'since'* across layers 1, 6, and 12. In layer 1, temporal and causal meanings are mixed across nodes; in middle layers, the mapper path shifts from causal to temporal meaning; in layer 12, the classes form separate components.

identical tokens despite differing semantics. We show that our agent-based approach not only recovers established findings but also uncovers new topological insights. We analyze the embedding spaces of a fine-tuned BERT, BERT-base, and the state-of-the-art but under-interpreted ModernBERT [61]. Additional cases appear in the Supplement.

7.2 Analysis of Fine-Tuned BERT

Following TopoBERT [41], we fine-tune BERT-base on the *preposition supersense disambiguation* task, which assigns semantic categories to prepositions. We use the *supersense Role* labels from STREUSLE v4.2 [46], containing 41 labels over 4,282 preposition tokens. The model is fine-tuned for 7 epochs, then we extract 768-dimensional embeddings from each of its 12 layers. We apply mapper to each layer's embeddings with the same parameters as TopoBERT. GPT-4o [21] is the underlying model for the agents.

Contextualization Process of *'since'*. Prior work shows that BERT word representations become more contextualized in later layers. We use the word *'since'*, which can be temporal or causal, to probe this process and track how its usage changes across layers. Searching for this token, we find that in layer 1, *'since'* instances with different labels are mixed (Figure 6). They then follow a path that shifts from the "Explanation" label to temporal labels by layer 6. By layer 12, they separate into three distinct components and mix with other words that share their respective labels. We examine the path in layer 6 using the Path Explainer, which yields an explanation with a verification score of 0.8. It shows that *'since'* is first used temporally, marking events after a point in time, then shifts toward causal and conditional uses. The path reveals intermediate stages of contextualization, showing how *'since'* gradually separates its temporal and causal meanings across layers.

Trajectory Analysis of *'about'*. As shown in Figure 7, the word *'about'* appears with three supersense roles: *Approximator*, *Topic*, and *Stimulus* (i.e., the emotional or cognitive trigger of an experience), and each role forms a component. To investigate transitions among them, we select nodes A and B and use *'about'* sentences as the source and target. The Trajectory Explainer generates 13 intermediate sentences through minimal edits, with key changes identified by LLMs highlighted in yellow. We embed *'about'* in each sentence, assign it to the nearest mapper node or edge, and connect the assignments into a trajectory. The trajectory traverses all three components: *'about'* first shifts from an *Approximator* in *"20 minutes"* to the *Topic* of *"the car repair"* between sentences 1 and 2, and then transitions from *Topic* to *Stimulus* between sentences 6 and 7 as *"some thoughts"* become *"doubts"*. This example illustrates how trajectory analysis reveals semantically meaningful transitions among different supersense roles.

Possessive Pronouns. Middle BERT layers (3–6) encode syntactic token function while remaining semantically contextualized [43]. To probe this, we analyze possessive pronouns (e.g., '*my*', '*her*') in layer 4. We compare a mapper component with its corresponding points in PCA and UMAP projections in Figure 8. In the PCA plot, the words *'my'*, *'your'*, and *'our'* are mixed in the left region of the projection, while *'her'*, *'his'*, and *'their'* are mixed on the right. In the UMAP plot, these six words are instead split into five largely disjoint clusters. The mapper graph, however, represents them as a single connected component that gradually refines into distinct nodes for each word while retaining their

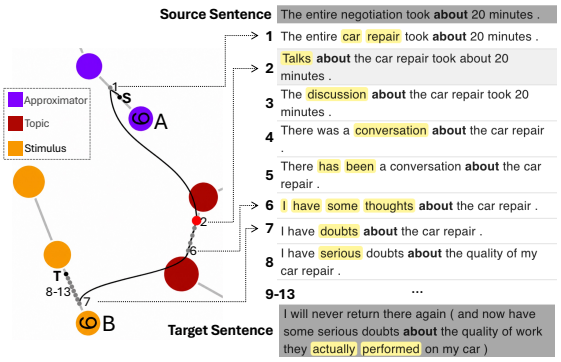

Fig. 7: Trajectory of *'about'* from node A to node B in layer 12 of the fine-tuned BERT, with 13 LLM-generated intermediate sentences. Key changes between consecutive sentences are highlighted in yellow, and the focus word is shown in bold. Sentence 2 is selected in the table (gray) and its corresponding position on the trajectory is highlighted in red.

