Eurographics Conference on Visualization (EuroVis) 2019
M. Gleicher, H. Leitte, and I. Viola
(Guest Editors)

Volume 38 (2019), Number 3

Latent Space Cartography: Visual Analysis of Vector Space Embeddings

Yang Liu[1], Eunice Jun[1], Qisheng Li[1] and Jeffrey Heer[1]

[1] Paul G. Allen School of Computer Science & Engineering, University of Washington

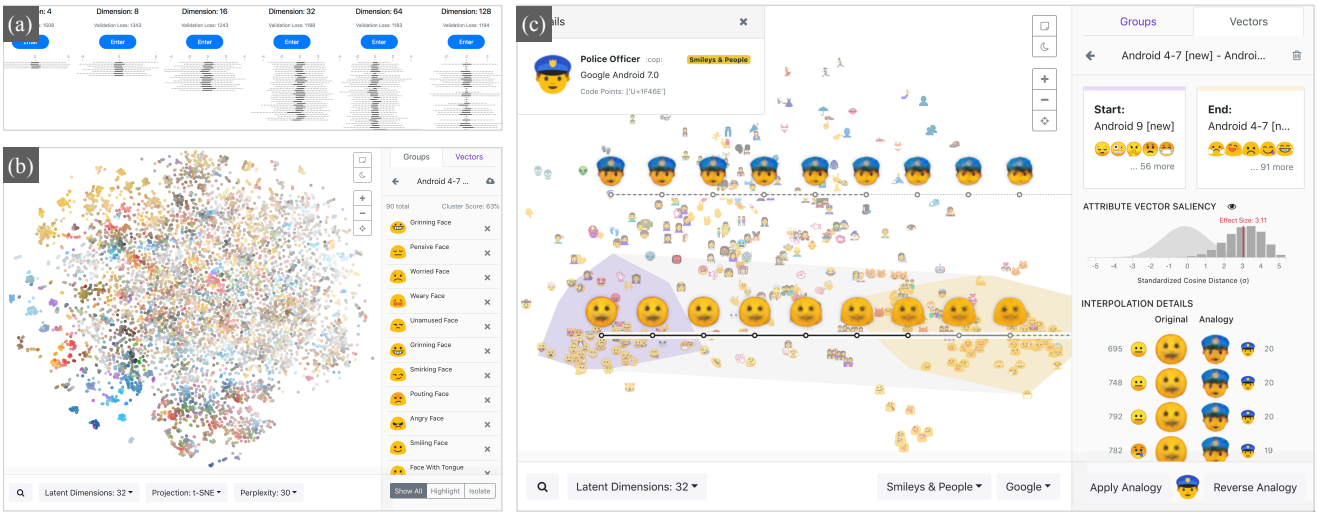


Figure 1: Interpreting latent spaces from variational autoencoders trained on emoji images. (a) The user starts with summary metrics for latent space variants, (b) then drills down to an overview distribution of a chosen latent space. (c) To map out a semantic relationship, the user defines an attribute vector, examines the custom projection to the vector axis, applies analogies and assesses the relationship uncertainty.

Abstract

Latent spaces—reduced-dimensionality vector space embeddings of data, fit via machine learning—have been shown to capture interesting semantic properties and support data analysis and synthesis within a domain. Interpretation of latent spaces is challenging because prior knowledge, sometimes subtle and implicit, is essential to the process. We contribute methods for "latent space cartography", the process of mapping and comparing meaningful semantic dimensions within latent spaces. We first perform a literature survey of relevant machine learning, natural language processing, and scientific research to distill common tasks and propose a workflow process. Next, we present an integrated visual analysis system for supporting this workflow, enabling users to discover, define, and verify meaningful relationships among data points, encoded within latent space dimensions. Three case studies demonstrate how users of our system can compare latent space variants in image generation, challenge existing findings on cancer transcriptomes, and assess a word embedding benchmark.

CCS Concepts

• ***Human-centered computing*** → *Visualization; Visual analytics;*

1. Introduction

A central goal of unsupervised machine learning is to distill unlabeled data into more useful and ideally meaningful *data representations*. In Natural Language Processing (NLP), state-of-the-art applications often depend on *word embeddings*: vector representations of words learned from unlabeled text corpora. In generative modeling, methods like variational autoencoders (VAEs) [KW13] and generative adversarial networks (GANs) [GPAM*14] create latent spaces for modeling and synthesis of data [CN17].

Despite arising from different algorithms serving distinct purposes, these representations are all vector spaces of reduced dimensionality (relative to the input), intended to produce more general features that helpfully characterize the input. These representations are often referred to as *latent spaces*. Latent spaces are important mainly in two aspects. First, they can provide insights into the data, sometimes revealing important relationships we are unaware of. For instance, linguists analyze word embeddings to understand how words change meaning over time [HLJ16], and scientists use latent spaces to assist discovery of new biological pathways in ovarian

cancer [WG18]. Second, latent spaces can serve as feature spaces for downstream machine learning applications, including classifiers and other supervised predictors. In both cases, interpretation of latent spaces can assist the evaluation of models, help explain model performance, and more generally aid understanding of what, exactly, a model has "learned". Interpretation is especially critical when evaluating models for which we lack ground truth data.

Interpretation of latent spaces often requires subtle and implicit domain knowledge, for which human judgment is essential. This human-centric nature makes visual analysis methods attractive. Currently, most examples in the literature rely on static, ad-hoc visualizations. Several interactive tools exist [LBT*18, HG18, STN*16, LNH*18], but they either focus on a specific domain and a narrow set of tasks, or tackle each task separately without combining them into an integrated tool. We seek to guide users through a comprehensive workflow that supports tasks common to latent spaces across various input data types and learning algorithms.

