Embedding Comparator: Visualizing Differences in Global Structure and Local Neighborhoods via Small Multiples

Angie Boggust*
MIT CSAIL
Cambridge, Massachusetts, USA
aboggust@csail.mit.edu

Brandon Carter*
MIT CSAIL
Cambridge, Massachusetts, USA
bcarter@csail.mit.edu

Arvind Satyanarayan
MIT CSAIL
Cambridge, Massachusetts, USA
arvindsatya@mit.edu

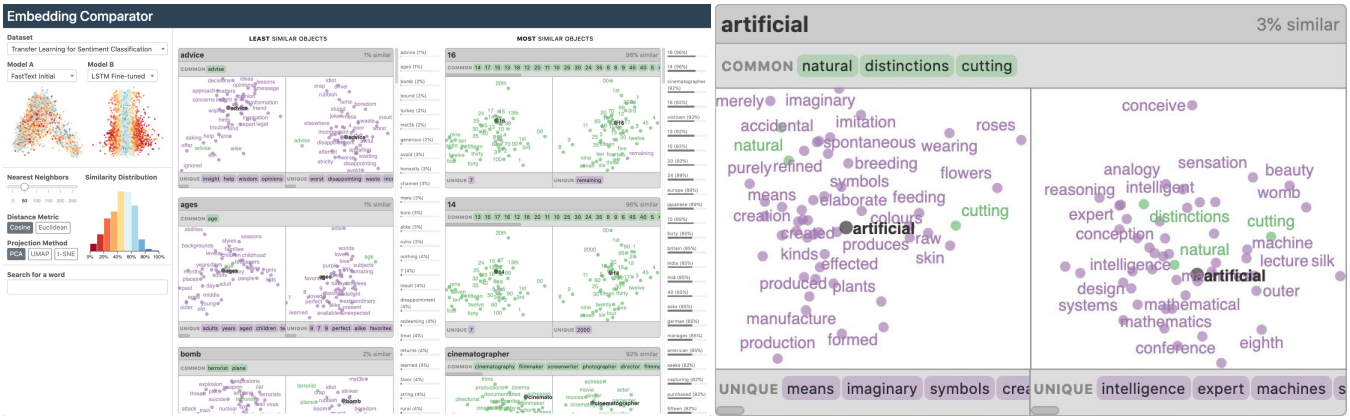

Figure 1: The Embedding Comparator (left) facilitates comparisons of embedding spaces via *local neighborhood dominoes*: small multiple visualizations depicting local substructures (right).

ABSTRACT

Embeddings mapping high-dimensional discrete input to lower-dimensional continuous vector spaces have been widely adopted in machine learning applications as a way to capture domain semantics. Interviewing 13 embedding users across disciplines, we find comparing embeddings is a key task for deployment or downstream analysis but unfolds in a tedious fashion that poorly supports systematic exploration. In response, we present the Embedding Comparator, an interactive system that presents a global comparison of embedding spaces alongside fine-grained inspection of local neighborhoods. It systematically surfaces points of comparison by computing the similarity of the k-nearest neighbors of every embedded object between a pair of spaces. Through case studies across multiple modalities, we demonstrate our system rapidly reveals insights, such as semantic changes following fine-tuning, language changes over time, and differences between seemingly similar models. In evaluations with 15 participants, we find our system accelerates comparisons by shifting from laborious manual specification to browsing and manipulating visualizations.

*Both authors contributed equally to this research.

IUI '22, March 22–25, 2022, Helsinki, Finland

ACM ISBN 978-1-4503-9144-3/22/03.
https://doi.org/10.1145/3490099.3511122

CCS CONCEPTS

• **Human-centered computing** → **Visualization systems and tools**; • **Computing methodologies** → *Machine learning*.

KEYWORDS

machine learning, embedding spaces, visualization system, interactive, small multiples

ACM Reference Format:
Angie Boggust, Brandon Carter, and Arvind Satyanarayan. 2022. Embedding Comparator: Visualizing Differences in Global Structure and Local Neighborhoods via Small Multiples. In *27th International Conference on Intelligent User Interfaces (IUI '22), March 22–25, 2022, Helsinki, Finland*. ACM, New York, NY, USA, 21 pages. https://doi.org/10.1145/3490099.3511122

1 INTRODUCTION

Embedding models map high-dimensional discrete objects into lower-dimensional continuous vector spaces such that vectors of related objects are located close together. Although the individual dimensions and structure of embedding spaces can be challenging to interpret, embeddings have become widely used in machine learning (ML) applications because their structure usefully captures domain-specific semantics. For example, in natural language processing (NLP), embeddings map words into real-valued vectors in a way that co-locates semantically similar words [50].

A critical task when working with embedding models is evaluating their learned representations. For instance, users may wish to determine if embeddings can transfer to a low-resource domain (e.g., applying general English embeddings to medical text [28]). In speech recognition [5], computer vision [3], recommendation [34],

computational biology [6, 7], multimodal learning [56], and computational art [15, 19], evaluating embeddings has informed training procedures and revealed the impact of different training datasets, model architectures, hyperparameters, and model initializations.

To understand how people evaluate and compare embeddings, we conducted a series of semi-structured interviews with users across disciplines who frequently use embedding models as part of their research or in application domains (Section 3). Users balance between examining global semantic structure via dimensionality reduction plots and inspecting local neighborhoods of specific embedded objects. Our conversations reveal shortcomings of these approaches, including unprincipled object selection strategies that rely heavily on domain knowledge or repetitive ad hoc analysis and siloed tools that focus on either one model at a time or depict only the global structure of the embedding space. As a result, users feel concerned that they may miss unexpected insights or lack a comprehensive understanding of the embedding space. Moreover, users cannot develop tight feedback loops or rapidly iterate between generating and answering hypotheses as their current processes include limited interactive capabilities and, thus, require tedious manual specification.

In response, we present the *Embedding Comparator*, an interactive system for analyzing a pair of embedding models. Drawing on the insights from our formative interviews, the Embedding Comparator balances between visualizing the embeddings' global structures with comparing the local neighborhoods (Section 4). To easily identify the similarities and differences between the two models, the system calculates a similarity score for every embedded object based on its reciprocal local neighborhood (i.e., the number of nearest neighbors an object shares between the two models and how many are unique). These scores are visualized in several ways, including through a histogram of scores, through color-encoding the global geometry plots, and critically, through *local neighborhood dominoes*: small multiple visualizations that facilitate rapid comparisons of local substructures. A variety of interactive mechanics help facilitate a tight iterative loop between analyzing these global and local views — for instance, by interactively selecting points in the global plots, or by searching for specific objects, users can filter dominoes, and hovering over dominoes highlights their points in the global views to provide additional context.

We demonstrate how the Embedding Comparator helps scaffold and accelerate real-world exploration and analysis of embedding spaces through case studies and first-use studies. Using tasks based on our formative interviews, we show how our system supports use cases, such as understanding the effects of fine-tuning, conducting linguistic analysis, and exploring multimodal embeddings. Our system design enables the replication of previously published results using only a handful of interactions and without the need for task-specific metrics. As we demonstrate in case studies (Section 5) and validate in first-use studies (Section 6), the Embedding Comparator shifts the process of analyzing embeddings from tedious and error-prone manual specification to browsing and manipulating a series of visualizations.

The Embedding Comparator is freely available as open-source software, with source code at: https://github.com/mitvis/embedding-comparator, and a live demo at: http://vis.mit.edu/embedding-comparator.

2 RELATED WORK

2.1 Machine Learning Interpretability

ML models are widely regarded as being "black boxes" as it is difficult for humans to reason about how models arrive at their decisions [40]. Numerous tools help users understand model behavior [26], including visualizations for specific architectures [4, 42, 62, 63]. More general techniques involve evaluating input feature importance [9, 55, 59, 64], saliency [51], or neuron activations [31]. Boxer [18] compares discrete-choice classifiers, but treats them as black boxes without considering their internals. In contrast to these methods, our focus is on comparing the *representations* learned by *different* models, as internal representations may differ even while input saliency or input-output behavior remains the same. In our formative interviews (Section 3), we found users often compare these internal representations (e.g., to identify semantic differences between hidden layers of a particular model).

2.2 Visual Embedding Techniques and Tools

Interpreting the representations learned at the embedding layers of ML models is challenging as embedding spaces are generally high-dimensional and latent. In visual analytics, a variety of techniques have been developed to visualize high-dimensional data and span multiple stages of the visualization pipeline, including data transformation, visual mapping, and view transformation [43]. To reason about embedding spaces, researchers often project the high-dimensional vectors down to two or three dimensions using techniques such as principal component analysis (PCA) [30], t-SNE [69], and UMAP [47]. Visualizing these projections reveals the global geometry of these spaces as well as potential substructures such as clusters, but effectively doing so may require careful tuning of hyperparameters [72] — a process that can require non-trivial ML expertise. Prior work has demonstrated that the choice of dimensionality reduction technique can impact downstream data analysis [77]. Thus, the Embedding Comparator precomputes projections using PCA, t-SNE, and UMAP, and it provides a modular system design so users can use a dimensionality reduction technique of their choice. (Section 4.2). The system defaults to PCA, which highlights the global structure of the embedding space and is deterministic, a desire of our formative interviewees (Sections 3 and 4). Tight integration between dimensionality reduction visualizations and interactivity has been shown to be an integral component of visual analytics pipelines [57], and we adopt this strategy in our system design (Section 4).

By default, many projection packages generate visualizations that are static and thus do not facilitate a tight question-answering feedback loop as users need to repeatedly regenerate visualizations, slowing down the exploration process. Recently, researchers have begun to explore interactive systems for exploring embeddings including via direct manipulation of the projection [27, 53, 60], interactively filtering and reconfiguring visual forms [22, 67], and defining attribute vectors and analogies [44]. While our approach draws inspiration from these prior systems, and similarly provides facilities for exploring local neighborhoods, the Embedding Comparator primarily focuses on identifying and highlighting the *similarities* and *differences* between different representations of embedded objects. To do so, we compute a similarity metric for every embedded

object and use this metric to drive several interactive visualizations (Section 4).

2.3 Methods for Comparing Embedding Spaces

To compare spaces, some techniques align embeddings through linear transformation [10, 20, 21, 49, 65] or alignment of neurons or the subspaces they span [39, 70]. In contrast, the Embedding Comparator does not align the embeddings and can be used in cases where a linear mapping between the spaces does not exist, which may occur if they have different structures [49]. Our system exposes the objects that are most and least similar between two vector spaces via a reciprocal local neighborhood similarity metric, and local neighborhood based metrics have been shown to usefully capture differences in embedding spaces [20]. As desired by our interviewees, our system is deterministic and faithful to the reciprocal local neighborhoods of the models (unlike transformations that rely on linear maps or weights learned by stochastic gradient optimization).

Other techniques require users to first identify and query particular objects of interest or are restricted to particular types of data or models. Unlike parallel embeddings [2], our system does not rely on clustering or assume embeddings form clusters corresponding to semantically meaningful concepts. EmbeddingVis [38] compares network (graph) embeddings through relationships between node metrics and embeddings. Liu et al. [41] identify differences in how word2vec and GloVe embeddings capture syntactic relationships, but do so by visualizing semantic and syntactic analogies specifically in the context of neural word embeddings. Similarly, Liu et al. [44] evaluate consistency of attribute and analogy vectors across latent embedding spaces. In contrast, our system does not rely on analogies, which may only exist for certain classes of embedding models [1, 16]. Heimerl et al. [22] present visualizations of nearest neighbors and co-occurrences for word embeddings to show how the meaning of a word changes over time, but do not automatically identify such objects whose representations differ most between models. Wang et al. [71] use cosine distance to find nearest neighbors between embeddings and then arbitrarily sample words to determine if the semantic meanings differ between the models. Other work has demonstrated differences in representations learned by convolutional neural networks (CNNs) by deconvolution of representations of sampled images [78], alignment of hidden units with interpretable concepts [3], or projections of representations at different layers or epochs [54]. The Python software package `repcomp` [58] quantifies the difference between two embedding spaces through local neighborhood similarity but only outputs a single global similarity value between the spaces and does not permit fine-grained inspection of object-level differences. In contrast, the Embedding Comparator computes a similarity metric for every embedded object and initializes its view to begin with objects that are the most and least similar between the two models. As we find in our formative interviews (Section 3), users would benefit from tools that systematically identify objects of interest, alleviating the need to sample them in a random or biased way. Moreover, we present these objects in an interactive graphical system that facilitates rapid exploration of differences between the embedding spaces.