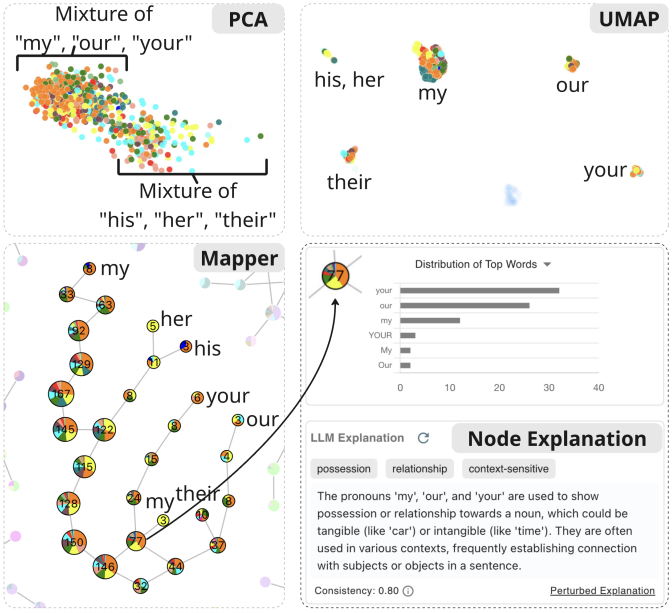

Fig. 8: Mapper component with its corresponding points in PCA and UMAP. The right panels show the metadata and explanation of a node at the separation of the words *my*, *our*, and *your*.

internal connectivity. For example, we highlight a node at the branching point where *'my'*, *'your'*, and *'our'* begin to separate; the explanation generated by the mapper agents correctly captures their shared role as possessive pronouns expressing relationships to a noun.

7.3 Use Cases: BERT-Base vs. ModernBERT

We use the Groningen Meaning Bank (GMB) dataset [5] with BERT-base and ModernBERT for embedding extraction. GMB provides deeply annotated English texts with many polysemous words in varied contexts, well suited to our use case. Guided by user feedback, we select the 'CIA World Factbook' subset, remove stop words, and obtain 27,504 tokens across 2,252 sentences. We extract 768-dimensional word embeddings from BERT-base's 12 layers and ModernBERT's 22 layers to compare their embedding spaces and better understand ModernBERT. We apply mapper graphs to all layers of both models. To standardize resolution, we use a density-based cover that divides each point cloud into 50 equally sized intervals with 30% overlap. For DBSCAN, we set minPts to 5 (as in TopoAct [39]) and choose ϵ as the 50th percentile of the fifth nearest-neighbor distance distribution.

7.3.1 Contextualization in Final Layers

In the final layer, BERT-base forms several medium-sized components, while ModernBERT yields one large component plus a few medium ones. Using the Component Explainer, we analyzed these structures and obtained explanations with high consistency scores.

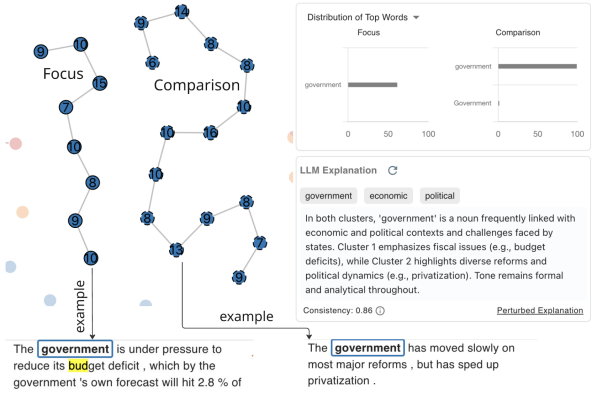

Fig. 9: Two components formed by the word 'government' at the final layer of ModernBERT and the comparison explanation.

A major BERT-base component contains the words 'economic', 'economy', and 'GDP'. The explanation indicates that these terms are *mainly used to describe countries' financial status*. In contrast, ModernBERT's large component groups 'economy' and 'GDP' with broader terms such as 'tourism' and 'agriculture'. Its explanation shows that these words occur in varied contexts describing the economy in relation to geopolitical entities and social structures. This contrast suggests that ModernBERT may capture more integrative, cross-domain relations, whereas BERT-base tends to preserve fine-grained semantic distinctions, consistent with the view that ModernBERT has stronger contextualization capability [61]. In the last layer, both models split 'government' into components reflecting fiscal vs. political contexts. As shown in Figure 9, ModernBERT forms two 'government' components, one tied to *fiscal issues* and the other to *reform and political dynamics*. ModernBERT also reveals unique components, such as one about *the world's five oceans*. These results show that *Explainable Mapper* captures both common and rare contextual distinctions.

7.3.2 Contextualization Across Layers

Across layers in both models, words sharing a lemma (e.g., 'grow', 'grown', 'grew', 'growth') or the same word with different POS tags are separated in early layers, then gradually merge in middle and later layers, with some eventually integrating into broader contextual clusters. For instance, 'world' can be a common noun or a proper noun (e.g., 'World War'). In layer 1, these usages are clearly separated in both models. By layer 22 in ModernBERT, they form a path marking the shift from noun to proper noun (see Figure 10). Path explanation shows this path moving from global metrics to World Wars and the World Bank. In ModernBERT's final layer, the proper-noun 'world' becomes an outlier, indicating more dispersed, contextualized representations. In contrast, in BERT's final layer, it merges with words like 'war' and special tokens (e.g., 'II'), forming a World War I/II context.

8 Expert Evaluation

We conducted an expert study to evaluate *Explainable Mapper*. The goal was to assess the effectiveness of the workspace interfaces (Section 6), mapper taxonomy tasks (Section 4), and mapper agents (Section 5) in supporting experts' exploration of mapper graphs and generation of stable insights into language model embeddings.