In this paper, we contribute methods for *latent space cartography (LSC)*, the process of mapping and comparing meaningful semantic dimensions within the latent space. We first survey the literature across a range of research communities and bring together common interpretation tasks. We then integrate these tasks into a visual analysis system that enables users to quickly discover, define, and validate semantic relationships (or *attribute vectors*) within the latent space. We augment the system with additional novel methods to provide context, assess relationship uncertainty, and compare relationships. We explore use case scenarios for multiple data types, including image generation of emojis, scientific feature learning for cancer genomics, and word embeddings. Using LSC, we find a better justified interpretation of a scientific dataset that leads to potential new discoveries, and shed light on nuances overlooked by a state-of-the-art NLP test benchmark.

In summary, we contribute (1) a review of domain literature and a characterization of common latent-space interpretation goals and tasks; (2) a visual analysis system that supports these tasks, including novel projection strategies, visual and statistical methods to assess attribute vector uncertainty, and global attribute vector comparison methods; and (3) three case studies demonstrating new insights into diverse data types across multiple domains.

2. Background

We introduce the notion of latent spaces using two prominent examples: word embeddings and generative network models.

Word Embeddings. Word embeddings represent words as real-valued vectors, typically with 50 to 300 dimensions. A training algorithm, such as word2vec [MSC*13] or GloVe [PSM14], takes a large text corpus as input. After preprocessing, the algorithm constructs a co-occurrence matrix which encodes the probability of two words appearing in the same context. It then employs various strategies (*e.g.,* matrix factorization) to produce an embedding that preserves co-occurence information. The embeddings are used in various NLP applications, including parsing [SBMN13], named entity recognition [TRB10], and sentiment analysis [MDP*11].

As word embeddings are optimized for co-occurence information, words that frequently appear in similar contexts usually have similar vectors. Vector space coordinates may encode syntactic similarity (*run*, *running*), semantic similarity (*large*, *big*) or relatedness (*coffee*, *cup*) [HRK15, FGM*01, BTB14]. Additionally, word embeddings display intriguing linear regularities commonly referred to as *analogies* [MYZ13]. A canonical example shows that the vector(*king*) - vector(*man*) + vector(*woman*) produces a vector very close to vector(*queen*). This simple algebraic operation illustrates that a particular relationship might have a constant vector offset in the space. This vector offset is also called an *attribute vector* and the algebraic operation is called *attribute vector arithmetic*.

Latent Spaces in Generative Modeling. Generative models are able to synthesize realistic outputs resembling observed data after being trained on real examples. The variational autoencoder (VAE) [KW13] is a prominent generative modeling method. VAEs consist of two networks: an *encoder* to map an input datum to a latent vector, and a *decoder* to map a latent vector back to the input space. The training objective seeks to minimize the reconstruction error (how accurately a decoded example matches the original input) and the Kullback-Leibler divergence (how closely the latent variables follow a probability distribution) simultaneously.

Though the explicit objective of VAE is to generate realistic examples, it produces latent representations that can capture salient information about the data. Akin to word embeddings, these latent spaces are continuous multi-dimensional vector spaces. Researchers have found that similar analogical relationships exist in these latent spaces [RMC15]. That said, these generative models differ from word embeddings in an important way: while there only exists a one-way mapping from words to latent vectors in word embeddings, a VAE decoder can convert any latent vector to a reconstructed example, enabling synthesis of new instances.

3. Related Work

We draw on multi-dimensional data visualization techniques and explore the design space for visualizing latent spaces.

3.1. Visualizing Multi-dimensional Data

Dimensionality reduction techniques are common for visualizing multi-dimensional latent spaces. t-Distributed Stochastic Neighbor Embedding (t-SNE) [VdMH08] and Principal Component Analysis (PCA) [Jol11] are among the most popular methods, while Uniform Manifold Approximation and Projection (UMAP) [MH18] is a more recent alternative. t-SNE is a non-linear technique that aims to match neighbors in the original space to those in the lower dimensional embedding. Recent work has extended t-SNE to accelerate computation [VDM14, PMH*18], visualize non-metric similarities [VdMH12], and trade off accuracy for interactivity [PHL*16, PLVDM*17]. UMAP is another non-linear technique that better preserves inter-cluster relationships. These non-linear algorithms highlight cluster structures, but can obscure linear relationships among points. PCA is a linear transformation and so preserves linear relationships. Other popular multi-dimensional data visualization techniques, for instance scatter plot matrices [CLNL87] and parallel coordinates [ID90], are less relevant because neither the axes of a latent space nor the values along a latent space axis have intrinsic meaning. We use t-SNE, UMAP, and PCA in this paper.

Previous work has also proposed custom projection methods. Interaxis [KCPE16] and Explainers [Gle13] allow users to define semantic axes and subsequently project data points onto the axes. Applying this idea to word embeddings, related work [HG18, BCZ*16a, BCZ*16b] has mapped a set of words to two user-defined concept axes. We similarly enable users to construct a semantic axis from two opposing concepts, but unlike methods that simply layout two user-defined axes, we provide additional projection strategies to ensure orthogonality and highlight variations. Liu *et al.* [LBT*18] propose two projection schemes combining Support Vector Machines (SVM) and regularization, or SVM and PCA, to better visualize analogy pairs in word embeddings.

3.2. Visualizing Latent Spaces

The research literature involving latent spaces often employs visualizations for qualitative evaluation. Most of these visualizations are 2D scatter plots, using two intrinsic latent dimensions directly [Wet17, MNG17, YHSBK17], axes from dimensionality reduction techniques such as t-SNE (*e.g.,* [TWR*17, JYY*16, HSSQ17, MKS*15, YSD*18]) or PCA (*e.g.,* [SRM*16, MSSW16, UFDR16, FSBL17, GBWD*18]), or axes of custom projections [BCZ*16a, BCZ*16b]. Alternatively, some papers show a 2D grid of reconstructed examples [DTD*18, JBJ18, ZSE17, KPHL17]. These visualizations are typically ad-hoc and static.