		Users												
		Model-Driven								Data-Driven				
		1	2	3	4	5	6	7	8	9	10	11	12	13
Domain	Machine Learning (M) / Computational Biology (B) Digital Humanities (H)	M	M	M	M	M	M	B	B	M	H	H	B	H
Affiliation	Academia (A) / Industry (I)	A	A	I	A	A	A	A	I	I	A	A	A	A
Title	Graduate Student (S) / Engineer (E) Researcher (R) / Professor (P)	R	R	R	R	S	S	S	E	E	P	P	S	R
Tasks and Goals														
Project Types	Machine translation	X	X	X	X									
	Multimodal machine learning	X			X				X					
	Natural language processing		X	X					X					
	Computer vision	X			X		X		X					
	Knowledge graphs					X								
	Data compression						X							
	Computational biology			X		X		X	X	X			X	
	Computational linguistics										X	X		X
Use Cases	Understanding model performance	X	X	X	X	X								
	Selecting embeddings for model initialization	X			X		X							
	Selecting embeddings for downstream tasks	X	X		X	X			X	X	X	X	X	
	Understanding differences in datasets			X				X			X	X	X	X
Comparison Tasks	Different models on the same inputs	X	X		X	X		X	X					
	Different layers of the same model	X	X	X	X									
	Learned embeddings to ground truth data				X			X					X	
	Pre-trained and fine-tuned embeddings		X	X					X			X		
	Embedding spaces trained on different data									X	X	X	X	X
Processes and Chanllenges														
Tools	Dimensionality reduction	X	X	X	X	X	X	X				X		X
	Clustering	X	X		X	X		X			X	X		X
	K-nearest neighbors	X		X	X	X	X	X	X	X		X	X	X
Comparison Strategies	Select objects and analyze nearest neighbors	X		X	X	X	X	X	X	X		X	X	X
	Reduce dimensionality of the embedding space	X	X	X	X	X		X				X		X
	Evaluate performance on a downstream task	X	X		X		X		X	X	X			
	Align embedding spaces			X									X	X
Comparison Challenges	Selecting objects to analyze is ad hoc	X			X	X			X				X	X
	Querying single objects doesn't yield global insight	X			X				X			X		
	Dimension reduction may not segment the space		X		X	X								X
	Techniques produce inconsistent patterns				X			X		X				X

Figure 2: A matrix summary of our formative interviews grouped by user class. Users compare embedding spaces for a variety of tasks and goals, but find current processes unstructured and inconsistent due to their reliance on ad hoc exploration.

Concurrent with our work, Heimerl et al. [23] developed the embedding comparison system empComp. embComp facilitates top-down comparison of embeddings by displaying summary visualizations of embedding space differences and employing interaction to drill down to individual objects. Through a survey of embedding comparison tasks and motivating examples, Heimerl et al. [23] underscore the importance of local neighborhood overlap metrics to quantify embedding space differences, validating the Embedding Comparator's use of local neighborhood similarity. However, unlike embComp's top-down approach, the Embedding Comparator simultaneously visualizes global views of embedding structure alongside local views of individual objects and their common and unique neighbors to enable efficient analysis of both aspects (Design Goals 2 and 3). By interactively linking these global and local views (Design Goal 4), the Embedding Comparator permits rapid hypothesis generation and upholds known design goals of visual ML interpretability systems [25].

3 FORMATIVE INTERVIEWS

To understand how embedding spaces are analyzed and compared, and to identify process pitfalls and limitations of existing tools, we conducted a series of semi-structured interviews with 13 embedding users from both academia and industry. As listed in Fig. 2,

our users work in a diverse range of domains, hold a variety of titles, and use embeddings in a wide range of projects. We recruited embedding users from within our professional network (7/13) and through an open call in our organizations and on Twitter (6/13). Since our goal was to learn how users use embedding spaces, we required users to have experience working with high-dimensional data. The resulting participants all worked with learned embeddings; however, many of them were not machine learning experts and used embeddings for downstream tasks in other domains. We initially interviewed four users, and then conducted the remaining interviews throughout the development process to iteratively refine the Embedding Comparator and increase diversity of the participant pool.

We conducted semi-structured interviews with one interviewee at a time. Each interview lasted 30–60 minutes. We initially conducted interviews with users in person but switched to video chat due to the COVID-19 pandemic. In each interview, we began by describing our study objective: to understand the landscape of embedding use. We then asked users open-ended questions and encouraged users to describe their specific experience with embeddings such as *"describe how you used embedding spaces in a recent project."* Our questions aimed to answer the following questions:

- What types of projects are embeddings used in?
- How do embeddings help users achieve their goals?
- When do users compare embedding spaces?
- What tools and strategies are used to compare embeddings?
- What challenges do users face when comparing embeddings?

3.1 User Tasks and Goals

Through our interviews, we find users compare embeddings in consistent ways, but their tasks and goals for comparing embeddings differs significantly (Fig. 2). Specifically, we find users fall into two categories: *model-driven* users who compare embedding spaces as a way to understand model performance and behavior; and *data-driven* users who study embedded representations to uncover properties of the underlying data.

Model-driven users. Model-driven users analyze and compare embeddings as a way to develop a deep knowledge of how their models work and why. These users are interested in questions such as *what is responsible for the performance improvement between two models?* (U1–U5), *what embeddings should I use for initialization?* (U1, U4, U6), *what model will perform the best on my downstream task?* (U1, U2, U4, U5, U8), and *how do differences in training data affect model behavior?* (U3, U7). To answer these questions, model-driven users perform embedding comparison tasks such as: comparing learned embeddings from different layers of the same model, comparing embeddings from different models trained on the same data, comparing learned embeddings to the ground truth, and comparing generic embeddings to those that have been fine-tuned for a particular task.

Data-driven users. Data-driven users utilize embeddings as a way to represent and understand relationships in the data they study. These users typically work on projects in applied machine learning domains like computational biology or historical linguistics and use embedding spaces *"to investigate questions that have been made through non-computational methods"* (U13). Example questions from data-driven users we interviewed include: *what are the structural relationships between protein sequences?* (U12), *what is the relationship between x-ray images and corresponding radiology reports?* (U9), and *how has the plot speed in fiction novels shifted over time?* (U10). Use cases include selecting embeddings for downstream tasks (e.g., comparing pre-trained word embeddings to embeddings fine-tuned on task-specific literature for document classification [U11]) and using embeddings to analyze differences in the data (e.g., comparing word embeddings trained on literature from different centuries [U13] or comparing embedded protein relationships to ground truth data [U12]).

3.2 Current Processes and Challenges

Through our interviews, we find model-driven and data-driven users use similar tools, apply similar strategies, and run into common problems when comparing embedding spaces. A common workflow, referred to as *"global check and local check"* by U5 and their colleagues, involves comparing the global structures of embedding spaces via dimensionality reduction and then selecting objects within the space and comparing their k-nearest neighbors. Analyzing the global structure of embedding spaces enables users to gain insight into the semantic concepts represented by each embedding space, while comparing local neighborhoods of select objects can unearth unexpected relationships between objects or confirm user hypotheses. For example, when studying how literary texts personify non-human characters, U11 analyzed the global structure of their embedding spaces using dimensionality reduction techniques, finding the spaces had separated into discrete clusters representing personhood identifiers: professions, education types, etc. This finding propelled U11 to probe the local neighborhoods of known character types within each cluster like "*pilot*" to discover new associated personifying words like "*train conductor*".

Comparing local neighborhoods of specific objects was a critical task for the majority of our users (11/13), especially in cases when global projections were unable to meaningfully segment the space; however, many users expressed concerns that the way they selected objects to analyze was unprincipled and reduced their confidence in their results. Data-driven users selected objects using their domain expertise (e.g., words known to change over time [U11, U13] or proteins known to interact with SARS-CoV-2 [U12]), but expressed concern that this approach would not uncover unexpected results. Model-driven users selected objects they expected would be challenging for the model (U6), objects they hypothesized would display insightful differences (U1, U4, U5), or objects selected at random (U6); however, they were often concerned that this unsystematic approach prevented them from understanding the entire space and could be viewed as cherry-picking by the research community. To mitigate these concerns, many users additionally compared embeddings on quantitative downstream tasks (e.g., comparing word embeddings on genre classification [U10]). While doing so added some sense of rigor to their process, users found this procedure to be *"embarrassingly empirical"* (U10) because it did not provide the same rich insights users got from local neighborhood exploration. As U1 emphasized, *"Qualitative [analysis] is often more powerful than just a single quantitative number."*

Throughout our interviews, users expressed concerns about the unreliability of common analysis techniques such as t-SNE and embedding alignment algorithms (U4, U7, U9, U13). We found users distrust t-SNE due to its sensitivity to hyperparameters and stochasticity, which can lead to wildly different projections and often times misleading results [72]. Comparing embeddings using t-SNE caused users to distrust their conclusions as they were unable to draw a distinction between "real" findings and t-SNE artifacts. This led U4 to stop using t-SNE altogether, noting, *"You can tease apart whatever you want from t-SNE. Sometimes it shows you what you want and if it doesn't then you can spend a bit of time until you see what you want to see."* Users expressed similar concerns about embedding space alignment algorithms, such as orthogonal Procrustes, because the algorithms can produce unreliable and meaningless alignments (see Section 2.3). While U13 previously applied Procrustes alignment, they are hesitant to use it again in the future because they could *"only find the story [they] wanted to tell sometimes"* and struggled to determine whether their results were representative of the embedding spaces or resulted from the unpredictability of the method.

3.3 Embedding Comparator Design Goals

To inform the design of the Embedding Comparator, we distill our formative interviews into the following design goals:

1. **Surface similarities and differences for systematic comparison.** Across all users, the central goal of comparing embedding models is to understand the degree to which they are similar, and where the differences lie. A recurrent breakdown is how unprincipled and time-consuming identifying this information is with current approaches (e.g., performing a dimensionality reduction that does not result in informative separation or manually generating objects to analyze via k-nearest neighbors). Moreover, our interviews suggest that users are most interested in seeing the objects that are most similar or different — as U2 noted, *"It is most interesting what happens at the extremes."*
2. **Display projection plots to surface global semantic differences between embedding spaces.** The majority of our users (9/13) use dimensionality-reduced projection plots as a primary mechanism for evaluating embedding spaces. Besides users' familiarity with these views, our interviews highlighted that visualizing global structure provides necessary context to meet our first design goal as the overall shape, density, and clustering of a projection can reveal similarities and differences in a glance. For example, U1 described how viewing the global projection of a particular embedding model caused them to stumble upon an unexpected structure they later wrote a paper about.
3. **Compare local neighborhoods to identify object-level similarities and differences.** While projection plots provide useful global context, the substance of our users' analysis occurs by drilling down into and comparing the local neighborhoods of embedded objects. For example, U2 noted that even in their papers they *"often report a few nearest neighbors of their models to show that they capture [meaningful] properties"*. The Embedding Comparator must provide interface elements that facilitate rapid comparisons of reciprocal local neighborhoods for each embedded object.
4. **Interactively link global and local views for rapid analysis.** Users typically explained their global and local analysis as a single technique; however, current tools force users to complete each task independently by first projecting the high dimensional space and independently querying for objects of interest. The lack of interactivity slows down their analysis processes and makes it difficult to deeply explore and identify differences that are not obvious. Thus, the Embedding Comparator should use interactive techniques to link global and local views together, including allowing users to select local views from global views, highlight local neighborhoods within the global projections, and perform targeted searches to surface the local neighborhoods of specific embedded objects.
5. **Yield deterministic and reproducible results that inspire user confidence.** Our formative interviews revealed many users avoided unreliable analysis techniques such as t-SNE and Procrustes alignment. Users want to be confident the patterns they uncover are representative of true patterns in the embedding space and are not artifacts of non-deterministic tools. As such, the Embedding Comparator should leverage techniques that produce reproducible results users can trust.

4 SYSTEM DESIGN

Informed by our formative interviews, the Embedding Comparator computes a similarity score for every embedded object based on its reciprocal local neighborhoods. Critically, this similarity metric does not require the two models to have the same dimensionality nor do they need to be aligned in any way. As a result, the Embedding Comparator is capable of supporting a wide range of embedding models. Furthermore, the simplicity of computing overlap between reciprocal local neighborhoods makes this metric intuitive to a broad range of users, which we validate in our first-use studies (Section 6). To allow users to rapidly identify similarities and differences between the models, this similarity metric is visualized via a number of global views (including projection plots and a histogram distribution) as well as through *local neighborhood dominoes*: small multiple views of local substructures.

4.1 Computing Local Neighborhood Similarity

An embedding space is a function $E : \mathcal{V} \rightarrow \mathbb{R}^d$ mapping objects in vocabulary $\mathcal{V}$ into a d-dimensional real-valued vector space. For example, in NLP, $\mathcal{V}$ may be a vocabulary of English words, and E maps each word to a 200-dimensional vector.