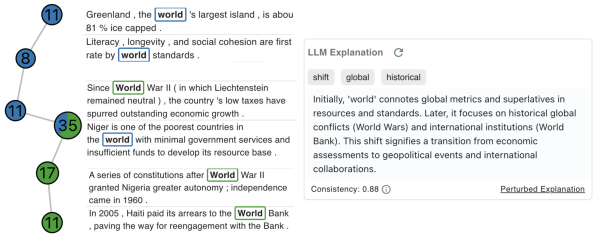

Fig. 10: A path of 'world' at layer 22 of ModernBERT that transits from noun (blue) to proper noun (green), and the path explanation.

8.1 Session Setup

Data and Pilot Study. For the study, we used embedding spaces from the GMB dataset with the BERT-base model (Section 7.3). After a three-user pilot, we refined the setup by removing stop words and tightening the DBSCAN ϵ parameter to reduce overload and mapper complexity. Because trajectory explanations were hard to use under time pressure, the expert study used them only as illustrative early-feedback examples.
Participants. We recruited six experts (E1–E6): one master's student, three PhD students, and two postdocs in computational linguistics, explainable AI, NLP, and ML. All were comfortable with contextualized embeddings and routinely used projections (e.g. PCA, UMAP) and nearest-neighbor analyses; none had used mapper before.
Procedure. We ran 90-minute, in-person sessions. After consent and a background questionnaire, participants got a short tutorial on mapper graphs and *Explainable Mapper*, then 10 minutes of free exploration and 20 minutes of think-aloud tasks interpreting nodes, paths, components, and edges (at least one example of each). We then showed a trajectory explanation—either between nodes in different components or within one component (see the Supplement). Each session closed with a brief interview, a post-questionnaire, and full recording.

8.2 Expert Feedback

We summarize expert feedback into four themes: workspace design, task usability, explanation and verification agents, and expert–agent collaboration. We examined 26 cases using the agents to interpret mapper elements; the questionnaire ratings are in the Supplement.

8.2.1 Workspace Design

All users found the system easy and intuitive. For example, E5 called it "*a really nice tool, the selection and everything was pretty intuitive*," and E3 noted that "*the beautiful interface really invites me to explore the data*." Opinions on the graph layout were mixed. The force-directed layout was favored for helping users "*immediately see some structures*" (E5) and being "*relatively easy to browse*" (E2). In contrast, E6 preferred the anchored layout for its closer relation to the projection views, and E1 and E4 used it to select nearby components for comparison. Users differed in how they identified parts of interest. Of the 26 cases, 13 were guided by graph encodings (e.g., node and edge glyphs, graph structure), 6 by the projection view, 5 by token queries, and 2 by annotations. Although annotations were appreciated by E3, E5, and E6—for example, E5 remarked, "*I like the component annotation; you get a really nice overview*"—they were underused overall, and E3 noted, "*most (annotation) keywords are identical and can be improved*."

8.2.2 Task Usability

Tasks based on nodes, edges, paths, and components were sufficient. E6 stated, "*All elements are useful*," and E3 noted, "*These tasks allow me to see all structures I want to explore*." Nodes and components were used most often; as E5 explained, "*Usually you're interested in either one instance, like one cluster, or you're interested in the whole disconnected component and see what's similar*." Edges were preferred by E2 for "*direct comparison*," while E5 felt edge explanations are useful only when "*there's an abrupt change between one node and the other*." Paths were also favored; E3 said, "*I'm really interested in how the topic changes along a path*." E1 suggested component comparison may be unnecessary when differences are already clear.

For trajectories, E5 found gradual changes helpful for explaining paths, making them easier to interpret. E3 wanted to use them to connect two components sharing a word. For visual design, E2 recommended reducing complexity by showing only intermediate sentences with substantial changes, such as when moving between mapper nodes.

8.2.3 Explanation and Verification Agents

Users appreciated the explanation agents as a fast way to summarize mapper elements. E2 noted it was "*way quicker to get the summary... I can use this list to draw my own conclusions, but this took quite a time*," and E4 described it as "*more efficient, especially if there were a lot of sentences*." E5 found it especially helpful for paths, where gradual