Researchers have also developed interactive visual analysis tools for latent spaces [STN*16, JSL*17, HG18, LBT*18, LNH*18]. Some tools focus on a subset of tasks [STN*16, LBT*18] in word embeddings, which we extend and bring to a broader range of latent spaces. Heimerl and Gleicher [HG18] catalog domain-specific tasks for word embeddings and propose separate designs for each task. We similarly perform a task analysis to include more diverse venues, and instead of tackling each task separately, we build our system to support an integrated workflow. Finally, our work is directly inspired by the article by Carter and Nielsen [CN17] that discusses semantic dimensions and applications of latent spaces.

More broadly, our work relates to the literature on visual analysis of neural networks (see [HKPC18] for a survey). Previous work has contributed techniques and systems to visualize hidden layers [ZF14, LSL*17], training process [LSC*18, PHVG*18], model architecture [WSW*18] and supervised learning results [RAL*17]. Analyzing these aspects of the neural network are complementary to our focus on understanding latent spaces.

4. Goals, Tasks, and Workflow

Our design goal is to support model users in interpreting latent spaces. Our primary target audience consists of end users of machine learning (ML) tools, capable of data science work but not necessarily developers of ML tools. We desire a general system that supports latent spaces from generative models and word embeddings, across various domains, so long as the goal is to map semantically meaningful relationships within a vector space representation of data. To provide guidance to model users and bring together techniques across multiple fields, we perform a structured literature review to understand common practices adopted by domain experts in understanding and evaluating latent spaces.

We started the literature search on arXiv [ArX], an interdisciplinary archive for diverse research communities. We searched using algorithm keywords, excluded irrelevant articles, and sampled a subset due to the vast quantity of papers. Following this procedure, we arrived at 78 papers from a wide range of publication venues, including 44 papers from ML, artificial intelligence, and computer vision related conferences and journals, 27 from NLP, and 7 from scientific fields such as Physics, Astronomy, and Biology (the detailed protocol and all surveyed articles are in the supplemental material). We analyzed these articles using an iterative coding method to catalog the goals and tasks in interpreting latent spaces.

4.1. Uses and Interpretation Goals

The domain literature uses latent spaces for various purposes:

- *Improve downstream tasks.* Some articles employ latent spaces as an intermediate step to improve downstream task performance. For instance, latent spaces can provide robust features to aid classification in semi-supervised learning, where only a small subset of observations have labels (*e.g.,* [KMRW14]).
- *Enable synthesis.* Articles on generative applications utilize latent spaces to synthesize specific types of outputs. Examples include synthesizing images [YYSL16], sentences [JZS17], music [HNP17], and molecules [JBJ18].
- *Understand data.* Researchers leverage latent spaces to identify valuable relationships, enhancing their understanding of the data and potentially leading to new discoveries. Examples include identifying cancer-associated genes [WG18] and quantifying shifts in word meaning [HLJ16].
- *Unspecified.* The remaining articles, typically proposing new unsupervised algorithms, make no assumption on how the learned latent spaces will be used.

In addition to simply involving latent spaces as an algorithmic component, the articles we reviewed all seek to further understand and compare latent spaces. We identify three high-level goals guiding the *interpretation* process:

- *Evaluate model.* Researchers use evidence from latent spaces to compare and evaluate proposed algorithms. Most notably, unsupervised algorithms often lack an ultimate task to evaluate against. Therefore, researchers routinely rely on *intrinsic evaluation*—methods that gauge a latent space using information within the space itself—to evaluate unsupervised models.
- *Explain model.* Researchers examine the internal working of latent spaces to explain (or, at least, gain intuitions of) the model. For instance, some papers use discriminative clustering in latent spaces—whether different classes form distinct clusters—to explain the overall classification performance (*e.g.,* [ADvdH17]).
- *Understand data.* Researchers explore latent spaces to understand the underlying structures in the data. They may find that clusters in the latent space reflect galaxy population bimodality [FSBL17], or the cosine similarity between the same words from different embeddings trained on different corpora signifies a change in word meaning [HLJ16].

During interpretation, researchers are interested in various aspects of the latent spaces. A recurring concern is that latent spaces organize information in a meaningful rather than random manner.

For instance, researchers might show that interpolation produces semantically meaningful sequences. They might also show that similar items appear in closer proximity in the embedding and that intriguing regularities such as analogical relationships exist in the space. Besides interpretability, researchers also verify other mathematical properties. They may ask if a continuous latent space is smooth, as abrupt transitions are considered a sign of memorization [RMC15]. Some projects verify specific properties or diagnostics, for instance the absence of a "posterior-collapse" [MSSW16] in which some latent dimensions have zero weights.

4.2. Tasks and Improvements

We describe six interpretation sub-tasks appearing most frequently in the domain literature. As the types of latent spaces affect tasks, we report frequency counts separately for word embeddings (24 in total) and latent spaces in generative models (54 in total).

T1: **View Reconstruction Examples.** A majority of articles on generative models (31/54) present a list of example generation results. These examples serve as qualitative evidence that the models produce compelling, plausible, yet novel outputs unseen in the training data. They also show reconstruction fidelity versus generation diversity, as there is usually a trade-off between the two.

T2: **View Interpolation Results.** Linear interpolation in a continuous latent space is done by following a path between two points and displaying outputs at sampled locations on the path (usually in steps of equal distance). A number of articles (18/54) adopt this approach. Interpolation sequences naturally show the smoothness of the latent space, and they are important indicators of interpretability. For instance, authors may point out subtle changes in transitions that reflect deeper domain-specific principles [CN17].

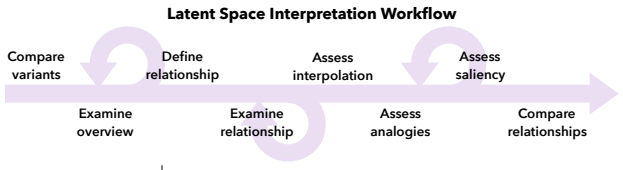
T3: **Examine Nearest Neighbors.** This task is employed by articles on both word embeddings (10/24) and generative models (6/54), but the emphasis is different. Word embedding articles use anecdotal nearest neighbors to qualitatively show what the algorithms manage to achieve. Articles on generative models use nearest neighbors from the training set as a comparison, demonstrating that the algorithms generate novel results indistinguishable from, or better than, the input.