Here, we consider two embedding spaces E_1 and E_2 over the same set of objects $\mathcal{V}$. The embedding spaces may have different bases, a different number of dimensions d, or stem from different modalities. For this reason, we compute an embedded object's similarity via the reciprocal local neighborhood around the object in each of the embedding spaces. Precisely, for each object $w \in \mathcal{V}$, we compute the *local neighborhood similarity* (LNS) of w between the two embedding spaces as:

$$\text{LNS}(w) = S(k\text{-NN}_1(w),\ k\text{-NN}_2(w)) \tag{1}$$

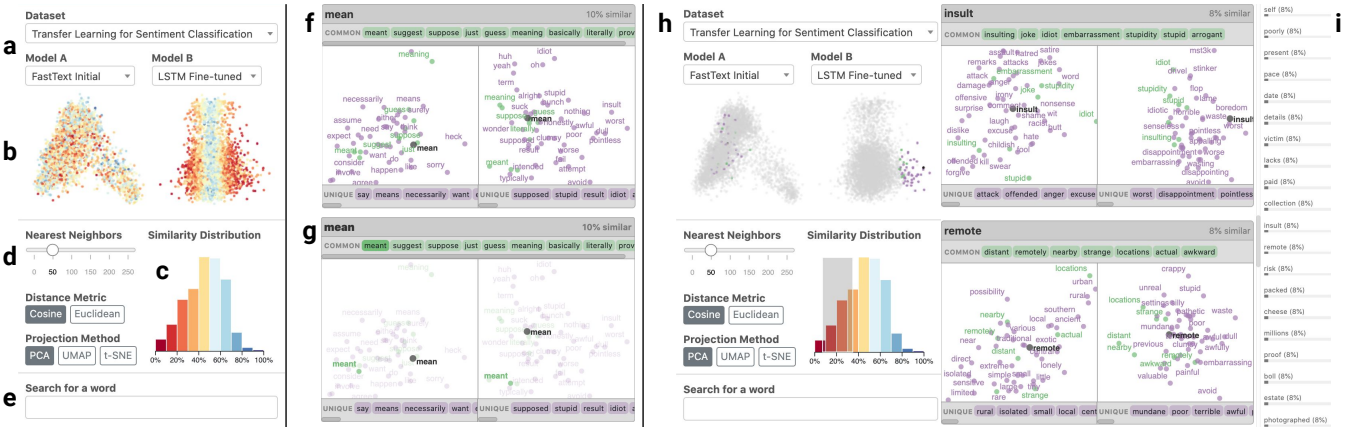

Figure 3: The Embedding Comparator interactively links global and local views. Users populate the Embedding Comparator by selecting a pair of embedding models (a). Global projections (b) visualize the geometry of the embeddings, and local neighborhood dominoes (f) visualize an object's local neighborhood. Users explore by brushing over the global projection plots (b) and similarity distribution (c), tuning the neighborhood parameters (d), searching for objects (e), and selecting dominoes from the domino outline (i). Hovering on a domino highlights its neighborhood in the global projection (h), and hovering on an object in a domino highlights that object in the local neighborhood plots (g).

where $k\text{-NN}_i(w)$ returns the k-nearest neighbors of w in embedding space E_i and S is a similarity metric between the two lists of nearest neighbors. We compute k-nearest neighbors with cosine or Euclidean distance [68] and take S as the Jaccard (intersection over union) similarity between the sets of neighbors. The Jaccard similarity between two sets C_1 and C_2 is defined as $J(C_1, C_2) = \frac{|C_1 \cap C_2|}{|C_1 \cup C_2|}$ and scales between 0 (the sets are disjoint) and 1 (the sets are identical). Users can choose between distance metrics in the interface or introduce additional functions via a JavaScript API call. The LNS metric is deterministic (Design Goal 5), and it enables the Embedding Comparator to systematically sort and identify objects of interest, alleviating the need for users to sample them in a random or biased way or to formulate hypotheses *a priori* about which objects to investigate (Design Goal 1).

The Embedding Comparator scales linearly with the number of objects. Given two d-dimensional embedding spaces with $|\mathcal{V}|$ objects, k-nearest neighbors for all objects are precomputed using an approximately $O(d \cdot |\mathcal{V}| \log |\mathcal{V}|)$ algorithm (ball tree) and LNS is computed in $O(|\mathcal{V}| \cdot k)$ time. Our system is implemented in JavaScript using React and performs efficiently on the real-world case studies explored in this paper (300-dimensional embeddings and roughly 6000 objects).

4.2 Global Views

The Embedding Comparator's left-hand sidebar (Fig. 3a–e) provides configuration options and interactive global views of the embedding spaces. A user begins by selecting a *dataset*, which specifies the embedding vocabulary, followed by two *models*, each of which defines the embedding space for that vocabulary (Fig. 3a). Users can load many models, and by decoupling datasets from models, the Embedding Comparator makes it easy to compare several models with the same vocabulary.

Beneath each model, the Embedding Comparator shows a *global projection* (Fig. 3b): a scatter plot that depicts the geometric structure of the model's embedding space, with object names shown in a tooltip on hover. Per Design Goal 2, these projections provide valuable context during exploration and help reveal interesting global properties (e.g., distinct clusters). Based upon our formative interviews and design goals, we load PCA projections by default because PCA is deterministic (Design Goal 5) and highlights the global structure of the embedding space. However, we provide a modular system design, so users can interactively switch to projections computed using PCA, UMAP, or t-SNE via buttons in the sidebar.

The *similarity distribution* (Fig. 3c) displays the distribution of LNS similarity values (Eq. 1) over all objects in the embedding space. Bars are colored using a diverging red-yellow-blue color scheme to draw attention to the most extreme values (objects that are the most and least similar between the two selected models) per Design Goal 1. We reapply this color encoding in the global projections to help users draw connections between the two visualizations and to reveal global patterns or clusters of objects with comparable similarity values (e.g., case studies of Fig. 4 and Fig. 5). Both the similarity distribution and the global projections can be used to interactively filter the dominoes (Design Goal 4; Section 4.3), and the *search bar* (Fig. 3e) can be used to populate the dominoes of specific object(s) of interest.

The *parameter controls* (Fig. 3d) enable the user to interactively change the value of k used to define the size of local neighborhoods for computing similarity, select the distance metric used for computing distance between vectors, and select the dimensionality reduction method used in the global projections and local neighborhood plots. The default value of k is 50, which we found to provide insightful results across our various experiments and case studies. Changes in either of these controls immediately update the rest of the interface (Design Goal 4).

4.3 Local Neighborhood Dominoes

To meet Design Goal 3, the Embedding Comparator introduces *local neighborhood dominoes*: a small multiples visualization to surface local substructures and facilitate rapid comparisons. Each domino consists of a set of interactive *neighborhood plots* and *common and unique neighbor lists*.

Neighborhood plots — side-by-side scatter plots that show the k nearest neighbors of the domino object in each model — graphically display the relationships between the object and its neighbors. These plots use the same projections as the global projection views (Section 4.2) to ensure all geometries are visualized consistently. Color encodes whether each neighbor is common to both models or unique to a single model. These neighbors are also displayed as separate scrollable lists above and below the plots and are sorted by the distance to the main object. For example, Fig. 3f shows the domino for the word *"mean"* from an embedding trained on text from the early 1800s (model A, left) to the 1990s (model B, right). To facilitate cross-model comparisons, dominoes are interactive: hovering over a neighbor in the plots or lists highlights it across the entire domino (Fig. 3g). Motivated by Design Goal 1 and our users' feedback that the most interesting insights often lie at the extremes, the default view of the Embedding Comparator lists two columns of dominoes (shown in Fig. 4): the first displays dominoes of the least similar objects (in increasing order) and the second displays those of the most similar objects (in decreasing order). To increase information scent [8], we adapt the Scented Widgets technique [74] and augment the scroll bars with a domino outline (Fig. 3i) that shows domino objects and their similarity scores.

The dominoes' information-dense display is designed to facilitate rapid acquisition of neighborhood-level insights. By scanning down the dominoes, users see not only the geometries involved but also specific common and unique neighbors to trigger hypothesis generation. Previous iterations of the Embedding Comparator used separate tabular lists to display the most and least similar objects across models (and common and unique neighbors for individual objects) but, in early user tests, we found that such a presentation produced a high cognitive load as users tried to map back and forth between the various lists. Thus, with the domino design, we encapsulate all local neighborhood information associated with a given embedded object into a single interface element while still supporting cross-model comparisons. For instance, in the *"mean"* domino in Fig. 3f, the neighbor plots reveal substructures within the local neighborhoods — in model B, the bottom appears to relate to the mathematical notion of *"mean"*, while the top is more synonymous with *"convey"* and shares neighbors with model A. We confirm this hypothesis by scanning the common and unique lists, where we observe more mathematical words under model B than A.

4.4 Linking Global and Local Views

Linking the global and local views (Design Goal 4) enables users to rapidly iterate between comparing the overall embedding spaces and inspecting individual objects. When hovering over a domino, the object and its local neighborhoods are highlighted in each of the respective global projections with the purple/green color encoding preserved (Fig. 3h). This interaction allows users to contextualize local neighborhoods within the overall embedding space. Similarly, interactive selections (e.g., lassoing or brushing) in the global projections or similarity distribution filter the dominoes, allowing users to drill down and investigate objects of interest.

5 EVALUATION: CASE STUDIES

We illustrate how the Embedding Comparator accelerates real-world workflows through case studies drawn from our formative interviews (see Appendix A for additional examples and Appendix B for implementation details).

5.1 Transfer Learning for Sentiment Classification

Our formative interviews reveal that users employ transfer learning to improve performance on downstream tasks [28, 45], but existing tools make it difficult to compare the trade-off between pre-trained embeddings' generalizability and fine-tuned embeddings' domain-specific expressiveness. To evaluate the Embedding Comparator on this task, we develop a transfer learning case study in collaboration with an ML researcher (model-driven user). Here, the researcher trains an LSTM network [24] to predict whether movie reviews express positive or negative sentiment. The researcher initializes the network using pre-trained fastText English word embeddings [48] and refines them using limited movie review data [46]. Once complete, the researcher wishes to investigate the effect of the refinement process: for example, identifying words that are ordinarily synonyms but not in the context of sentiment prediction or words that are not ordinarily synonyms but become interchangeable in the context of sentiment prediction.

We use the Embedding Comparator to compare the pre-trained fastText word embeddings with those fine-tuned for the sentiment analysis task. Our system immediately surfaces many insights about how the embedding space has changed as a result of fine-tuning. The global projection plots and similarity color encoding reveal words that changed the most due to fine-tuning have moved toward the outer regions of the fine-tuned model's embedding space (Fig. 4A). By hovering over and zooming into the fine-tuned model's global projection, the researcher finds positive sentiment words (e.g., *"favorites"*, *"gem"*, *"finest"*) moved toward the left side of the projection, negative sentiment words (e.g., *"worst"*, *"insult"*, *"forgetable"*) moved toward the right, and neutral words that did not change as a result of fine-tuning (e.g., *"part"*, *"lights"*, *"get"*) are in the center. These interactive global views enable the researcher to identify how fine-tuning the embeddings for this classification task has affected the global arrangement of objects in the vector space: unlike the pre-trained model that disperses sentiment throughout the global space, the fine-tuned model has specifically structured the embedding space based on sentiment.

Given the significant impact fine-tuning had on the global embedding space, the researcher is now interested in how the local neighborhoods have changed. Since inspection of the global projection plot revealed negative sentiment words on the right for the fine-tuned model, the researcher brushes the right side of the global projection plot, and the Embedding Comparator populates the dominoes with only those selected words (Fig. 4B). By inspecting local neighborhood dominoes, the researcher can understand how fine-tuning for sentiment analysis has affected the semantic meaning of

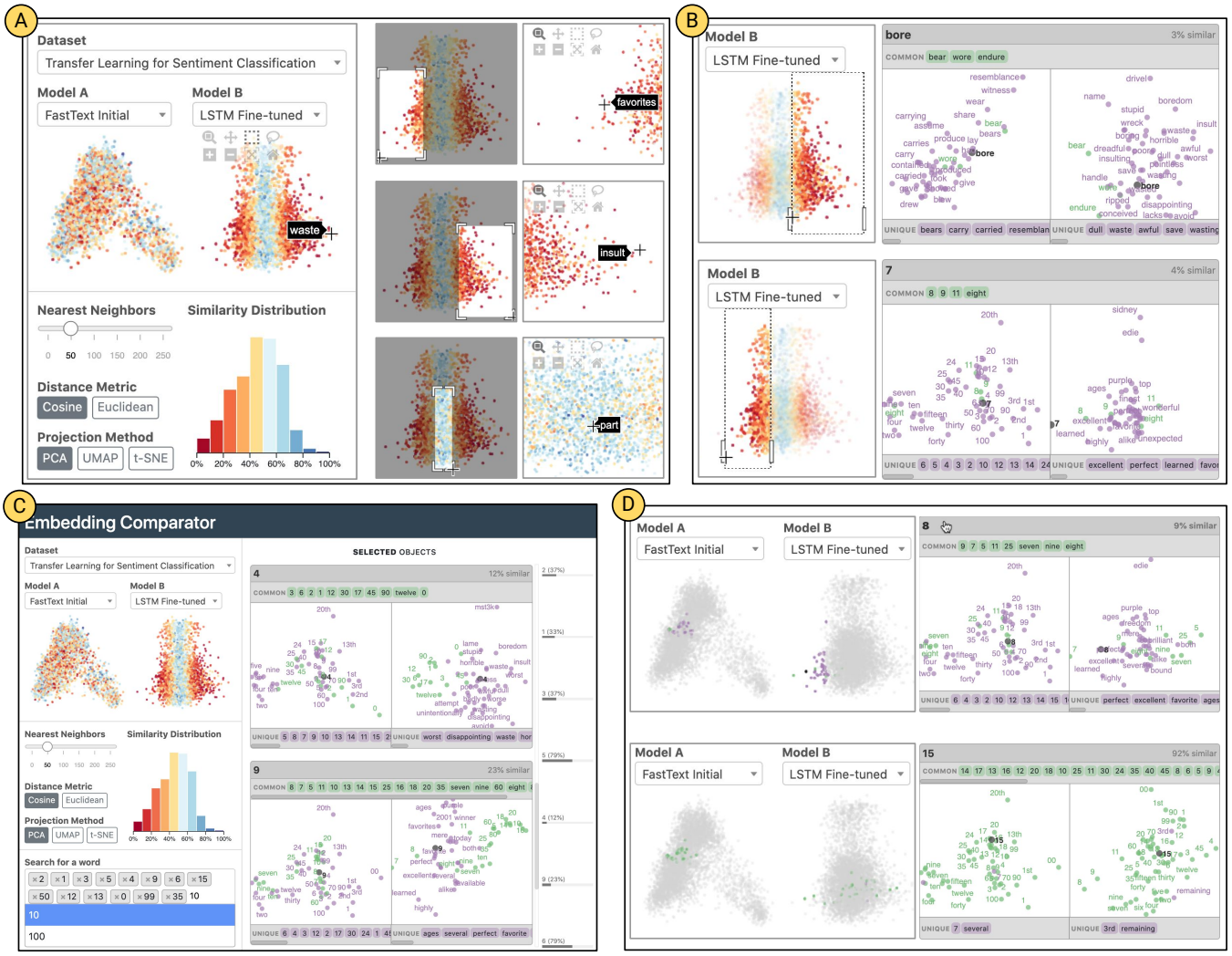

Figure 4: The Embedding Comparator applied to case study *Transfer Learning for Sentiment Classification* compares a word embedding model before and after fine-tuning for sentiment analysis. (A) The global projection plots reveal differences in the structure of the embedding space, such as the fine-tuned model separating negative and positive words. (B) Selecting regions of the global projection plots filters the dominoes, exposing neutral words that have taken on a sentiment meaning as a result of fine-tuning (e.g., *"bore"*). (C) Searching for particular words, such as numbers, shows that numbers in the range 1–10 have shifted in meaning from digits to numerical scoring. (D) Linking back to the global plots shows that this change manifests globally — while numbers cluster together in the pre-trained embeddings, they are separated in the fine-tuned embeddings.

specific objects. For example, in the fastText space *"bore"* is most closely related to *"bears"* and *"carry"* but, after fine-tuning, it is most closely related to *"dull"* and *"waste"*, indicating fine-tuning has had the intended effect. Next, the researcher analyzes the positive sentiment words by brushing the left side of the global projection plot. The resultant dominoes reveal that numbers less than 10 have also been affected by fine-tuning. For example, *"7"* has become more closely related to positive adjectives as a result of numeric scales used to rank movies within the reviews. Thus, by filtering the local neighborhood dominoes via the global projections, the Embedding Comparator exposes the types of words most affected by fine-tuning.