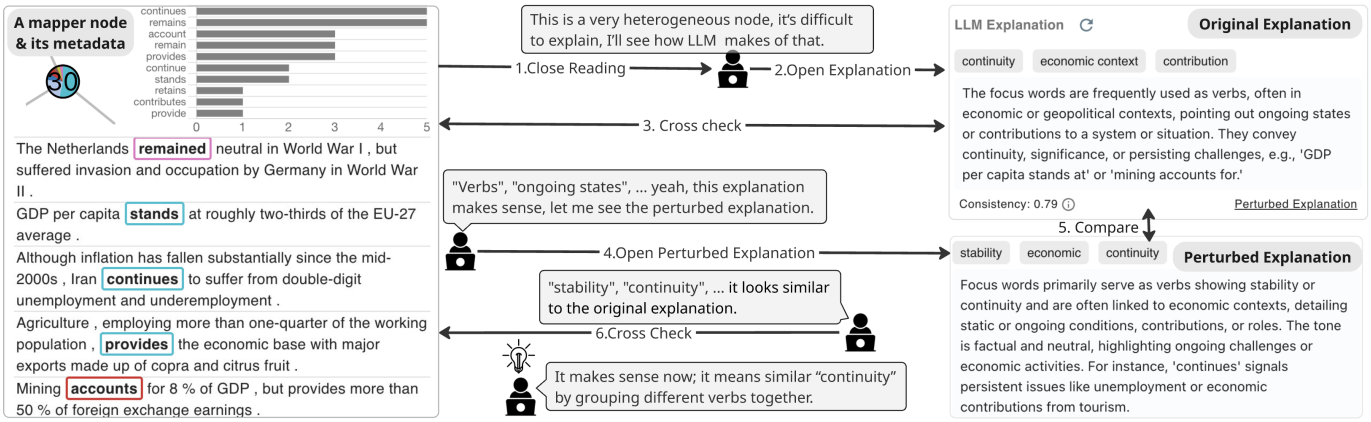

Fig. 11: A case where the agent guides E2 in interpreting a mapper node. Left: the mapper node and samples of its metadata. Middle: E2's interactions with the agent while interpreting the node. Right: node explanations generated by the agent.

changes across many sentences are hard to track manually. All users valued the keywords for a quick overview of the explanation.

For verification, users liked the consistency score between original and perturbed explanations as a quick indicator of stability. E3 found it "*super helpful; I will not read the explanation if the score is very low.*" E2, however, noted that the score is still a coarse signal and suggested highlighting disagreement regions between original and perturbed explanations. E6 treated the score as a quick sanity check and stressed the value of a reference point for "*what good looks like*."

E2 noted that explanations were "*mostly accurate but a bit generic*," and E5 found them sometimes wordy. E3 suggested using a table to summarize key points. E1 and E2 recommended highlighting perturbation locations in sentences and allowing users to interactively perturb text. E5 and E6 observed that the consistency score may not fully capture semantic differences but is useful as a "*quick reference point*," and they also recommended displaying the LLM prompts to users.

8.2.4 Expert–Agent Collaboration Analysis

We distilled the 26 cases into recurring interaction patterns between experts and the agent. After selecting a mapper element, experts either read its metadata carefully (e.g., token distributions, sentences), yielding a *concrete hypothesis* or *no hypothesis*, or skimmed it, yielding a *general hypothesis* or *no hypothesis*. Across the 26 cases, 7 involved concrete hypotheses, 7 involved general hypotheses, and 13 involved no initial hypothesis. Experts then opened the agent's explanations (e.g., original and perturbed explanations, consistency scores) and cross-checked them against the metadata, comparing them with any prior concrete hypothesis. This process was iterative: experts regenerated explanations when they wanted alternative interpretations or judged the quality (via consistency scores or their own assessment) to be low. They ultimately confirmed, rejected, or remained uncertain about each mapper element's hypothetical interpretation.

These workflows support expert interpretation differently, depending on experts' initial hypotheses. Figure 11 shows a case where an expert had *no hypothesis*. Expert E2 tried to interpret a node from its metadata but found it "*difficult to interpret*" due to heterogeneous words and types (e.g., continue, remain, account). With *no initial hypothesis*, he asked the agent for an explanation. The agent suggested the words represented "*verbs in an economic context*" and "*ongoing states.*" After checking the metadata, E2 found this plausible and confirmed that the perturbed explanation showed a similar pattern. He concluded: "*This node represents continuity by grouping different verbs.*" Additional workflow examples are provided in the Supplement.

9 Discussion and Limitations

LLM Hallucinations. LLMs are prone to hallucinations [19], and our Verification Agent mitigates this issue by evaluating explanation robustness through perturbations [15], although its effectiveness is ultimately constrained by the underlying models. Robustness indicates stability rather than correctness, and participant E2 observed that some explanations were overly general or internally inconsistent. Future work could further reduce hallucinations using explanation ensembles that cross-check outputs across models or prompts to flag unreliable explanations and estimate confidence, or by incorporating human feedback to refine the Explanation Agent.

Explanation Verification. Our perturbation strategy makes small sentence changes that minimally affect the focus word's embedding, assuming explanations should remain stable under such perturbations. Future work includes studying cases where minor edits cause large embedding shifts to probe explanation vulnerabilities. We emphasize that the Verification step is intended to assess robustness under perturbations, not the correctness or trustworthiness of an explanation. Experts must still validate LLM explanations against metadata and their own hypotheses. The consistency score serves as a coarse stability signal and a quick sanity check: low scores motivate explanation regeneration, while high scores suggest explanations for further inspection. However, high consistency does not guarantee faithfulness because of metric limitations (see the Supplement). Future work could incorporate additional quality measures, such as LLM-as-a-Judge [25]. Future work could also highlight local differences, add reference points, and link explanation segments to supporting data.