T4: **Perform Attribute Vector Arithmetic.** An example of this task for word embeddings shows that the vector(*king*) - vector(*man*) + vector(*woman*) produces a vector very close to vector(*queen*). This simple algebraic operation demonstrates that pairs of data points sharing a particular relationship have approximately constant vector offsets. We observe this task in both generative (3/54) and word embedding (10/24) models.

T5: **Compare Similarities.** This task is popular for benchmarks of word embeddings (13/24), testing how well the cosine similarity between word vectors matches ratings from human annotators.

T6: **Visualize Distribution.** Researchers visualize latent spaces using various means described in §3. This task is common to both generative (20/54) and word embedding (7/24) settings.

We augment the interpretation sub-tasks above by adding three improvements to cover potential blind spots in the literature and address deeper questions about semantic relationships:



Activity	Description
Compare variants	Compare latent space variants using quantitative metrics. (T4, T5)
Examine overview	Select a latent space and view the projected data distribution. (T6)
Define relationship	Define an attribute vector as an externalized relationship.
Examine relationship	View data distribution relative to the attribute vector. (T6)
Assess interpolation	View interpolation along the attribute vector. (T1, T2, T3, S1)
Assess analogies	Perform attribute vector arithmetic. (T3, T4, S1)
Assess saliency	Assess if the attribute vector represents a salient relationship. (S2)
Compare relationships	Examine how multiple attribute vectors relate. (S3)

Figure 2: A workflow for latent space interpretation. Background loops signify that all activities may be iteratively performed.

- S1: *Contextualize interpretation sub-tasks*. While visualizing data distribution (T6) provides useful context for other tasks (for instance showing where an interpolation (T2) sequence occurs or whether attribute vector arithmetic (T4) only works in certain subspaces), such aspects are rarely combined in prior work. We aim to contextualize interpretation tasks whenever possible.
- S2: *Assess relationship strength*. While domain literature routinely assigns linear vectors to semantic relationships (*e.g.,* T4), the "strength", or saliency, of such mapping is often overlooked. We hope to assess whether a linear vector captures a salient human-interpretable relationship.
- S3: *Compare multiple relationships*. Interpreting latent spaces can be viewed as mapping the machine-learned vector space to a human-friendly vector space where the axes correspond to meaningful relationships. To this end, we would like to enable users to judge how multiple semantic relationships relate.

4.3. Workflow

To integrate tasks T1-T6 and improvements S1-S3, we propose an analysis workflow to guide users through latent space interpretation (Figure 2). The workflow starts with summary metrics for one or more fitted latent space variants, then drills down to in-depth exploration of a selected latent space. As interpretation may be an iterative process – and not all tasks may apply in all cases – users should be free to skip, repeat, or return to previous activities. Our LSC system design is intended to support this workflow.

To demonstrate that LSC is generic to both generative models and word embeddings across multiple domains, we include built-in support for three data types: generative models of images, generative models of arbitrary numerical vectors, and word embeddings for text. That said, LSC is extensible to support other types of input data by implementing an appropriate rendering module. For instance, we could extend the tool to support audio data by implementing a waveform visualization with audio playback support.

5. System Walkthrough

We present our LSC workflow, features, and design choices in a usage scenario where Jimmy, a non-expert user of ML tools, uses our system to explore latent spaces of VAEs trained on emoji images. During the walkthrough, we note important distinctions to

other data types (*e.g.,* word embeddings), and elaborate on these details in §6. Other details on visual encodings, interactions, and mathematical definitions are available as supplemental material.

Jimmy is enthusiastic about design and deep learning. Upon reading an article [CN17] introducing how VAEs learn meaningful relationships, he is excited that a VAE might help him explore discrepancies in emoji styles across platforms and offer design inspiration for creating new emojis. To start, he crawls 24,000 emoji images from Emojipedia [Emo], which includes emojis from various platforms (*e.g.,* Apple, Google) and versions. After preprocessing, he trains multiple convolutional variational autoencoders using the same architecture, but he varies the number of latent dimensions as a hyper-parameter. From this process, Jimmy ends up with several latent spaces with differing dimensions. He now wants to discover and validate relationships in these vector spaces. Does the model learn meaningful semantics? Might it discover underlying design principles? How do results vary based on the dimensionality of the space? To answer these questions, Jimmy starts an exploratory analysis in LSC.

5.1. Initial Comparison

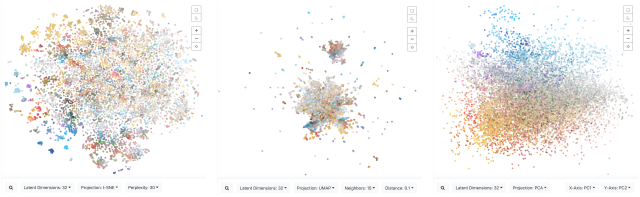
Jimmy wants to choose the best candidate latent space to initially analyze. Upon loading data, he lands on a summary page showing a few preliminary metrics. Seeing that the validation loss starts to plateau at dimension 32, he decides to focus on this dimension.

At the beginning of the interpretation workflow, LSC lists several quantitative metrics to facilitate initial comparison. For generative models, we show validation loss and the distribution of data on initial latent axes (Figure 1a), which helps to detect potential "posterior collapse" problems [MSSW16]. For word embeddings, we leverage existing similarity (T4) and analogy (T5) benchmarks in the NLP community, namely WordSim-353 [FGM*01], SimLex-999 [HRK15], MEN [BTB14], and Google's analogy dataset [MYZ13]. These summary results give users an initial quality judgment, helping them choose a reasonable latent space for an in-depth exploration.

5.2. Vector Space Overview

After selecting the 32-dimensional latent space, Jimmy enters an overview page showing how emojis distribute within the space. Seeing the apparent clustering structures, he brushes and hovers to inspect what emojis lie inside a cluster.