Intrigued by the number *"7"* result, the researcher wants to understand how many other numbers have changed as a result of fine-tuning, and uses the search functionality to analyze a variety of numbers in the range 1–100 (Fig. 4C). Glancing at the similarity scores in the domino outline shows which numbers have changed due to fine-tuning. Numbers typically used to convey sentiment, like 100, or those rarely used to convey sentiment, like 15, have not changed. However, numbers that previously did not convey sentiment but do in movie review data (2–4 and 6–9) have changed

significantly. The researcher hovers over each domino to locate each number in the global projection plots and confirms the model has learned the expected sentiment for each number (Fig. 4D). In the fastText projection, all numbers cluster together, consistent with general language usage. In the fine-tuned projection, 2–4 are on the negative sentiment (right) side, 6–9 are on the positive (left) side, and other numbers are in the center, consistent with the colloquial 1–10 movie review rating system.

The researcher we worked with on this case study was excited about the results the Embedding Comparator helped reveal. Generating these types of insights with prior tools would require significant tedious effort — besides manually constructing the necessary views (e.g., within a Jupyter notebook), the researcher would have needed to formulate hypotheses *a priori* about which specific words to investigate. In contrast, by calculating a similarity score for every word, and by visualizing local neighborhoods as dominoes, the Embedding Comparator surfaces this information more directly. As a result, the system transforms the process of comparing embedding models from requiring explicit and manual steering by the researcher, towards more of a *browsing* experience. This shift frees the researcher to focus on generating and answering hypotheses about their models.

5.2 Language Evolution via Diachronic Word Embeddings

Our second case study follows a linguist (a data-driven user) who employs embedding models to study the evolution of languages over time. Word embeddings can capture diachronic changes in language (i.e., changes over time) [21]. Here, we use diachronic word embeddings from HistWords [21], trained on English books written from 1800–2000 grouped by decade. We select embeddings from five decades (1800–1810, 1850–1860, 1900–1910, 1950–1960, and 1990–2000) and evaluate how the Embedding Comparator surfaces words whose meanings have evolved.

Fig. 5A shows the Embedding Comparator comparing embeddings of text written in 1900–1910 to text written in 1990–2000. By scrolling through the local neighborhood dominoes, the linguist can immediately replicate known insights [21], such as the change in the meaning of *"gay"* (from "happy" to "homosexual") and *"major"* (from "military" to "important") over the century. The Embedding Comparator also reveals words such as *"aids"*, whose meaning changed from "assists" to the disease HIV/AIDS, which scientists did not name until the early 1980s [17]. In contrast to the original analysis [21], with the Embedding Comparator, there is no need to align the various embedding spaces manually, nor do users need to define and compute a task-specific semantic displacement metric to uncover these findings. Our method for comparing embedded objects through reciprocal local neighborhoods is agnostic to application domains and tasks. As a result, the Embedding Comparator scaffolds and accelerates the analysis process for users regardless of their ML expertise. Novice users need minimal technical knowledge to replicate established linguistic analysis, while experts can devote their effort to designing task-specific metrics only when necessary.

To further compare the two models, the linguist varies the number of neighbors k using the nearest neighbors slider (Fig. 5B). Smaller values of k enable comparisons of how immediate neighbors of a word have changed over time. By dragging the slider to decrease k (e.g., $k = 12$), the linguist discovers words that have high similarity at small values of k, such as *"que"*, which has a small set of French words as its nearest neighbors in both models. This finding may be surprising to the linguist, given that the models were trained on English text. Greater values of k enable comparisons of larger neighborhoods. Words with high similarity at larger k have large neighborhoods that have remained consistent over time. Increasing k (e.g., to $k = 100$) reveals words such as days and months (e.g., *"april"* and *"tuesday"*) whose meanings have remained consistent between the time periods. This interaction enables the user to understand the meaning captured in the embedding spaces, particularly the size and position of different clusters.

The Embedding Comparator enables the linguist to easily compare alternate models, which is valuable in this case study since we have models trained on many decades. For example, using the drop-down menus, the linguist can compare 1800–1810 to 1900–1910. In doing so, the linguist finds that during the 19th century *"nice"* moves away from meaning "refined" and "subtle" and moves toward "pleasant", in line with previous findings [21]. Similarly, by fixing one model to the most recent decade (1990–2000) and varying the other, the user can look at the similarity distribution to reveal the pairwise diachronic changes. Immediately, the user observes that the similarity distribution is centered at 40% and is surprised by the degree of dissimilarity of language between the 1950s and 1990s, only a 40 year period. However, inspecting the dominoes reveals computer-related words like *"artificial"* (Fig. 1) and *"file"* have changed the most, referencing the significant technological shift over that period. Decreasing the decade continues to shift the similarity distribution leftward, suggesting language usage diverges as time diverges (Fig. 5C). Finally, comparing the earliest decade (1800–1810) to the most recent (1990–2000) has a distribution centered at 18% and very few words greater than 50% similar. Selecting the range greater than 50% by brushing on the similarity histogram (Fig. 5D) surfaces only numbers, indicating word usage changed significantly more than number usage over the past two centuries. These interactions provide evidence for the linguist to understand how language has continuously changed over two centuries.

5.3 Multimodal Emoji Representations

Our final case study demonstrates the Embedding Comparator in a non-NLP setting: comparing emoji representations. Since many of our formative users compared embeddings in computer vision and multimodal tasks, this case study compares emoji language embeddings (learned representations from textual emoji descriptions [13]) to emoji image embeddings (raw RGBA pixel values). The model-driven user hypothesizes the language embeddings arrange emojis based on semantics, and the image embeddings arrange emojis based on visual characteristics (e.g., shape, color). They are curious to explore if their hypothesis is correct and, if so, to understand how visual and semantic similarities are related.

The similarity distribution shown in the initial view (Fig. 6A) immediately reveals that emojis are represented quite differently between the two models: most emojis are 0–10% similar, and no

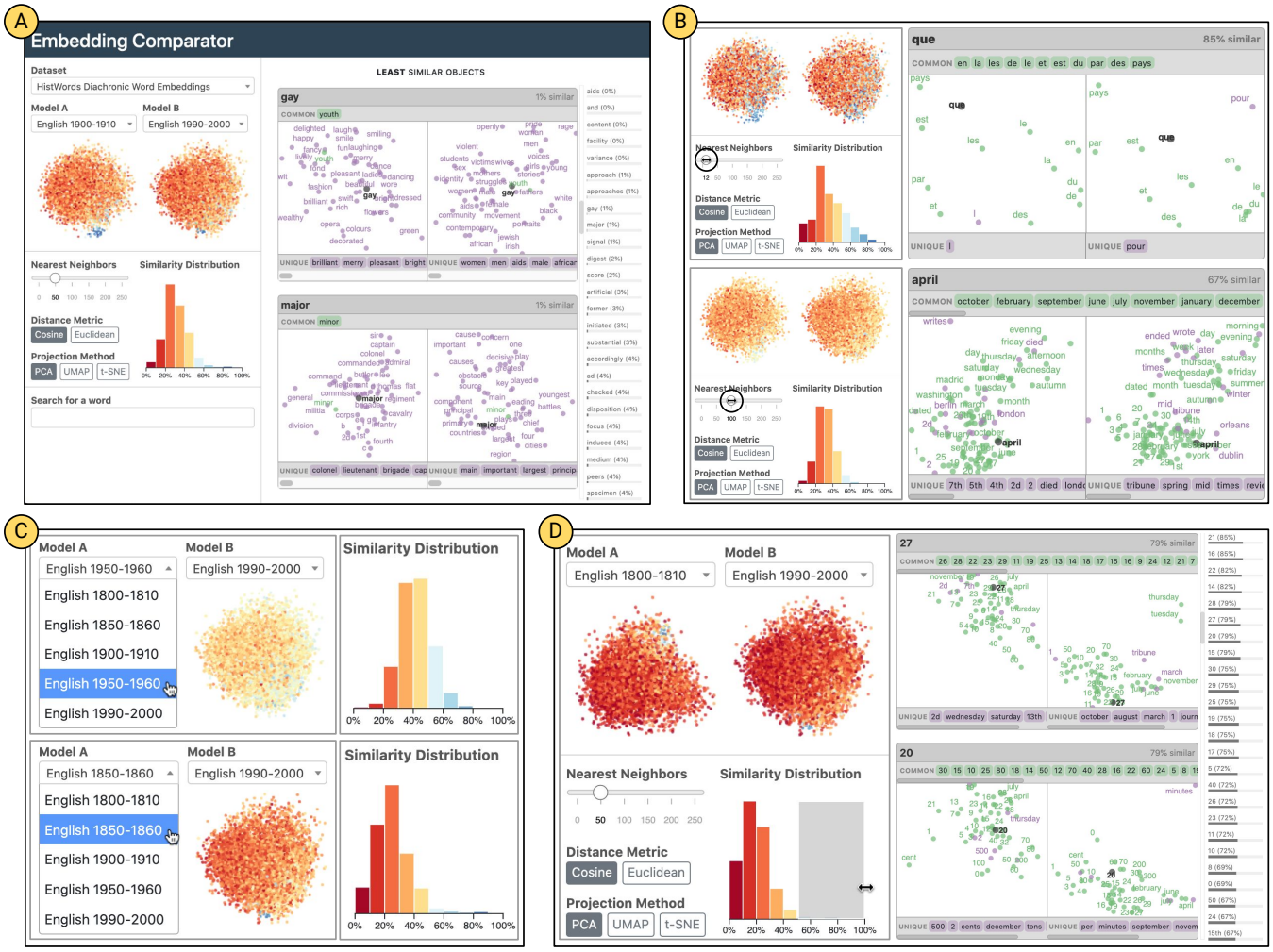

Figure 5: The Embedding Comparator applied to case study *Language Evolution via Diachronic Word Embeddings* compares word embedding models trained on literature from different decades. (A) The sorted domino lists demonstrate how English has changed over time, e.g., the word *"gay"* has changed in meaning from "happy" to "homosexual". (B) Varying the k-nearest neighbors slider identifies neighborhoods within the embeddings, such as a small cluster of French words and a larger neighborhood of months. (C) Selecting other decades exposes that language similarity decreases as the time between decades increases. (D) Comparing 1800 to 2000 and brushing the right side of the similarity distribution reveals that number usage has remained constant over two centuries.

emojis are more than 40% similar, suggesting visual and semantic similarity are often disjoint. By glancing at the domino outline, the user can identify the emojis that differ the most and least between models. While the most similar emojis are face emojis (i.e., smiling face, shocked face), the least similar emojis are more diverse objects (i.e., balloon, alligator). This finding supports the user's hypothesis because face emojis are similar in both meaning and shape/color, whereas a least similar emoji, like balloon, does not necessarily look like its semantically similar emojis.

Digging deeper, the user looks at the baseball emoji in the least similar list (Fig. 6A). The domino for the baseball emoji shows that the nearest neighbors in the language model are other sports-related emojis (e.g., soccer ball, football), while the neighbors in the image model are emojis with similar color/shape (e.g., speech bubble, rice bowl). The only shared neighbor between the two models is a soccer ball because it is white, round, and sports-related. Scrolling through the sorted unique neighbors lists at the bottom of the domino lets the user see how neighbors diverge as they become more distant from the baseball. For example, in the language model, more distant neighbors are less semantically similar — the first neighbor is a golf flag, the 13th neighbor is a dartboard, and the 50th neighbor is a gaming controller. Similarly, in the image model, more distant neighbors are less visually similar — the first neighbor is a white circle, while the 17th neighbor is a birthday cake, and the 50th neighbor is a cow. The domino visualization and interactions enable

Figure 6: The Embedding Comparator applied to case study: *Multi-modal Emoji Representations*, compares language embeddings learned from textual descriptions of emojis to image embeddings comprising the emojis' raw pixel values. The language model captures the semantic similarity of emojis, while the image model captures visual similarities. (A) For example, the language embeddings position the baseball emoji with other sports, whereas the image embeddings position it with other white round objects. (B) Hovering on dominoes highlights their position in the global projection plot, revealing a tight cluster of faces in the image model and a more dispersed cluster of medical emojis in the language model. (C) Looking closely at local neighborhoods exposes local clusters, such as face and animal clusters in the language embeddings or cat face and human face clusters in the image embeddings.

the user to confirm that the embedding spaces have captured their expected representations.