The Trajectory Explainer remains exploratory. Despite prompt and user-edit safeguards, trajectories may still contain inconsistencies or unintended semantic shifts. Future work will explore ensemble-based generation and filtering and evaluate its utility through expert studies.

Workspace Design. Experts used the projection view for navigation and preferred the anchored layout for alignment, suggesting that both should be used together. Because clutter in the anchored layout often led users to the force-directed layout, future designs could combine both through positional forces. Mapper parameters control graph granularity, but fine-grained analyses often produce large graphs that remain difficult to explore. Multi-scale analysis with on-demand subgraph expansion is a promising direction.

Learnability is essential for adoption. Some experts struggled with the force-directed layout or forgot comparison workflows. Future work includes improved onboarding, guided tutorials, and better graph layouts and annotations for users new to mapper or visual analytics.

Evaluation. We evaluate usability, insight generation, and support for expert reasoning. To enable quantitative evaluation, future work should establish benchmarks for topological interpretability and systematic comparison methods, particularly for evaluating agreement between expert- and LLM-generated explanations when multiple valid interpretations exist. Time constraints and participants' limited familiarity with mapper graphs limited each expert to a few examples, restricting exploration of the embedding space. Longer studies with more participants would provide more comprehensive feedback.

Decoder-only Models. We focus on encoder-based language models because they provide bidirectional contextualized token embeddings. Extending *Explainable Mapper* to decoder-only models (e.g., Llama and GPT) is left for future work, as their causal attention mechanism requires adapted perturbation strategies and prompt designs.

Acknowledgments

This work was partially supported by the U.S. National Science Foundation under Grants IIS-2145499, IIS-2205418, and DMS-2134223, and by the Swiss National Science Foundation under Project 10003068. We used Claude Sonnet 5 via Cursor for limited coding assistance, including debugging, minor code refactoring, and refining prompts for the explanation and verification agents.

References

[1] K. Amara, R. Sevastjanova, and M. El-Assady. SyntaxShap: Syntax-aware explainability method for text generation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 4551–4566. Association for Computational Linguistics, Bangkok, Thailand, Aug. 2024. doi: 10.18653/v1/2024.findings-acl.270 3

[2] S. Amershi, D. Weld, M. Vorvoreanu, A. Fourney, B. Nushi, P. Collisson et al. Guidelines for human-ai interaction. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, pp. 1–13. Association for Computing Machinery, New York, NY, USA, 2019. doi: 10.1145/3290605.3300233 3

[3] M. Berger. Visually analyzing contextualized embeddings. In *IEEE Visualization Conference (VIS)*, pp. 276–280, 2020. doi: 10.1109/VIS47514.2020.00062 3

[4] A. Boggust, B. Carter, and A. Satyanarayan. Embedding Comparator: Visualizing Differences in Global Structure and Local Neighborhoods via Small Multiples. In *27th Int. Conf. on Intelligent User Interfaces*, pp. 746–766, 2022. doi: 10.1145/3490099.3511122 1, 3, 5

[5] J. Bos, V. Basile, K. Evang, N. J. Venhuizen, and J. Bjerva. *The Groningen Meaning Bank*, pp. 463–496. Springer Netherlands, Dordrecht, 2017. doi: 10.1007/978-94-024-0881-2_18 7

[6] M. Carriere, B. Michel, and S. Oudot. Statistical analysis and parameter selection for mapper. *Journal of Machine Learning Research*, 19(12):1–39, 2018. doi: 10.48550/arXiv.1706.00204 2, 3

[7] N. Chalapathi, Y. Zhou, and B. Wang. Adaptive covers for mapper graphs using information criteria. In *IEEE International Conference on Big Data (Big Data)*, 2021. doi: 10.1109/BigData52589.2021.9671324 2

[8] D. Chandrasekaran and V. Mago. Evolution of semantic similarity—a survey. *ACM Computing Surveys (Csur)*, 54(2), art. no. 41, 37 pp., Feb. 2021. doi: 10.1145/3440755 5

[9] H. J. Cho, J. Zhao, S. W. Jung, E. Ladewig, D.-S. Kong, Y.-L. Suh et al. Distinct genomic profile and specific targeted drug responses in adult cerebellar glioblastoma. *Neuro-Oncology*, 21(1):47–58, 2019. doi: 10.1093/neuonc/noy123 3

[10] S. Choi, G. Yoo, and J.-Y. Lee. BridG MT: Enhancing llms' machine translation capabilities with sentence bridging and gradual mt. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 26018–26042, 2025. doi: 10.48550/arXiv.2410.11693 4

[11] P. Dlotko. Ball mapper: a shape summary for topological data analysis. arXiv preprint arXiv:1901.07410, 2019. doi: 10.48550/arXiv.1901.07410 2

[12] M. El-Assady and C. Moruzzi. Which biases and reasoning pitfalls do explanations trigger? decomposing communication processes in human–ai interaction. *IEEE Computer Graphics and Applications*, 42(6):11–23, 2022. doi: 10.1109/MCG.2022.3200328 3