Once users decide to focus on a latent space, LSC visualizes an overview (T6) to orient them for further exploration. LSC shows a 2D t-SNE scatter plot by default (Figure 1b, Figure 3a). We choose t-SNE because it prioritizes cluster structures within the data, and these clusters reveal how the latent space learns and organizes information. For example, one might use prior knowledge and metadata labels to judge if a cluster is meaningful, answering questions such as "does the latent space group emojis based on platforms (*e.g.,* Google versus Apple), content (*e.g.,* faces versus flags), color, or shape?" Similarly, UMAP projection (Figure 3b) is available as an alternative. Besides the non-linear projections, LSC also supports PCA. PCA enables users to examine salient linear dimensions



(a) (b) (c)

Figure 3: Vector space overview. We apply (a) t-SNE, (b) UMAP, and (c) PCA to visualize sample distributions. Here each dot represents an emoji image, colored by its average pixel color.

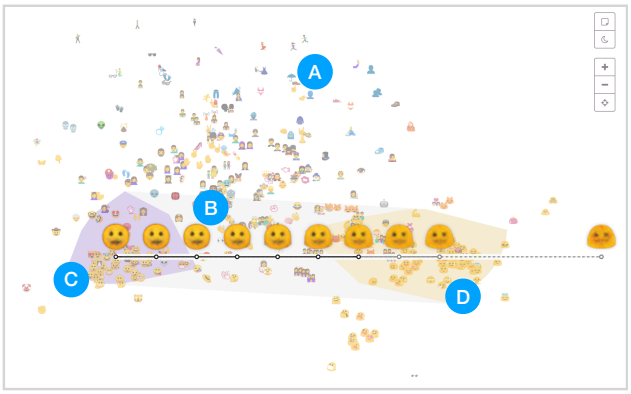


Figure 4: Attribute vector view. Samples (a) are projected onto the attribute vector dimension. By default, the x-axis is the attribute vector direction and the y-axis is the largest principal component of the remaining dimensions. The attribute vector (b) is plotted along with the convex hulls of the starting (c) and ending (d) concepts. We show interpolation results where applicable.

in the data free of the distortions of t-SNE and UMAP. For example, the first two principal components in Figure 3c show that silhouette and color dimensions contribute to the largest variations in emojis.

5.3. Attribute Vector Mapping

As he gains initial familiarity with the vector space, Jimmy begins to formulate more specific questions. He notes that Android version 9 adopts a distinct style for face emojis 😀 compared to its earlier versions 😁. He wants to understand if the latent space is sensitive to this trend, so he proceeds to define an attribute vector.

As users explore and discover meaningful relationships, LSC enables users to externalize such relationships as attribute vectors. To define an attribute vector, users first collect examples from two opposing concepts that constitute the relationship. They might rely on existing metadata labels (*e.g.,* all Apple emojis containing "woman" in their names), or interactively group samples when exploring the vector space overview (*e.g.,* all emojis within a cluster in t-SNE). The attribute vector is then defined according to the endpoints of the centroids of the two opposing concepts.

Jimmy creates an attribute vector that transitions from examples like 😀 😗 😛 to examples like 😁 🤭 😊. He selects the attribute vector to enter a view where all emojis are projected, using the attribute vector as a primary axis. He explores the space to understand how emojis distribute along the spectrum of the two styles.

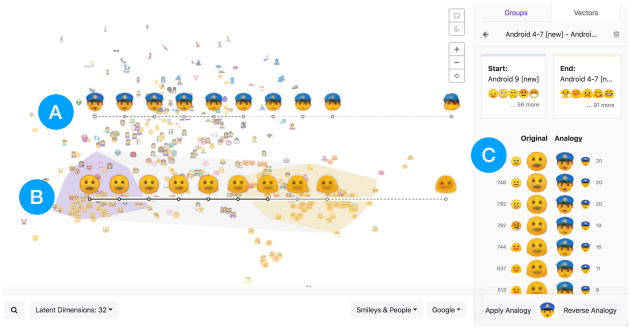


Figure 5: Analogy. An analogy vector (a) is obtained by adding the attribute vector (b) to an emoji example. In (c), the nearest neighbor in the training data is shown beside each reconstructed result.

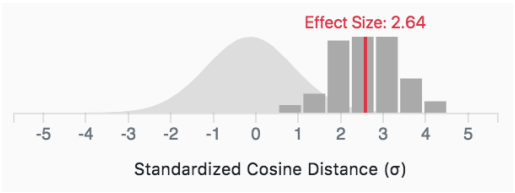


Figure 6: Pair alignment to assess relationship saliency. We compare a histogram of pairwise cosine distances between attribute pairs to a background distribution of random pairs. We standardize the unit using the pooled standard deviation of the two distributions.

Once a user creates an attribute vector, we offer a new perspective to explore the latent space, treating the attribute vector as a dimension. The user might assess how other samples distribute in the spectrum: what examples are considered more similar to one concept than the other in the latent space? Essentially, we assume the relationship manifests as a linear structure within the latent space.

To this end, LSC performs a custom projection (T6) onto the attribute vector (Figure 4a). By default, the x-axis is the direction of the attribute vector and the y-axis is the first principal component (largest eigenvector) of the remaining dimensions. Unlike previous methods that project onto two attribute vectors, we ensure that the two axes are orthogonal in the original space, thus avoiding angle distortions (supporting math is supplemental material). The default y-axis highlights variations in data, but users are free to map the y-axis to other metadata fields, for instance grouping samples by category to inspect the distribution within each category.

"How about generation quality? Might the model create a series of emojis that transition from one style to another?" With these thoughts, Jimmy inspects the interpolation sequence.

We interpolate (T2) along the attribute vector where applicable (Figure 4b), showing reconstructed examples (T1) at constant steps along the line. We also show the nearest neighbor (T3) in the training set beside each generated image as a comparison (Figure 5c). Together, these operations allow users to gauge reconstruction quality and verify geometric (*e.g.,* abrupt transitions) and semantic (*e.g.,* meaningful transitions) properties. The interpolation sequence is plotted in the custom projection (S1), making it possible to take into account the surrounding data distribution when reasoning about the results.