Given that distance between emojis corresponded to distance in meaning for the baseball emoji, the user is interested in understanding how the embedding spaces have captured meaning. By studying other local neighborhood dominoes of the least similar objects, the user finds that the running shoe, ATM, and masked face emojis in the image model share no semantic similarity to their nearest neighbors but are similar in shape and color. In contrast, these objects are semantically similar to their nearest neighbors in the language model. By analyzing the masked face emoji more closely, the user finds that its nearest neighbors in the language model are related to health (e.g., hospital, apple, ambulance), and in the image model are other faces (crying face, smiling face) (Fig. 6B). Hovering over the masked face domino highlights each neighborhood in the global projections. This interaction reveals that the faces in the image embedding space form a tighter cluster than the health-related emojis in the language embedding space. It makes sense that the faces form a tight cluster because faces are very similar images, whereas the health-related emojis are related to various other contexts (e.g., ambulances with cars, apples with food). To confirm this finding is not an artifact of the projection method, the user switches between PCA, UMAP, and t-SNE, finding the cluster pattern persists under each technique. Through these interactions, the user is confident that the language embeddings capture semantics learned from the emoji textual descriptions, and the image representations only capture visual similarities of the emoji images.

Next, the user shifts to scroll through the most similar objects and finds the grinning cat emoji. The local neighborhood plots within this domino reveal two distinct clusters of emojis in the language model (happy face emojis and animal emojis) and two clusters in the image model (face emojis and cat emojis) (Fig. 6C). By slowly reducing the neighborhood size (e.g., to 20 and 10), the animal emoji cluster in the language model slowly disappears, indicating the face emojis are more similar to the grinning cat than other animal emojis. The user can change from PCA to t-SNE or UMAP to verify other projection methods also capture this difference. Interactively hovering over the common objects of the domino and looking at the neighbor's location in the neighborhood plots further reveals that all common neighbors of the grinning cat (a positive sentiment emoji) are found solely in the happy face cluster in the language model. In contrast, the common neighbors are found in both of image model's clusters, depending on whether the neighbor is a cat or a face. Together with the user's previous findings, interaction with the dominoes further uncovers how the language embeddings model emoji meaning, and the image embeddings model emoji appearance.

6 EVALUATION: FIRST-USE STUDIES

The Embedding Comparator was designed to help users efficiently analyze the similarities and differences between embedding spaces. We performed first-use studies to evaluate its effectiveness. To simulate users' typical processes (as reported in our formative interviews Section 3), we had participants compare two embedding spaces using their standard method in a Jupyter Notebook [33] and with the Embedding Comparator. In particular, users compared

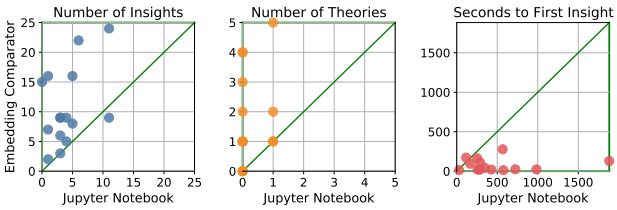

Figure 7: Using the Embedding Comparator, participants gained more insights and theories and reduced their time to first insight.

HistWords [21] embeddings from 1800–1810 to embeddings from 1990–2000 (Section 5.2). We chose HistWords embeddings because they model English language over time, so no task-specific knowledge was required. We recruited participants via an open call within our organization. We performed first-use studies via video chat with 15 representative users: six computer science graduate students, one psychology graduate student, one computational biology graduate student, four ML engineers, and three post-docs. To ensure representative findings, we recruited users who had experience analyzing and comparing embeddings and self-reported having researched or used embeddings in their work.

Each first-use study lasted 60–90 minutes, and we compensated participants with $20 Amazon gift cards. Participants spent half of the study using the Embedding Comparator and half using the Jupyter Notebook. We randomized the starting interface: seven participants started with the Embedding Comparator, and eight participants started with the Jupyter Notebook. We began each condition with an introduction to the embeddings and prompted the participants to determine the similarities and differences of the two embeddings spaces. To enable a fair comparison, in the Jupyter Notebook, we provided code to plot dimensionality reduction projections of the embeddings and print the nearest neighbors of an object. Further, we encouraged participants to use any tools — including their code or online GUIs — that would help them analyze the embeddings. We asked participants to think aloud throughout the study and, at the end of each condition, to enumerate the similarities and differences between the two embedding spaces and describe what they would do if they had access to the original models and data.

6.1 Quantitative Results

We reviewed recordings of each study session and measured the *insights* and *theories* our users identified using each interface. We define an *insight* as the user indicating that an embedded object, or type of embedded object, is represented similarly or differently in the two embedding spaces in a meaningful way (e.g., "*gay*" moving in meaning from "happy" to "homosexual" or an embedding space projection showing meaningful clusters). We say the user had a *theory* if they generated a reason for an insight or expressed a desire to look at the original data or models to understand an insight.

We measure the number of insights, number of theories, and time to first insight (measured from the end of our introduction) for both conditions and show participant-level results in Fig. 7. We compute statistical significance using a single-tailed paired

sample t-test. Users developed their first insight faster ($p = 0.003$) using the Embedding Comparator (μ = 1:16 min, σ = 1:20 min) as compared to the Jupyter Notebook (μ = 8:13 min, σ = 7:54 min). Users also developed more insights ($p = 0.004$) using the Embedding Comparator ($\mu = 10.7$, $\sigma = 6.6$) as compared to the Jupyter Notebook ($\mu = 4.1$, $\sigma = 3.3$), and users generated more theories ($p = 0.001$) using the Embedding Comparator ($\mu = 1.7$, $\sigma = 1.6$) as compared to the Jupyter Notebook ($\mu = 0.3$, $\sigma = 0.5$).

6.2 Qualitative Results

In the Jupyter notebook condition, despite having access to starter code and being encouraged to use their own code or outside resources, participants struggled to generate meaningful hypotheses that resulted in insights about the embedding spaces. For example, a common workflow used by nine participants involved generating words they expected to differ between the two spaces and comparing their nearest neighbors. This process caused a user's biases to drive their exploration (e.g., *"I am looking for a word with two meanings where the meanings are very different"* [P6]; or, *"because I am Arab, I want to look at [the word] Arab"* [P1]). As a result, users rarely uncovered unexpected results, and they were often frustrated because they had to rely on their intuition — as P1 said, *"I don't have a sophisticated way of looking through this. Some 1800s and 1900s history knowledge would be useful."* Another typical workflow was to evaluate global differences in the embedded representations through summary statistics [P2, P4, P8, P9, P11, P12], cluster analysis [P2, P4, P5, P6], or object ranking [P2, P9, P15]. While these analyses provided insight, they required participants to carefully design experiments that were often tedious to implement. Using the expressiveness of the Jupyter Notebook, users were able to run experiments that the Embedding Comparator does not support (e.g., computing the average spread of a cluster [P4]). However, after using the Embedding Comparator, participants expressed that even though it used different mechanisms to compare the embedding spaces, it often provided them the insights they had been trying to surface in the Jupyter Notebook. For example, P15 said, *"I was trying to do something similar [in the Jupyter Notebook]"*, and P8 expressed, *"This was precisely what I was trying to articulate with the [Jupyter Notebook]."*

When using the Embedding Comparator, users generated insights quickly by looking at the most and least similar lists. Common insights included the meaning of numbers and religious words (e.g., "*catholic*" and "*god*") staying constant over time (P1, P2, P5, P7–P13); "*aids*" changing in meaning from "help" to "HIV" (P1, P3, P6, P8, P11–P13); and, "*logic*", "*calculated*", and "*volume*" shifting towards mathematical denotations (P1, P3, P5, P9, P10, P12, P13). Using the Embedding Comparator, users were able to gain insights into global semantic differences between the embedding spaces. For instance, P9 found *"words that are commonly used without significant changes to their meaning over time are similar between the two embeddings, whereas other words that have changed their usage over time or correspond to a new invention, like the car, are of course dissimilar in the two embeddings. I did not find this in my previous analysis."*

Users found the dominoes invaluable in their exploration, saying *"it's hard to compress all this information in a Jupyter Notebook, so it's nice to have a tool to be able to browse a lot of words at once"* (P5), and *"I am able to see [objects] in a graphical view which helps me see [similarities and differences]"* (P3). Using the dominoes, users were able to analyze many objects at a time and speed up their analysis. For instance, P13 found dominoes allowed them to *"get a quick sense of what words distinguish one [embedding space] from the other"* and P1 articulated, *"[the dominoes] allowed me to get a better idea of things because I got more exposure to more words faster"*.

Our first-use studies also validate our local neighborhood similarity metric and the use of this metric to drive our system by sorting dominoes. Interestingly, without prior knowledge of the Embedding Comparator, one participant implemented a metric similar to ours for computing the local neighborhood intersection during their Jupyter exploration, suggesting local neighborhood similarity is intuitive to users. However, their implementation was not optimized and did not scale to compute similarity for a sample of more than 40 words, leading to time spent waiting for code to run and uneasiness debugging. Since the Embedding Comparator calculates similarity for all embedded objects automatically when embeddings are loaded, it shifts users to immediately focus on meaningful parts of the analysis process as opposed to tedious implementation.

Laying out the dominoes ordered by similarity aided users in understanding the similarities and differences between the embedding spaces. P1 attributed seeing *"a lot more words and what is most similar and least similar"* as the reason why they generated insights faster using the Embedding Comparator and compared the two interfaces by saying *"in the Notebook words were randomly chosen, but here they are sorted, which lets you see things in a more organized way and lets words stick out."* P1 found it to be invaluable when generating insights, saying the *"main thing that was useful was having [the Embedding Comparator] order everything, being able to look at the different parts of the distribution, and seeing [objects] in order - the least similar and the most similar."* Similarly, P5 reported *"seeing the differences in sets in the nearest neighborhoods is a much easier way to compare the sets of embeddings"* and P2 summarized their experience by saying, *"[the Embedding Comparator] presented the information in a better way and just by doing that we have some sort of a breakthrough."*

After analyzing the dominoes in the default view of the Embedding Comparator, users varied the parameter controls to further compare models and test hypotheses. Users often brushed over the similarity distribution to analyze words at a specific similarity value (P1, P5, P7–P15), used the search bar to analyze words that had come to mind during their analysis (P2, P3, P5–P15), and varied k to explore the hierarchy of a word's local neighborhood (P1, P2, P6, P7, P9–P15). For example, after discovering the neighbors of "*content*" were related to "satisfied" in one model and "substances" in the other, P10 wanted to understand "how these models represent [words] that have two meanings." They hypothesized that both models may have learned both definitions and k was simply too small to see the full neighborhood. After increasing k, P10 found the spaces were still very dissimilar, leading them to theorize one dataset may have contained more scientific text than the other.

When using the Embedding Comparator, users were also able to generate numerous theories including ideas for improving the data representation (e.g., preprocessing numbers into a learned <num>

token [P5]); new experimental procedures (e.g., training embeddings on native English books vs. books translated from Chinese as a way to compare culture [P7]); reasons for representation failures (e.g., word frequency affects semantic stability over time [P1, P9, P10, P14], or words used in many contexts are susceptible to being represented differently between the two embedding spaces [P2, P10, P12, P13]); and, questions about the original data (*what is the distribution of genres included in the datasets?* [P5–P7, P9, P10, P12] and *what was the data preprocessing pipeline?* [P3, P12]). After using both interfaces, P5 noted the Embedding Comparator *"made it very clear that they were differences in the embedding spaces"* as compared to the Jupyter Notebook where *"it is a lot harder to find the differences."* As P4 said, using the Embedding Comparator *"would be very helpful to my research."*

7 DISCUSSION AND FUTURE WORK

We present the Embedding Comparator, a novel interactive system for comparing embedding spaces. Informed by formative interviews conducted with embedding users across disciplines, our design balances between visualizing information about the overall embedding spaces and enabling exploration of local neighborhoods. To directly surface similarities and differences, a similarity score is computed for every embedded object and encoded across global and local views. To facilitate rapid comparisons, we introduce *local neighborhood dominoes*: small multiple visualizations of local geometries and lists of common and unique objects. Interactively linking visualizations of the global embedding space and dominoes permits a tight iterative loop enabling users to alternate between exploring the global space and diving into specific objects of interest. Through case studies of tasks described by our formative interviewees, we demonstrate how the Embedding Comparator transforms the analysis process from requiring tedious and error-prone manual specification to browsing and interacting with graphical displays. Moreover, by computing a similarity score for each object, and using it to drive the various views, the Embedding Comparator immediately surfaces interesting insights, and published domain-specific results can be replicated with only a handful of interactions. High-dimensional datasets also often occur outside of ML, such as from biological gene and protein expression assays used to study cancer [12] and sensor data used to assess air quality [14]. While the case studies presented in our paper are focused on real-world use cases of embeddings in ML described by our interviewees, the Embedding Comparator can be used to compare high-dimensional datasets from other sources.

An important component of our system design was computing a similarity metric for each object across both embedding spaces. In our interviews, we found users were concerned they may miss unexpected insights because their existing comparison workflows often rely upon ad hoc object selection strategies. Thus, we chose this metric to systematically prioritize objects that are the most and least similar between the embedding spaces (Design Goal 1), and we encode the similarity scores throughout the global and local views. While existing systems often require task-specific metrics or alignment of embedding spaces (Section 2), our metric is agnostic to domain and model and is applicable even when the two embedding spaces stem from different modalities (e.g., natural language and images in Section 5.3). In practice, our system surfaced known linguistic insights without requiring task-specific metrics or embedding alignment, while being more extensible to additional tasks described by our interviewees. We expect that generalizable metrics computed on similarity of local neighborhoods can likewise scaffold future embedding interpretability systems as we demonstrated with the Embedding Comparator.