[13] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. 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, 1996. doi: doi/10.5555/3001460.3001507 2

[14] S. Garg and G. Ramakrishnan. BAE: Bert-based adversarial examples for text classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6174–6181, 2020. doi: 10.18653/v1/2020.emnlp-main.498 3

[15] A. Hedström, L. Weber, D. Krakowczyk, D. Bareeva, F. Motzkus, W. Samek et al. Quantus: An explainable ai toolkit for responsible evaluation of neural network explanations and beyond. *Journal of Machine Learning Research*, 24(34):1–11, 2023. doi: 10.48550/arXiv.2202.06861 9

[16] F. Heimerl and M. Gleicher. Interactive analysis of word vector embeddings. In *Computer Graphics Forum*, vol. 37, pp. 253–265. Wiley Online Library, 2018. doi: 10.1111/cgf.13417 3

[17] S. Holter and M. El-Assady. Deconstructing human-ai collaboration: Agency, interaction, and adaptation. In *Computer graphics forum*, vol. 43, p. e15107. Wiley Online Library, 2024. doi: 10.1111/cgf.15107 3

[18] C.-Y. Hsieh, C.-K. Yeh, X. Liu, P. Ravikumar, S. Kim, S. Kumar et al. Evaluations and methods for explanation through robustness analysis. In *International Conference on Learning Representation (ICLR)*, 2021. doi: 10.48550/arXiv.2006.00442 3

[19] L. Huang, W. Yu, W. Ma, W. Zhong, Z. Feng, H. Wang et al. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2), art. no. 42, 55 pp., Jan. 2025. doi: 10.1145/3703155 9

[20] Z. Huang, D. Witschard, K. Kucher, and A. Kerren. VA + Embeddings STAR: A State-of-the-Art Report on the Use of Embeddings in Visual Analytics. *Computer Graphics Forum*, 2023. doi: 10.1111/cgf.14859 1, 3

[21] A. Hurst, A. Lerer, A. P. Goucher, A. Perelman, A. Ramesh, A. Clark et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024. doi: 10.48550/arXiv.2410.21276 7

[22] G. Jawahar, B. Sagot, and D. Seddah. What does BERT learn about the structure of language? In *Proc. of the Annual Meeting of the Association for Computational Linguistics*, pp. 3651–3657. ACL, Florence, Italy, July 2019. doi: 10.18653/v1/P19-1356 6

[23] R. Jeitziner, M. Carriére, J. Rougemont, S. Oudot, K. Hess, and C. Brisken. Two-Tier Mapper, an unbiased topology-based clustering method for enhanced global gene expression analysis. *Bioinformatics*, 35(18):3339–3347, 2019. doi: 10.1093/bioinformatics/btz052 3

[24] D. Jin, Z. Jin, J. T. Zhou, and P. Szolovits. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, pp. 8018–8025, 2020. doi: 10.1609/aaai.v34i05.6311 3

[25] D. Li, B. Jiang, L. Huang, A. Beigi, C. Zhao, Z. Tan et al. From generation to judgment: Opportunities and challenges of llm-as-a-judge. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 2757–2791, 2025. doi: 10.18653/v1/2025.emnlp-main.138 9

[26] N. F. Liu, M. Gardner, Y. Belinkov, M. E. Peters, and N. A. Smith. 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)*, pp. 1073–1094. Association for Computational Linguistics, Minneapolis, Minnesota, 2019. doi: 10.18653/v1/N19-1112 6

[27] S. Liu, P.-T. Bremer, J. J. Thiagarajan, V. Srikumar, B. Wang, Y. Livnat et al. Visual exploration of semantic relationships in neural word embeddings. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):553–562, 2017. doi: 10.1109/TVCG.2017.2745141 3

[28] S. Marjanovic, I. Augenstein, and C. Lioma. Investigating the impact of model instability on explanations and uncertainty. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 11854–11879. Association for Computational Linguistics, Bangkok, Thailand, Aug. 2024. doi: 10.18653/v1/2024.findings-acl.705 3

[29] J. C. Mathews, S. Nadeem, A. J. Levine, M. Pouryahya, J. O. Deasy, and A. Tannenbaum. Robust and interpretable PAM50 reclassification exhibits survival advantage for myoepithelial and immune phenotypes. *NPJ Breast Cancer*, 5(30), 2019. doi: 10.1038/s41523-019-0124-8 3

[30] R. R. Menon and S. Srivastava. DISCERN: Decoding systematic errors in natural language for text classifiers. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 19565–19583, 2024. doi: 10.18653/v1/2024.emnlp-main.1091 4

[31] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pp. 3111–3119, 2013. doi: 10.48550/arXiv.1310.4546 3

[32] M. Moradi and M. Samwald. Evaluating the robustness of neural language models to input perturbations. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 1558–1570, 2021. doi: 10.18653/v1/2020.acl-main.755 3

[33] H. Mozannar, J. Lee, D. Wei, P. Sattigeri, S. Das, and D. Sontag. Effective human-ai teams via learned natural language rules and onboarding. *Advances in Neural Information Processing Systems*, 36:30466–30498, 2023. doi: 10.48550/arXiv.2311.01007 3, 4