While the style changes as expected for the averaged examples, Jimmy wonders how the model applies the style transformation to other emojis. He picks 😇 and applies the attribute vector.

Using LSC, a user can verify how well an attribute vector applies to an arbitrary sample (Figure 5). One can pick an example in the projected view and ask LSC to perform attribute vector arithmetic (T4): add or subtract the attribute vector from the sample. They can then observe if the result matches their expectations and thereby better understand how the latent space encodes information. LSC draws the resulting vector in the projected data distribution (S1), making it easy to answer questions such as "does the analogy work only for points near the starting convex hull?".

We display different results depending on the types of latent spaces. For generative models of image data, we show interpolated examples (T2). As word embeddings lack the backward mapping from an arbitrary latent vector to a word, we list the top-k nearest neighbors (T3) of the start and end locations (Figure 14).

Now Jimmy would like to know if this attribute vector reliably represents a salient relationship. He examines the convex hulls and reasons about the pairwise cosine similarity plot.

Next, we enable users to assess the consistency and saliency of the linear relationship captured by the attribute vector (S2). First, LSC visualizes the projected convex hulls alongside the attribute vector (Figure 4). It plots convex hulls enclosing examples in two concepts (purple and yellow) and all possible pairs between individual examples (gray). While the attribute vector is essentially an *average* of the relationship, the convex hulls serve as an *uncertainty* measure, indicating the "spread" of the individual pairs.

Though the convex hulls intuitively illustrate how well-aligned the individual pairs are, they are still dependent on the projection method. To complement this, LSC visualizes pair alignment in the original space. We adopt a *relative* formulation, comparing alignment of pairs within the attribute vector to random pairs. As shown in Figure 6, we first visualize the distribution of pairwise cosine similarities between pairs defining the attribute vector. We then collect a large number of pairs between random examples and plot their pairwise cosine similarity. Well-separated distributions imply that the attribute vector captures a salient relationship difficult to observe by random chance. To enable comparison of a single attribute vector across latent space variants as well as the relative "strength" of attribute vectors within the same latent space, we further standardize the unit as Cohen's d, using pooled standard deviation. This choice ensures robust comparisons across latent spaces, as average cosine similarities decrease with higher dimensionality.

5.4. Comparing Attribute Vectors

Jimmy goes on to investigate more attribute vectors. As the vectors accumulate, he wonders how they relate. He views the vectors in several global projections and checks their orthogonality.

Given a set of attribute vectors, we provide multiple means for users to evaluate how they relate (S3). First, we support visualizing the attribute vectors globally in any available projection (Figure 7). Projecting attribute vectors to a linear projection such as PCA or

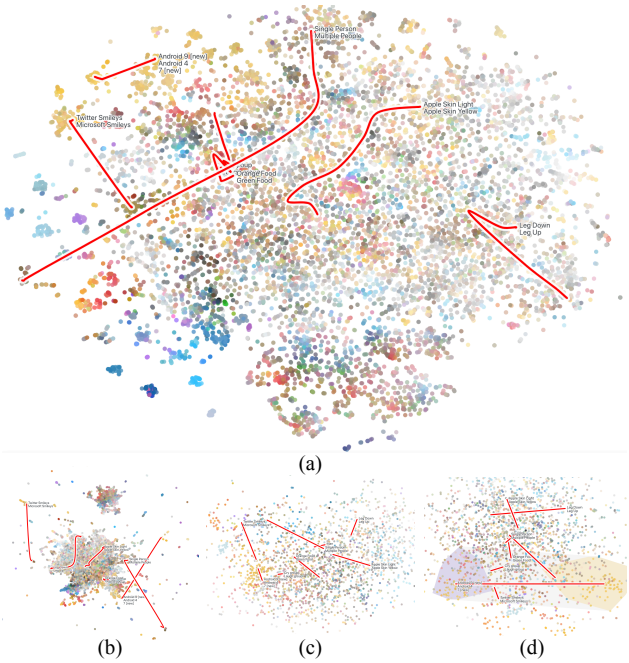


Figure 7: Visualizing attribute vectors in a global view, including (a) t-SNE, (b) UMAP, (c) PCA and (d) attribute vector projection.

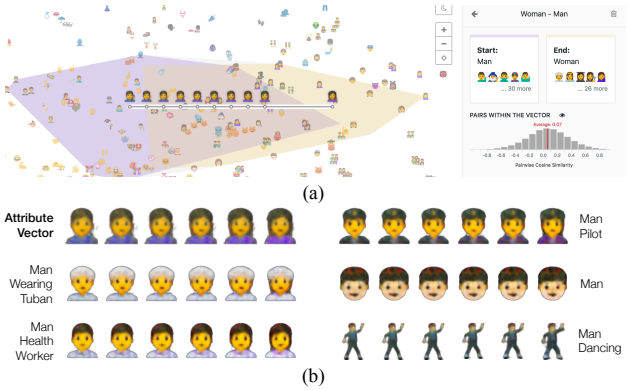


Figure 8: (a) An attribute vector between man and woman faces. (b) Applying the attribute vector to man emojis often "grows" the hair, but fails for faces without shoulders or full-body views.

our custom projection is straightforward, as we simply apply the projection transformation to the vector. For non-linear projections, we approximate the path of an attribute vector. We (1) sample along the attribute vector in the original high-dimensional space to obtain control points, (2) project the control points into the 2D embedding, and (3) render a Catmull–Rom spline [CR74] along the projected control points. UMAP supports step (2) as it can map new points to an existing embedding, but t-SNE lacks such support. We approximate projected control point positions in t-SNE embeddings using a weighted nearest neighbor approach. Specifically, we first obtain k nearest neighbors of a control point in the original space. These nearest neighbors are existing data points, so we can obtain their coordinates in the 2D embedding. We then compute a weighted average, where the weight is the inverse of the distance between a neighbor point and the control point. We use this weighted average as the coordinate of the projected control point in 2D. More mathematical details are provided in §2.2 of the supplemental material.