Another critical aspect of our design was displaying embedded objects as small multiples of local neighborhood dominoes. While small multiples are a well-understood visualization technique, they remain relatively under-utilized in interpretability systems which largely focus on deeply exploring one input instance at a time. In contrast, our design is motivated by insights from visualization recommender systems [75, 76], which have promoted breadth-first exploration of data by adapting Edward Tufte's maxim of prioritizing data variation over design variation [66]. However, a naive application of the small multiples technique can yield an overwhelming experience which burdens users with knowing which small multiple to attend to. Thus, to bootstrap the exploration process, the Embedding Comparator populates its initial view with two lists of small multiples covering the least similar and most similar objects — the objects our formative interviewees often described starting their process by examining. Future work might consider further iterating on our design by incorporating small multiple summarization techniques such as interactive piling [37].

The design of the Embedding Comparator suggests several other avenues for compelling future work. For example, we applied the Embedding Comparator to compare latent embeddings of chemical molecules [61] (Appendix A.2). Chemical molecules can be represented as textual SMILES strings or as 2-D structural diagrams. While the 2-D rendering produces readable dominoes, the length of the SMILES strings limits rapid analysis of the dominoes. We designed dominoes for concise visual representations of embedded objects; thus, one direction for future work is extracting concise representations from long objects and extending comparison visualizations for long representations. For instance, for sentences or document embeddings [36], it would be interesting to explore techniques for identifying specific components of each object (e.g., a subset of words in a document) responsible for the differences in the embeddings and visually display these via dominoes. Further, in cases where many objects are highly dissimilar (as seen in the chemical molecules example), our LNS metric cannot prioritize specific examples that differ most between the two models. Future work can explore alternative sorting criteria or enable users to supply domain-specific metrics in these situations. Finally, our design goals suggest future work visualizing n-way model comparisons to accelerate workflows comparing many embedding models and incorporating training data to contextualize an object's similarities or differences across the models.

ACKNOWLEDGMENTS

This work is supported by a grant from the MIT-IBM Watson AI Lab. Research was also sponsored by the United States Air Force Research Laboratory and the United States Air Force Artificial Intelligence Accelerator and was accomplished under Cooperative Agreement Number FA8750-19-2-1000. The views and conclusions

contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the United States Air Force or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein. We also thank Jonas Mueller, Hendrik Strobelt, and Alan Lundgard for helpful feedback.

REFERENCES

[1] Carl Allen and Timothy M. Hospedales. 2019. Analogies Explained: Towards Understanding Word Embeddings. In *Proceedings of the International Conference on Machine Learning (ICML)*, Vol. 97. PMLR, Long Beach, USA, 223–231.

[2] Dustin L. Arendt, Nasheen Nur, Zhuanyi Huang, Gabriel Fair, and Wenwen Dou. 2020. Parallel Embeddings: a Visualization Technique for Contrasting Learned Representations. In *Proceedings of the International Conference on Intelligent User Interfaces (IUI)*. ACM, Cagliari, Italy, 259–274.

[3] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. 2017. Network Dissection: Quantifying Interpretability of Deep Visual Representations. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Honolulu, USA, 3319–3327.

[4] David Bau, Jun-Yan Zhu, Hendrik Strobelt, Bolei Zhou, Joshua B. Tenenbaum, William T. Freeman, and Antonio Torralba. 2019. GAN Dissection: Visualizing and Understanding Generative Adversarial Networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*. OpenReview.net, New Orleans, USA.

[5] Samy Bengio and Georg Heigold. 2014. Word Embeddings for Speech Recognition. In *Proceedings of the Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA, Singapore, 1053–1057.

[6] Tristan Bepler and Bonnie Berger. 2019. Learning protein sequence embeddings using information from structure. In *Proceedings of the International Conference on Learning Representations (ICLR)*. OpenReview.net, New Orleans, USA.

[7] Maxwell L Bileschi, David Belanger, Drew H Bryant, Theo Sanderson, Brandon Carter, D Sculley, Alex Bateman, Mark A DePristo, and Lucy J Colwell. 2022. Using deep learning to annotate the protein universe. *Nature Biotechnology* (2022).

[8] John Carroll and Peter Pirolli. 2003. *HCI Models, Theories and Frameworks*. Morgan Kaufmann, San Francisco, CA, Chapter Exploring and Finding Information, 157–191.

[9] Brandon Carter, Jonas Mueller, Siddhartha Jain, and David K. Gifford. 2019. What made you do this? Understanding black-box decisions with sufficient input subsets. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*. PMLR, Naha, Japan, 567–576.

[10] Juntian Chen, Yubo Tao, and Hai Lin. 2018. Visual Exploration and Comparison of Word Embeddings. *Journal of Visual Languages & Computing* 48 (2018), 178–186.

[11] François Chollet et al. 2015. Keras. https://keras.io.

[12] Robert Clarke, Habtom W Ressom, Antai Wang, Jianhua Xuan, Minetta C Liu, Edmund A Gehan, and Yue Wang. 2008. The properties of high-dimensional data spaces: implications for exploring gene and protein expression data. *Nature Reviews Cancer* 8, 1 (2008), 37–49.

[13] Ben Eisner, Tim Rocktäschel, Isabelle Augenstein, Matko Bosnjak, and Sebastian Riedel. 2016. emoji2vec: Learning Emoji Representations from their Description. In *Proceedings of the International Workshop on Natural Language Processing for Social Media (SocialNLP@EMNLP)*. ACL, Austin, USA, 48–54.

[14] Daniel Engel, Klaus Greff, Christoph Garth, Keith Bein, Anthony S. Wexler, Bernd Hamann, and Hans Hagen. 2012. Visual Steering and Verification of Mass Spectrometry Data Factorization in Air Quality Research. *IEEE Transactions on Visualization and Computer Graphics* 18, 12 (2012), 2275–2284.

[15] Jesse H. Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Mohammad Norouzi, Douglas Eck, and Karen Simonyan. 2017. Neural Audio Synthesis of Musical Notes with WaveNet Autoencoders. In *Proceedings of the International Conference on Machine Learning (ICML)*, Vol. 70. PMLR, Sydney, Australia, 1068–1077.

[16] Kawin Ethayarajh, David Duvenaud, and Graeme Hirst. 2019. Towards Understanding Linear Word Analogies. In *Proceedings of the Conference of the Association for Computational Linguistics*. ACL, Florence, Italy, 3253–3262.

[17] Centers for Disease Control (CDC) et al. 1982. Update on acquired immune deficiency syndrome (AIDS)–United States. *Morbidity and Mortality Weekly Report (MMWR)* 31, 37 (1982), 507.

[18] Michael Gleicher, Aditya Barve, Xinyi Yu, and Florian Heimerl. 2020. Boxer: Interactive Comparison of Classifier Results. *Computer Graphics Forum* 39, 3 (2020), 181–193.

[19] David Ha and Douglas Eck. 2018. A Neural Representation of Sketch Drawings. In *Proceedings of the International Conference on Learning Representations (ICLR)*. OpenReview.net, Vancouver, Canada.

[20] William L. Hamilton, Jure Leskovec, and Dan Jurafsky. 2016. Cultural Shift or Linguistic Drift? Comparing Two Computational Measures of Semantic Change. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. ACL, Austin, USA, 2116–2121.

[21] William L. Hamilton, Jure Leskovec, and Dan Jurafsky. 2016. Diachronic Word Embeddings Reveal Statistical Laws of Semantic Change. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*. ACL, Berlin, Germany.

[22] Florian Heimerl and Michael Gleicher. 2018. Interactive Analysis of Word Vector Embeddings. *Computer Graphics Forum* 37, 3 (2018), 253–265.

[23] Florian Heimerl, Christoph Kralj, Torsten Moller, and Michael Gleicher. 2020. embComp: Visual Interactive Comparison of Vector Embeddings. *IEEE Transactions on Visualization and Computer Graphics* (2020), 1–1.

[24] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (1997), 1735–1780.

[25] Fred Hohman, Andrew Head, Rich Caruana, Robert DeLine, and Steven Mark Drucker. 2019. Gamut: A Design Probe to Understand How Data Scientists Understand Machine Learning Models. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI)*. ACM, Glasgow, Scotland, 579.

[26] Fred Hohman, Minsuk Kahng, Robert Pienta, and Duen Horng Chau. 2019. Visual Analytics in Deep Learning: An Interrogative Survey for the Next Frontiers. *IEEE Transactions on Visualization and Computer Graphics* 25, 8 (2019), 2674–2693.

[27] Thomas Höllt, Nicola Pezzotti, Vincent van Unen, Frits Koning, Boudewijn P. F. Lelieveldt, and Anna Vilanova. 2018. CyteGuide: Visual Guidance for Hierarchical Single-Cell Analysis. *IEEE Transactions on Visualization and Computer Graphics* 24, 1 (2018), 739–748.

[28] Jeremy Howard and Sebastian Ruder. 2018. Universal Language Model Fine-tuning for Text Classification. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*. ACL, Melbourne, Australia, 328–339.

[29] Wengong Jin, Regina Barzilay, and Tommi S. Jaakkola. 2018. Junction Tree Variational Autoencoder for Molecular Graph Generation. In *Proceedings of the International Conference on Machine Learning (ICML)*, Vol. 80. PMLR, Stockholm, Sweden, 2328–2337.

[30] Ian T Jolliffe. 1986. Principal Components in Regression Analysis. In *Principal Component Analysis*. Springer, 129–155.

[31] Minsuk Kahng, Pierre Y. Andrews, Aditya Kalro, and Duen Horng (Polo) Chau. 2018. ActiVis: Visual Exploration of Industry-Scale Deep Neural Network Models. *IEEE Transactions on Visualization and Computer Graphics* 24, 1 (2018), 88–97.

[32] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*. San Diego, USA.

[33] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian E. Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica B. Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, Carol Willing, and Jupyter Development Team. 2016. Jupyter Notebooks — a publishing format for reproducible computational workflows. In *Proceedings of the International Conference on Electronic Publishing (ElPub)*. IOS Press, Göttingen, Germany, 87–90.

[34] Yehuda Koren, Robert M. Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (2009), 30–37.

[35] Matt J. Kusner, Brooks Paige, and José Miguel Hernández-Lobato. 2017. Grammar Variational Autoencoder. In *Proceedings of the International Conference on Machine Learning (ICML)*, Vol. 70. PMLR, Sydney, Australia, 1945–1954.

[36] Quoc V. Le and Tomás Mikolov. 2014. Distributed Representations of Sentences and Documents. In *Proceedings of the International Conference on Machine Learning (ICML)*, Vol. 32. JMLR, Beijing, China, 1188–1196.

[37] Fritz Lekschas, Xinyi Zhou, Wei Chen, Nils Gehlenborg, Benjamin Bach, and Hanspeter Pfister. 2021. A Generic Framework and Library for Exploration of Small Multiples through Interactive Piling. *IEEE Transactions on Visualization and Computer Graphics* 27, 2 (2021), 358–368.

[38] Quan Li, Kristanto Sean Njotoprawiro, Hammad Haleem, Qiaoan Chen, Chris Yi, and Xiaojuan Ma. 2018. EmbeddingVis: A Visual Analytics Approach to Comparative Network Embedding Inspection. In *Proceedings of the Conference on Visual Analytics Science and Technology (VAST)*. IEEE, Berlin, Germany, 48–59.

[39] Yixuan Li, Jason Yosinski, Jeff Clune, Hod Lipson, and John E. Hopcroft. 2016. Convergent Learning: Do different neural networks learn the same representations?. In *Proceedings of the International Conference on Learning Representations (ICLR)*. San Juan, Puerto Rico.

[40] Zachary C. Lipton. 2018. The Mythos of Model Interpretability: In Machine Learning, the Concept of Interpretability is Both Important and Slippery. *Queue* 16, 3 (2018), 31–57.

[41] Shusen Liu, Peer-Timo Bremer, Jayaraman J. Thiagarajan, Vivek Srikumar, Bei Wang, Yarden Livnat, and Valerio Pascucci. 2018. Visual Exploration of Semantic Relationships in Neural Word Embeddings. *IEEE Transactions on Visualization and Computer Graphics* 24, 1 (2018), 553–562.

[42] Shusen Liu, Zhimin Li, Tao Li, Vivek Srikumar, Valerio Pascucci, and Peer-Timo Bremer. 2019. NLIZE: A Perturbation-Driven Visual Interrogation Tool for Analyzing and Interpreting Natural Language Inference Models. *IEEE Transactions*

on Visualization and Computer Graphics 25, 1 (2019), 651–660.

[43] Shusen Liu, Dan Maljovec, Bei Wang, Peer-Timo Bremer, and Valerio Pascucci. 2017. Visualizing High-Dimensional Data: Advances in the Past Decade. *IEEE Transactions on Visualization and Computer Graphics* 23, 3 (2017), 1249–1268.

[44] Yang Liu, Eunice Jun, Qisheng Li, and Jeffrey Heer. 2019. Latent Space Cartography: Visual Analysis of Vector Space Embeddings. *Computer Graphics Forum* 38, 3 (2019), 67–78.

[45] Jonathan Long, Evan Shelhamer, and Trevor Darrell. 2015. Fully Convolutional Networks for Semantic Segmentation. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Boston, USA, 3431–3440.

[46] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning Word Vectors for Sentiment Analysis. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*. ACL, Portland, USA, 142–150.

[47] Leland McInnes, John Healy, Nathaniel Saul, and Lukas Großberger. 2018. UMAP: Uniform Manifold Approximation and Projection. *Journal of Open Source Software* 3, 29 (2018), 861.