[34] A. Patania, F. Vaccarino, and G. Petri. Topological analysis of data. *EPJ Data Science*, 6(7), 2017. doi: 10.1140/epjds/s13688-017-0104-x 3

[35] J. Pennington, R. Socher, and C. D. Manning. GloVe: Global vectors for word representation. *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014. doi: 10.3115/v1/D14-1162 3

[36] E. Purvine, D. Brown, B. Jefferson, C. Joslyn, B. Praggastis, A. Rathore et al. Experimental observations of the topology of convolutional neural network activations. In *Proceedings of the 37th AAAI Conference on*

Artificial Intelligence (AAAI), 2023. doi: 10.1609/aaai.v37i8.26134 1, 3

[37] K. Qian, S. Wan, C. Tang, Y. Wang, X. Zhang, M. Chen et al. VarBench: Robust language model benchmarking through dynamic variable perturbation. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 16131–16161, 2024. doi: 10.48550/arXiv.2406.17681 3

[38] Y. Qiang, S. Nandi, N. Mehrabi, G. Ver Steeg, A. Kumar, A. Rumshisky et al. Prompt perturbation consistency learning for robust language models. In *Findings of the Association for Computational Linguistics: EACL 2024*, pp. 1357–1370. Association for Computational Linguistics, St. Julian's, Malta, Mar. 2024. doi: 10.18653/v1/2024.findings-eacl.91 3

[39] A. Rathore, N. Chalapathi, S. Palande, and B. Wang. TopoAct: Exploring the shape of activations in deep learning. *Computer Graphics Forum*, 40(1):382–397, 2021. doi: 10.1111/cgf.14195 1, 2, 3, 4, 7

[40] A. Rathore, S. Dev, J. M. Phillips, V. Srikumar, Y. Zheng, C.-C. M. Yeh et al. VERB: Visualizing and interpreting bias mitigation techniques for word representations. *ACM Transactions on Interactive Intelligent Systems*, 14(1):1–34, 2023. doi: 10.1145/3604433 1

[41] A. Rathore, Y. Zhou, V. Srikumar, and B. Wang. TopoBERT: Exploring the topology of fine-tuned word representations. *Information Visualization*, 22(3):186–208, 2023. doi: 10.1177/14738716231168671 1, 2, 3, 5, 6, 7

[42] D. Ren, F. Hohman, H. Lin, and D. Moritz. Embedding Atlas: Low-friction, interactive embedding visualization. In *2025 IEEE Visualization and Visual Analytics (VIS)*, pp. 191–195, 2025. doi: 10.1109/VIS60296.2025.00044 3

[43] A. Rogers, O. Kovaleva, and A. Rumshisky. A primer in BERTology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics*, 8:842–866, 2020. doi: 10.1162/tacl_a_00349 1, 2, 7

[44] D. Romero-Alvarado, J. Hernández-Orallo, and F. Martínez-Plumed. How resilient are language models to text perturbations? In *International Conference on Intelligent Data Engineering and Automated Learning*, pp. 85–96, 2024. doi: 10.1007/978-3-031-77731-8_8 3

[45] P. Rosen, M. Hajij, and B. Wang. Homology-preserving multi-scale graph skeletonization using mapper on graphs. *IEEE Workshop on Topological Data Analysis and Visualization (TopoInVis)*, pp. 10–20, 2023. doi: 10.1109/TopoInVis60193.2023.00008 3

[46] N. Schneider and N. A. Smith. A corpus and model integrating multiword expressions and supersenses. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1537–1547. Association for Computational Linguistics, Denver, Colorado, May–June 2015. doi: 10.3115/v1/N15-1177 7

[47] R. Sevastjanova, E. Cakmak, S. Ravfogel, R. Cotterell, and M. El-Assady. Visual comparison of language model adaptation. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):1178–1188, 2022. doi: 10.1109/TVCG.2022.3209458 1, 3

[48] R. Sevastjanova, R. Gerling, T. Spinner, , and M. El-Assady. LayerFlow: Layer-wise exploration of llm embeddings using uncertainty-aware interlinked projections. *Computer Graphics Forum*, 2025. doi: 10.1111/cgf.70123 1, 3, 4

[49] R. Sevastjanova, A.-L. Kalouli, C. Beck, H. Hauptmann, and M. El-Assady. Explaining Contextualization in Language Models using Visual Analytics. In *Proceedings of the Association for Computational Linguistics*, ACL. ACL, 2021. doi: 10.48448/1bf4-bg31 3

[50] R. Sevastjanova, A.-L. Kalouli, C. Beck, H. Hauptmann, and M. El-Assady. LMFingerprints: Visual explanations of language model embedding spaces through layerwise contextualization scores. *Computer Graphics Forum*, 41(3):295–307, 2022. doi: 10.1111/cgf.14541 1, 3, 6