Next, we assist users in evaluating the relative similarity and orthogonality of the attribute vectors, by showing the cosine similarity between attribute vectors. Since attribute vectors represent linear relationships, orthogonal vectors represent independent dimensions that do not vary together. These semantic dimensions might ultimately become axes to re-orient the latent space.

5.5. Cross-Model Assessment

Now Jimmy feels that he has explored enough of the 32 dimensional space. He returns to the summary page to see how the attribute vectors hold up across models.

We treat attribute vector sets as a user-defined evaluation metric. In the summary page, in addition to initial automated assessments, we supply quality measures of all the attribute vectors in each latent space. For these quantitative summary scores of attribute vectors, we use Cohen's d values indicating pair alignment within an attribute vector as described before. Users might use the summary to compare latent spaces, identify a potentially different space variant, or start a new iteration of exploration. We also allow users to export attribute vectors for use in external analysis tools.

6. Case Studies

We demonstrate LSC in case studies for diverse domains, providing insights for one novel application (emojis) and two drawn from the literature (cancer transcriptomes and word embeddings).

6.1. Case Study: Emojis (Generative Image Analysis)

We present insights discovered by external users—students from a visualization class—during their exploration of the emoji dataset described in §5. These examples demonstrate how LSC supports interpretation goals toward explaining (what does the model really learn?) and evaluating (which model is best for synthesis?) models.

What does the latent space learn about gender?

Julia creates an attribute vector that transitions emojis depicting a male face to those depicting a female (Figure 8a). Figure 8b shows interpolation sequences (T2) that she explores. Observing how the attribute vector turns an averaged male face into female, she hypothesizes that the latent space encodes gender differences as the length of the hair. Applying the attribute vector (T4) to other emojis of a man's face generally work as expected. For example, interpolating from *Man Health Worker* correctly grows the hair. The analogical relationship also holds for slightly more complex emojis such as *Man Wearing Turban* and *Man Pilot*. Note that the transformation does not touch irrelevant properties (*e.g.,* the turban).

Julia also finds surprising cases where the analogy breaks. While she expects that it is easy to turn the simple emoji *Man* into its female counterpart, the emoji remains practically unchanged by the attribute vector. The attribute vector also fails to transform a full-body view like *Man Dancing*. She generalizes from these failure cases that the latent space does not represent gender stereotypes or hair length in any *semantic* sense, but instead simply adds dark pixels to specific locations of the image.

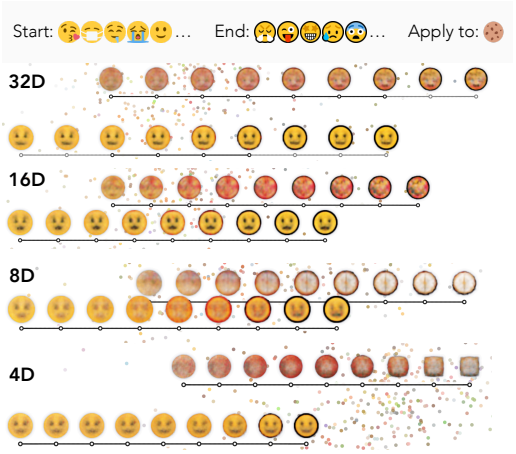


Figure 9: Comparing an attribute vector across latent spaces. The attribute vector adds an outline stroke to smileys. The relationship begins to be entangled with other image properties (*e.g.,* the color or shape) when applied to a cookie emoji in lower dimensions.

How does an attribute vector hold up across latent space variants?

Jane notices that Microsoft emojis have a distinctive style: most have a thick, black outline stroke. In contrast, Twitter emojis lack such outline. She defines an attribute vector between smileys from these two platforms (Figure 9). She then applies the attribute vector (T4) to various samples to verify that this relationship, namely adding an outline stroke, holds in different locations inside the latent space (S1). In the 32-dimensional space that she starts with, this attribute vector successfully transforms examples as expected.

She then uses this attribute vector to evaluate other latent spaces. She switches to 16-, 8-, and 4-dimensional spaces and applies the attribute vector to the same example, *Cookie*. She observes progressive degradation in interpolation results (T2) as the dimensionality decreases. In the 16-dimensional space, despite correctly adding the outline stroke, the transition affects other irrelevant image properties: the reconstructed cookies fail to maintain a constant color. The 8-dimensional space confuses the black stroke with surrounding shadows, and the 4-dimensional space even produces a rectangle. Note that in these latent spaces, the averaged attribute vector itself still works reasonably well, suggesting that the lower-dimensional spaces capture the relationship locally but can not allocate an orthogonal linear dimension for encoding this relationship.

6.2. Case Study: Cancer Transcriptomes

A meaningful latent space learned by unsupervised algorithms could be a powerful tool to assist scientific discovery [CN17]. Way and Greene [WG18] analyze a latent space of gene expression data for cancer patients, fit using a VAE, hoping to discover new biological pathways. One of their goals is to find the genes most differentially expressed in high grade serous ovarian cancer subtypes. To achieve this goal, they compute the average of samples in each cancer subtype and subtract the averages of two subtypes to obtain an attribute vector. They then identify the *single* latent dimension corresponding to the largest attribute vector component, inspect the associated decoder weights, and perform subsequent analysis. The fixation on individual latent dimensions is problematic, as simply

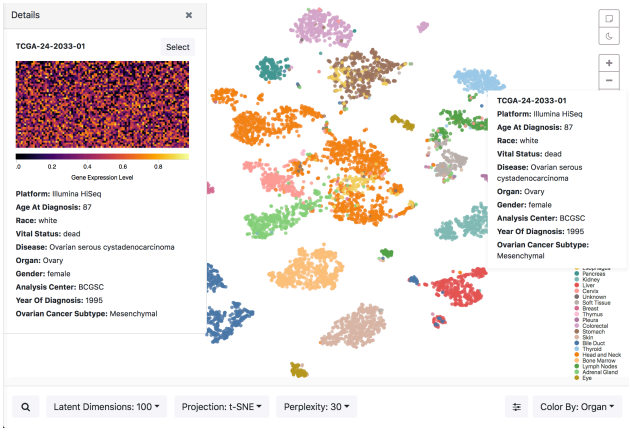


Figure 10: Re-analyzing a latent space of gene expression profiles of cancer patients, fit using a VAE by [WG18].

rotating the latent space and updating the decoder will produce identical results, yet with different latent space dimensions — latent space dimensions may be arbitrarily oriented.