[48] Tomás Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in Pre-Training Distributed Word Representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*. ELRA, Miyazaki, Japan.

[49] Tomás Mikolov, Quoc V. Le, and Ilya Sutskever. 2013. Exploiting Similarities among Languages for Machine Translation. arXiv:1309.4168

[50] Tomás Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems (NIPS)*. Curran Associates, Inc., Lake Tahoe, USA, 3111–3119.

[51] Chris Olah, Arvind Satyanarayan, Ian Johnson, Shan Carter, Ludwig Schubert, Katherine Ye, and Alexander Mordvintsev. 2018. The Building Blocks of Interpretability. *Distill* (2018).

[52] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. ACL, Doha, Qatar, 1532–1543.

[53] Nicola Pezzotti, Boudewijn P. F. Lelieveldt, Laurens van der Maaten, Thomas Höllt, Elmar Eisemann, and Anna Vilanova. 2017. Approximated and User Steerable tSNE for Progressive Visual Analytics. *IEEE Transactions on Visualization and Computer Graphics* 23, 7 (2017), 1739–1752.

[54] Paulo E. Rauber, Samuel G. Fadel, Alexandre X. Falcão, and Alexandru C. Telea. 2017. Visualizing the Hidden Activity of Artificial Neural Networks. *IEEE Transactions on Visualization and Computer Graphics* 23, 1 (2017), 101–110.

[55] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, San Francisco, USA, 1135–1144.

[56] Andrew Rouditchenko, Angie Boggust, David Harwath, Brian Chen, Dhiraj Joshi, Samuel Thomas, Kartik Audhkhasi, Hilde Kuehne, Rameswar Panda, Rogerio Feris, Brian Kingsbury, Michael Picheny, Antonio Torralba, and James Glass. 2021. AVLnet: Learning Audio-Visual Language Representations from Instructional Videos. In *Proceedings of the Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA, Brno, Czechia, 1584–1588.

[57] Dominik Sacha, Leishi Zhang, Michael Sedlmair, John Aldo Lee, Jaakko Peltonen, Daniel Weiskopf, Stephen C. North, and Daniel A. Keim. 2017. Visual Interaction with Dimensionality Reduction: A Structured Literature Analysis. *IEEE Transactions on Visualization and Computer Graphics* 23, 1 (2017), 241–250.

[58] Dan Shiebler. 2018. repcomp. https://pypi.org/project/repcomp/

[59] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. 2017. Learning Important Features Through Propagating Activation Differences. In *Proceedings of the International Conference on Machine Learning (ICML)*, Vol. 70. PMLR, Sydney, Australia, 3145–3153.

[60] Daniel Smilkov, Nikhil Thorat, Charles Nicholson, Emily Reif, Fernanda B. Viégas, and Martin Wattenberg. 2016. Embedding Projector: Interactive Visualization and Interpretation of Embeddings. arXiv:1611.05469

[61] Teague Sterling and John J. Irwin. 2015. ZINC 15 – Ligand Discovery for Everyone. *Journal of Chemical Information and Modeling* 55, 11 (2015), 2324–2337.

[62] Hendrik Strobelt, Sebastian Gehrmann, Michael Behrisch, Adam Perer, Hanspeter Pfister, and Alexander M. Rush. 2019. Seq2Seq-Vis: A Visual Debugging Tool for Sequence-to-Sequence Models. *IEEE Transactions on Visualization and Computer Graphics* 25, 1 (2019), 353–363.

[63] Hendrik Strobelt, Sebastian Gehrmann, Hanspeter Pfister, and Alexander M. Rush. 2018. LSTMVis: A Tool for Visual Analysis of Hidden State Dynamics in Recurrent Neural Networks. *IEEE Transactions on Visualization and Computer Graphics* 24, 1 (2018), 667–676.

[64] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic Attribution for Deep Networks. In *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR, Sydney, Australia, 3319–3328.

[65] Luchen Tan, Haotian Zhang, Charles L. A. Clarke, and Mark D. Smucker. 2015. Lexical Comparison Between Wikipedia and Twitter Corpora by Using Word Embeddings. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*. ACL, Beijing, China, 657–661.

[66] Edward R Tufte and Peter R Graves-Morris. 1983. *The Visual Display of Quantitative Information*. Vol. 2. Graphics Press, Cheshire, USA.

[67] Cagatay Turkay, Erdem Kaya, Selim Balcisoy, and Helwig Hauser. 2017. Designing Progressive and Interactive Analytics Processes for High-Dimensional Data Analysis. *IEEE Transactions on Visualization and Computer Graphics* 23, 1 (2017), 131–140.

[68] Peter D. Turney and Patrick Pantel. 2010. From Frequency to Meaning: Vector Space Models of Semantics. *Journal of Artificial Intelligence Research* 37 (2010), 141–188.

[69] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing Data using t-SNE. *Journal of Machine Learning Research* 9, 86 (2008), 2579–2605.

[70] Liwei Wang, Lunjia Hu, Jiayuan Gu, Zhiqiang Hu, Yue Wu, Kun He, and John E. Hopcroft. 2018. Towards Understanding Learning Representations: To What Extent Do Different Neural Networks Learn the Same Representation. In *Advances in Neural Information Processing Systems (NeurIPS)*. Curran Associates, Inc., Montréal, Canada, 9607–9616.

[71] Yanshan Wang, Sijia Liu, Naveed Afzal, Majid Rastegar-Mojarad, Liwei Wang, Feichen Shen, Paul R. Kingsbury, and Hongfang Liu. 2018. A comparison of word embeddings for the biomedical natural language processing. *Journal of Biomedical Informatics* 87 (2018), 12–20.

[72] Martin Wattenberg, Fernanda Viégas, and Ian Johnson. 2016. How to Use t-SNE Effectively. *Distill* (2016).

[73] David Weininger. 1988. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *Journal of Chemical Information and Computer Sciences* 28, 1 (1988), 31–36.

[74] Wesley Willett, Jeffrey Heer, and Maneesh Agrawala. 2007. Scented Widgets: Improving Navigation Cues with Embedded Visualizations. *IEEE Transactions on Visualization and Computer Graphics* 13, 6 (2007), 1129–1136.

[75] Kanit Wongsuphasawat, Dominik Moritz, Anushka Anand, Jock D. Mackinlay, Bill Howe, and Jeffrey Heer. 2015. Voyager: Exploratory Analysis via Faceted Browsing of Visualization Recommendations. *IEEE Transactions on Visualization and Computer Graphics* 22, 1 (2015), 649–658.

[76] Kanit Wongsuphasawat, Zening Qu, Dominik Moritz, Riley Chang, Felix Ouk, Anushka Anand, Jock D. Mackinlay, Bill Howe, and Jeffrey Heer. 2017. Voyager 2: Augmenting Visual Analysis with Partial View Specifications. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI)*. ACM, Denver, USA, 2648–2659.

[77] Jiazhi Xia, Yuchen Zhang, Jie Song, Yang Chen, Yunhai Wang, and Shixia Liu. 2022. Revisiting Dimensionality Reduction Techniques for Visual Cluster Analysis: An Empirical Study. *IEEE Transactions on Visualization and Computer Graphics* 28, 1 (2022), 529–539.

[78] Wei Yu, Kuiyuan Yang, Yalong Bai, Hongxun Yao, and Yong Rui. 2014. Visualizing and Comparing Convolutional Neural Networks. arXiv:1412.6631

A ADDITIONAL CASE STUDIES

A.1 Word Embeddings Pre-trained on Different Corpora

This case study evokes use cases experienced by both model-driven and data-driven users in our formative interviews: choosing between models that appear equally viable for use in a downstream application (e.g., predicting topics based on customer review text). GloVe [52] is a popular model that offers several variants of embeddings trained with different datasets. Here, we demonstrate how the Embedding Comparator can be used to understand the impact of training GloVe with data from either Wikipedia & Newswire or Twitter and replicates known lexical differences between the corpora.

Given the user is tasked with selecting a pretrained model, they begin by analyzing the global shape of each embedding space (Fig. A.1A). While the PCA global projection plots show objects continuously distributed in both models, switching to UMAP projections reveals structure and clusters in global embedding spaces. By hovering over and zooming into the UMAP global projection plots, the user finds clusters of months, names, body parts, and food words within each model. This finding gives the user confidence that both models have learned meaningful representations of natural language and may be suitable for their task.

Since each model captures meaningful natural language structure, the user is now interested in identifying critical differences between the two models that may help them choose the best model for their task. The Embedding Comparator immediately reveals a number of differences between these two pre-trained embedding models that arise due to differences in the underlying training data (Fig. A.1B). Among the words that differ most are shorthand or slang expressions such as *"bc"*, *"bout"*, and *"def"*. Scrolling through the local neighborhood dominoes for these words and comparing their unique neighbors reveals specific ways in which their semantic meanings differ between the two models. For example, in the Wikipedia & Newswire model, *"def"* is used to mean defeat and *"beats"* (as in sporting results) and hence countries (e.g., *"canada"* and *"usa"*), whereas in the Twitter model, *"def"* relates to conversational words such as *"definitely"* and *"probably"*. Another insight revealed by our system is the difference in languages present in the training corpora. Words such as *"era"*, *"dale"*, and *"solo"* take on their English meanings in the Wikipedia & Newswire model, but are related to Spanish words in the Twitter model. This finding suggests that the Twitter model was trained on multi-lingual text, while the Wikipedia & Newswire model may have been trained solely on English text. Finally, the Embedding Comparator reveals how words such as *"swift"* and *"galaxy"* may be used very differently in different media. In Wikipedia & Newswire, *"swift"* refers to the adjective swift (i.e., quick), whereas on Twitter, *"swift"* refers to the musical artist Taylor Swift. Likewise, *"galaxy"* refers either to space or to Samsung Galaxy electronics based on whether embeddings were trained on Wikipedia & Newswire or on Twitter, respectively.

Lexical comparison between Wikipedia and Twitter via embeddings has been explored in previous work [65]. For data-driven users, the Embedding Comparator surfaces characteristic words previously reported to differ most between the two corpora, including *"bc"* and *"ill"* (Fig. A.1B). Moreover, these words are immediately surfaced as a result of our similarity metric without requiring alignment of the two embedding spaces or introducing variance by learning a linear transformation using a stochastic procedure. Using these insights from the Embedding Comparator, a user can make a more informed decision about which set of embeddings may be more appropriate to adopt for their system. For example, if classifying longer or more formal customer reviews, the model trained on Wikipedia & Newswire would likely perform better, but if classifying casually written reviews that contain slang or multilingual text, the Twitter corpus may generalize better to real-world data.

A.2 Chemical Molecules

We applied the Embedding Comparator to compare latent embeddings of chemical molecules [61] learned by two different variational autoencoder (VAE) architectures, Grammar VAE [35] and junction tree VAE [29] (Fig. A.2). This application demonstrates a potential limitation of our system on long object labels. In Fig. A.2A, the molecules are represented as textual SMILES strings [73], which are often used to describe chemical structures. We find that these string representations are long, obfuscating the ability to rapidly scroll through dominoes. Fig. A.2B shows the same models but where a 2-D structure of each molecule is shown instead, rendering dominoes more readable.

B CASE STUDY DETAILS

Here, we detail the datasets and preprocessing steps used in our case studies.

B.1 Transfer Learning for Sentiment Classification

We downloaded pre-trained fastText [48] word embeddings, which are available online at https://fasttext.cc/docs/en/english-vectors.html. We use the 300-dimensional `wiki-news-300d-1M` embeddings consisting of 1 million word vectors trained on Wikipedia 2017, UMBC webbase corpus, and statmt.org news datasets.

We train an LSTM [24] to classify binary sentiment in movie reviews from the Large Movie Review dataset [46] containing 25000 training reviews and 25000 test reviews from the Internet Movie Database (IMDb). We use default tokenization settings for this dataset as provided in Keras [11]. We define our vocabulary as the top 5000 most frequent words in the movie review dataset and truncate reviews to a maximum length of 500 words (with pre-padding). Our recurrent neural network architecture is defined as follows:

1. **Input/Embeddings Layer**: Sequence with 500 words. The word at each timestep is represented by a 300-dimensional embedding.
2. **LSTM**: Recurrent layer with 100-unit LSTM (forward direction only, dropout = 0.2, recurrent dropout = 0.2).
3. **Dense**: 1 neuron (sentiment output), sigmoid activation.

Prior to training, the embeddings are initialized using the pre-trained fastText embeddings. Of our vocabulary of size 5000, 4891

Figure A.1: The Embedding Comparator, applied to case study: *Word Embeddings Pre-trained on Different Corpora*, compares a word embedding model trained on Wikipedia and Newswire text to a model trained on Twitter text. (A) Despite using the same architecture, the similarity distribution and global projection plots show the models represent words differently. (B) Looking at the dominoes suggests the Wikipedia/Newswire model was trained on proper English text, while the Twitter model contains Spanish words (e.g., *"solo"*) and emphasizes popular culture references (e.g., *"swift"*).