[51] D. Shi, F. Cheng, T. Weinkauf, A. Oulasvirta, and M. El-Assady. DxHF: Providing high-quality human feedback for llm alignment via interactive decomposition. In *Proceedings of the 38th annual acm symposium on user interface software and technology*. Association for computing Machinery, 2025. doi: 10.1145/3746059.3747600 5

[52] G. Singh, F. Mémoli, and G. E. Carlsson. Topological methods for the analysis of high dimensional data sets and 3D object recognition. *Eurographics Symposium on Point-Based Graphics*, pp. 91–100, 2007. doi: 10.2312/spbg/spbg07/091-100 1, 2, 3

[53] V. Sivaraman, Y. Wu, and A. Perer. Emblaze: Illuminating machine learning representations through interactive comparison of embedding spaces. In *27th Int. Conf. on Intelligent User Interfaces*, pp. 418–432, 2022. doi: 10.1145/3490099.3511137 3

[54] P. Solunke, V. Guardieiro, J. Rulff, P. Xenopoulos, G. Y.-Y. Chan, B. Barr et al. Mountaineer: Topology-driven visual analytics for comparing local explanations. *IEEE Transactions on Visualization and Computer Graphics*, 30(12):7763–7775, 2024. doi: 10.1109/TVCG.2024.3418653 3

[55] I. Tenney, D. Das, and E. Pavlick. BERT rediscovers the classical NLP pipeline. In *Proc. of the Annual Meeting of the Association for Computational Linguistics*, pp. 4593–4601. ACL, Florence, Italy, July 2019. doi: 10.18653/v1/P19-1452 6

[56] H. Voigt, M. Meuschke, S. Zarrieß, and K. Lawonn. KeywordScape: Visual document exploration using contextualized keyword embeddings. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 137–147, 2022. doi: 10.18653/v1/2022.emnlp-demos.14 3

[57] A. Waldis, Y. Perlitz, L. Choshen, Y. Hou, and I. Gurevych. Holmes: A benchmark to assess the linguistic competence of language models. *Transactions of the Association for Computational Linguistics*, 12:1616–1647, 2024. doi: 10.1162/tacl_a_00718 4

[58] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, and M. Zhou. MiniLM: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in Neural Information Processing Systems*, 33:5776–5788, 2020. doi: 10.48550/arXiv.2002.10957 5

[59] Y. Wang, Z. Zhang, and R. Wang. Element-aware summarization with large language models: Expert-aligned evaluation and chain-of-thought method. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8640–8665. Association for Computational Linguistics, Toronto, Canada, July 2023. doi: 10.18653/v1/2023.acl-long.482 3

[60] Z. J. Wang, F. Hohman, and D. H. Chau. WizMap: Scalable interactive visualization for exploring large machine learning embeddings. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pp. 516–523. Association for Computational Linguistics, Toronto, Canada, July 2023. doi: 10.18653/v1/2023.acl-demo.50 3

[61] B. Warner, A. Chaffin, B. Clavié, O. Weller, O. Hallström, S. Taghadouini et al. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2526–2547. Association for Computational Linguistics, Vienna, Austria, July 2025. doi: 10.18653/v1/2025.acl-long.127 7, 8

[62] X. Yan, S. Liu, K. Thopalli, and B. Wang. Visual exploration of feature relationships in sparse autoencoders with curated concepts. In *Mechanistic Interpretability Workshop at NeurIPS 2025*. doi: 10.48550/arXiv.2511.06048 3

[63] X. Yan, X. Xuan, J. P. Ono, J. Guo, V. Mohanty, S. A. Kumar et al. VISLIX: An XAI framework for validating vision models with slice discovery and analysis. In *Computer Graphics Forum*, vol. 44, p. e70125. Wiley Online Library, 2025. doi: 10.1111/cgf.70125 4

[64] Q. Yang, A. Steinfeld, C. Rosé, and J. Zimmerman. Re-examining whether, why, and how human-ai interaction is uniquely difficult to design. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, pp. 1–13. Association for Computing Machinery, New York, NY, USA, 2020. doi: 10.1145/3313831.3376301 3

[65] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi. BERTScore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*, 2019. doi: 10.48550/arXiv.1904.09675 5

[66] T. Zhang, F. Ladhak, E. Durmus, P. Liang, K. McKeown, and T. B. Hashimoto. Benchmarking large language models for news summarization. *Transactions of the Association for Computational Linguistics*, 12:39–57, 2024. doi: 10.1162/tacl_a_00632 3

[67] Y. Zhou, N. Chalapathi, A. Rathore, Y. Zhao, and B. Wang. 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, 2021. doi: 10.1109/PacificVis52677.2021.00021 1, 2, 3, 5

[68] Y. Zhou, Y. Zhou, J. Ding, and B. Wang. Visualizing and analyzing the topology of neuron activations in deep adversarial training. *Topology, Algebra, and Geometry in Machine Learning (TAGML) Workshop at ICML*, 2023. 1, 3

[69] J. E. Zini and M. Awad. On the explainability of natural language processing deep models. *ACM Computing Surveys*, 55(5), art. no. 103, 31 pp., dec 2022. doi: 10.1145/3529755 1