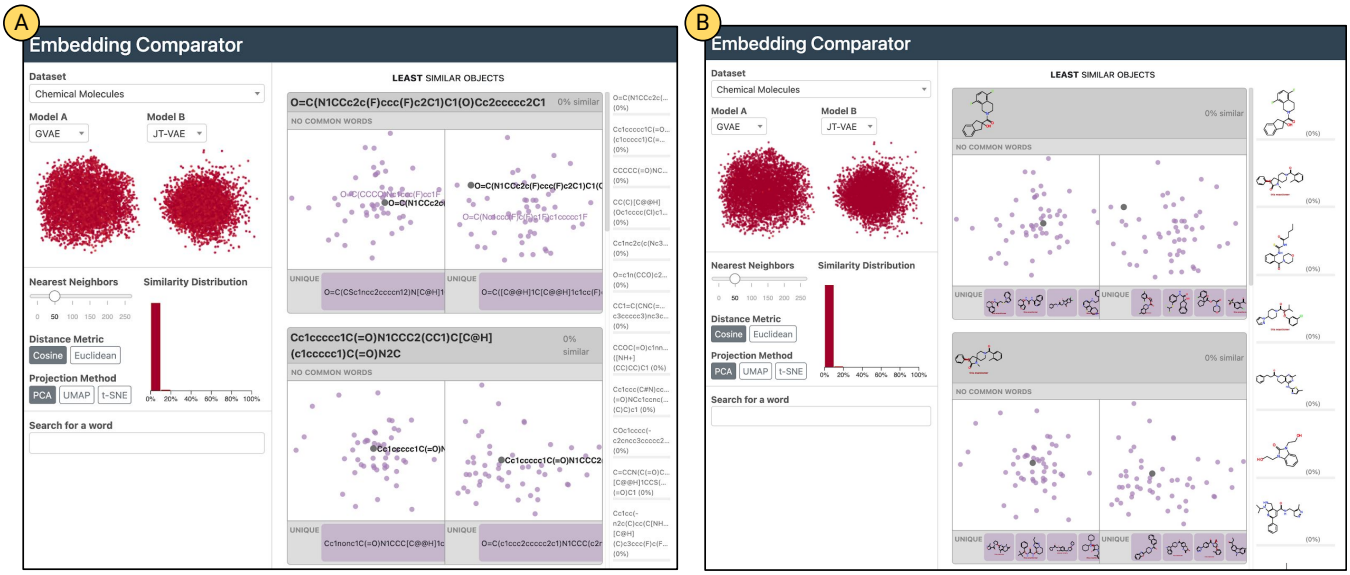

Figure A.2: View of the Embedding Comparator applied to two embeddings of chemical molecules learned by different variational autoencoder architectures. (A) Molecules are represented as SMILES strings. (B) Molecules are represented as 2-D structures.

tokens were present in the fastText embeddings. For tokens not present in fastText, we initialize embeddings as all-zero vectors.

We train our model for 3 epochs (batch size = 64) with the Adam optimizer [32] using default parameters in Keras [11] to minimize binary cross-entropy on the training set. The final model achieves 84.7% test set accuracy (85.4% training set accuracy). We did not further tune the architecture or hyperparameters.

For analysis in the Embedding Comparator, we output the initial fastText embeddings and fine-tuned embeddings for the 4891 words whose embeddings were initialized from fastText.

B.2 Language Evolution via Diachronic Word Embeddings

We use the HistWords dataset [21] in our case study of diachronic word embeddings, which have been shown to exhibit changes in semantic meaning of words over time. We use pre-trained word embeddings from [21], accessed at https://nlp.stanford.edu/projects/histwords/. We use the `All English (1800s-1990s)` set of embeddings, which are 300-dimensional word2vec embeddings [50]. This dataset provides word embeddings trained on English books from each decade from 1800 to 2000. For exploration in the Embedding Comparator, we select embeddings taken from five different decades spanning this time period: 1800-1810, 1850-1860, 1900-1910, 1950-1960, and 1990-2000. We filter each embedding space to the top 10000 most frequent words from its decade and compute the intersection of these sets over the five decades we selected, producing a vocabulary containing 6121 words from each model for comparison in the Embedding Comparator.

B.3 Multimodal Emoji Representations

We use pre-trained 300-dimensional emoji embeddings from emoji2vec [13], accessed from https://github.com/uclnlp/emoji2vec. Our image (pixel) embedding model uses emoji images obtained from https://github.com/iamcal/emoji-data/blob/master/sheet_apple_16.png. The raw $18 \times 18 \times 4$ RGBA images are flattened into a 1296-dimensional vector, which is the pixel model embedding. We use 670 total emojis that are common across these datasets.

B.4 Word Embeddings Pre-trained on Different Corpora

In this case study, we use pre-trained embeddings from GloVe [52], available online at https://nlp.stanford.edu/projects/glove/. The Wikipedia/newswire embeddings were trained on the Wikipedia 2014 and Gigaword 5 (newswire text) datasets containing 6 billion tokens (GloVe 6B), while the Twitter word embeddings were trained on text from Twitter tweets containing 27 billion tokens (GloVe 27B). We use the 100-dimensional embeddings trained on each of these corpora. We filter each of the embedding models to the top 10K most frequent words from its respective corpus and then intersect the resulting vocabularies, giving a shared vocabulary containing 3303 words. We use the Embedding Comparator to compare embeddings from each model for words in this shared vocabulary.

C ADDITIONAL DOMINOES

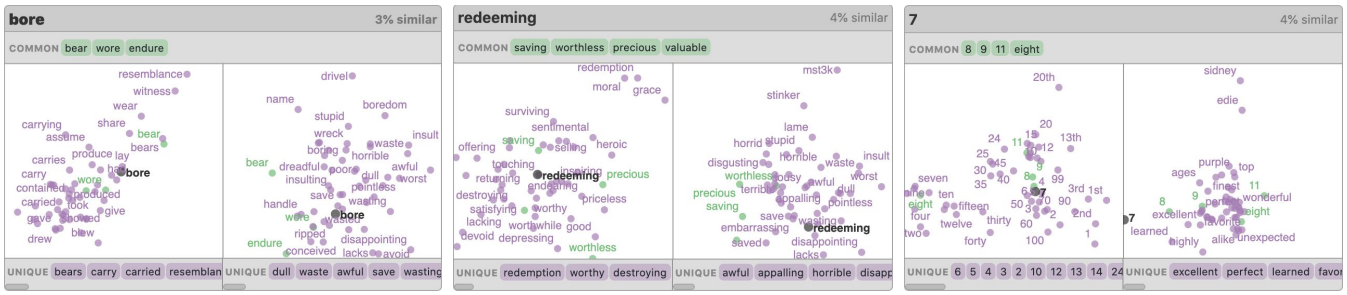

Figure A.3: Additional dominoes from case study: *Transfer Learning for Sentiment Classification*. The word *"bore"* has changed in meaning from a general definition: "carried", to a more sentiment rich definition: "dull". *"redeeming"* has changed from the positive sentiment definition: "compensate for faults" to a negative sentiment definition likely related to the reviewer idiom "no redeeming qualities". The number *"7"* has changed from its definition as a numeric symbol to a number indicative of score (e.g., 7 out of 10).

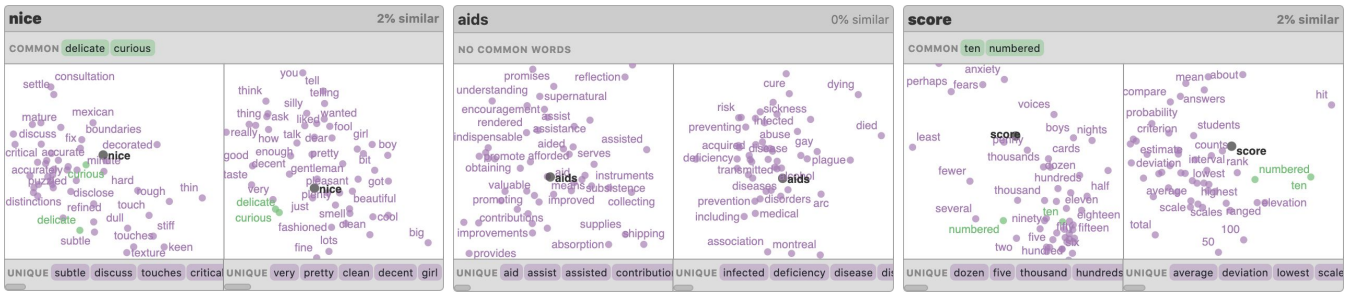

Figure A.4: Additional dominoes from case study: *Language Evolution via Diachronic Word Embeddings*. The domino for *"nice"* shows its definition change from "fine" in 1800-1810 to "pleasant" in 1900–1910. The word *"aids"* was synonymous with "assists" in 1900–1910 but later became associated with HIV/AIDS in 1990–2000. Over the course of the 20th century, *"score"* changed from a measure of time (e.g., four score) to a measure of rank.

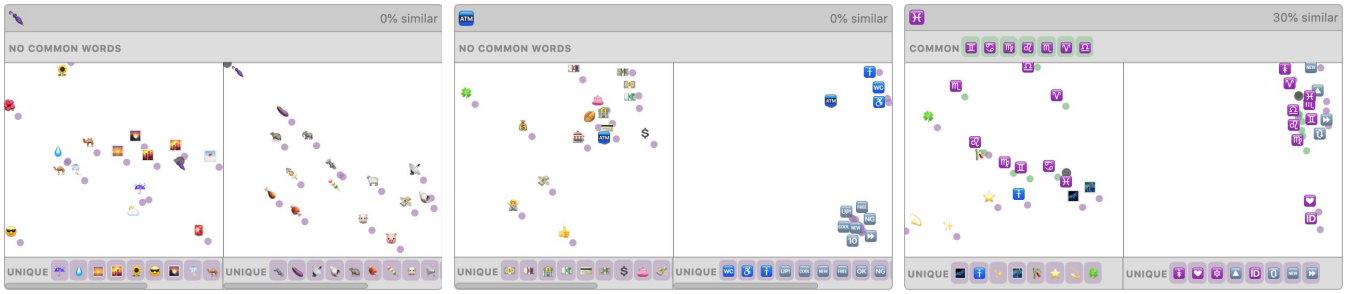

Figure A.5: Additional dominoes from case study: *Multimodal Emoji Representations*. In the language model, the umbrella emoji is related to other weather emojis, whereas in the image model it is related to other slanted emojis. The ATM emoji is related to money emojis in the language model and related to other blue square emojis in the image model. The Pisces emoji is related to other astrology signs in both models because they share astrological meaning and are visually similar.

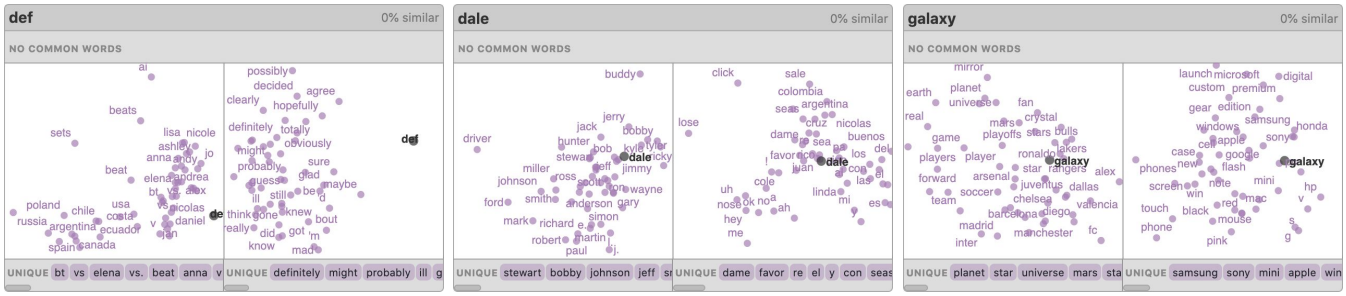

Figure A.6: Additional dominoes from case study: *Word Embeddings Pre-trained on Different Corpora*. Using the model trained on news text *"def"* is short for "defeated", whereas using the model trained on Twitter data *"def"* is slang for "definitely". The word *"dale"* is an English name in the news model, but is a represented by its Spanish meaning in the Twitter model. *"galaxy"* in the news model is related to space, but in the Twitter model is related to the Samsung Galaxy line of phones.

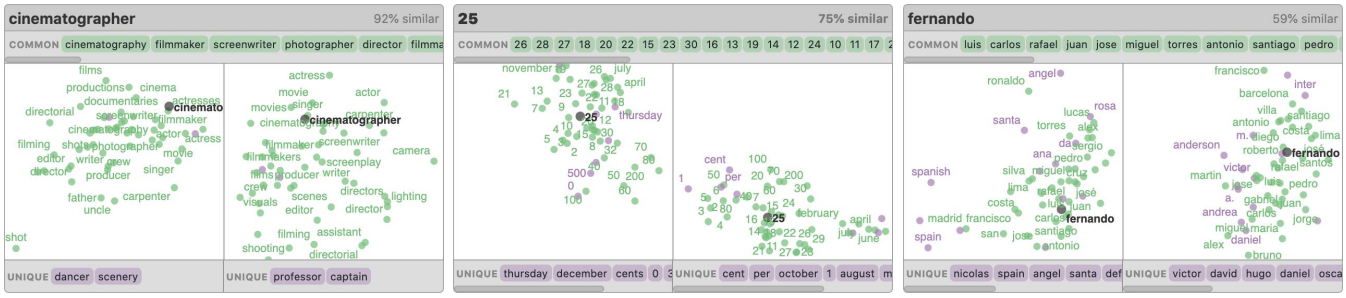

Figure A.7: Additional dominoes where the neighborhood of the word has not changed. From the *Transfer Learning for Sentiment Classification* case study, *"cinematographer"* does not change when fine tuning a general English model on a movie review dataset because *"cinematographer"* already refers to the film industry. From the *Language Evolution via Diachronic Word Embeddings* case study, *"25"* has not changed from 1800 to 2000, indicating that numbers are not susceptible to chronological changes in meaning. *"fernando"* from the *Word Embeddings Pre-trained on Different Corpora* case study does not change in meaning whether the model was trained on news data or Twitter data likely because it is a proper noun with no other meanings.