Here, we present how LSC can support interpreting biological latent spaces to understand data and assist scientific discovery (Figure 10). We first define an attribute vector connecting samples in two ovarian cancer subtypes, *differentiated* and *proliferative*. We then view the projection onto the attribute vector axis (T6). We use the filter feature to include only relevant samples and change the y-axis of the projection to a categorical field that labels ovarian cancer subtypes (Figure 11a). In the updated projection (Figure 11b), we confirm that samples in differentiated and proliferative subtypes are well separated along the attribute vector axis. We repeat this procedure for the other two subtypes, *mesenchymal* and *immunoreactive*. Following the workflow of LSC naturally circumvents the pitfall of fixating on individual latent axes.

As the input to the VAE are gene expression profiles with 5,000 dimensions, LSC uses methods to visualize abstract vector input. To show details for a specific patient, LSC renders a heatmap of gene expression levels (Figure 10). In the attribute vector view, LSC lists the genes that are most differentially expressed in one concept versus another (T4, Figure 11c). We obtain this list using quantile-based thresholding, selecting genes 2.5 standard deviations from the mean (details in supplemental material). LSC can export this gene list (Figure 11d) in CSV format for further analysis.

We examined the overlap between our resulting gene lists and those in prior work [WG18]. The agreement is poor, including multiple cases with a null intersection. We performed pathway analyses on the gene list and the results show interesting differences, pointing to new biological pathways of potential relevance to cancer subtypes. As we do not have resources to further confirm the results, we shared our analysis with the authors of [WG18] to obtain their feedback. These domain experts agreed that our analysis approach using LSC is correct and responded enthusiastically to the tool.

6.3. Case Study: Word Embeddings

We now demonstrate two analysis scenarios for word embeddings. We first show that with a few simple interactions, our sys-

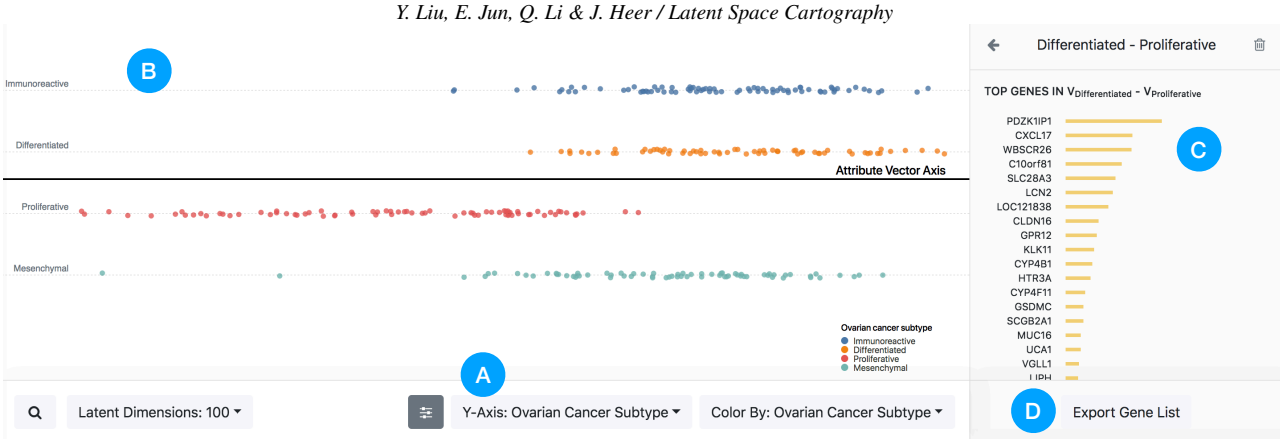


Figure 11: Analysis of an attribute vector between two high grade serous ovarian cancer subtypes, *differentiated* and *proliferative*. We change the y-axis (a) of the projection and confirm that samples in both subtypes are well-separated along the attribute vector axis (b). The list on the right (c) shows genes that are most differentially expressed in one subtype versus another, which can be exported (d) for further analysis.

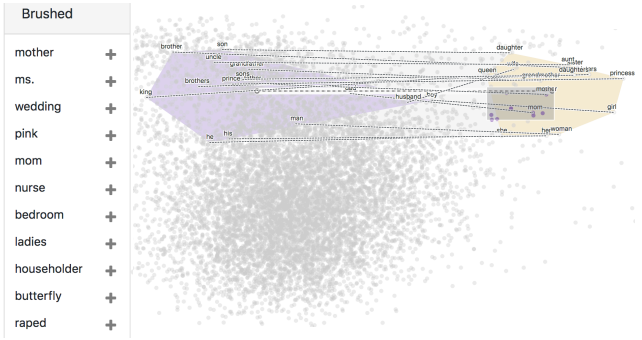


Figure 12: Gender bias in word embeddings. Words are projected onto an attribute vector for gendered names. Brushing the female concept convex hull reveals words that reflect gender stereotypes.

tem reproduces insights into gender stereotypes in word embeddings [BCZ*16a]. We then turn to the analysis of Google's analogy benchmark, and discuss how the quantitative test scores might obscure important nuances in an embedding. In both scenarios, we analyze the 10,000 most frequent words in the 50-, 100- and 300-dimensional pre-trained GloVe embeddings [PSM14].

6.3.1. Gender Biases in Word Embeddings

We first demonstrate how users might use LSC to quickly replicate findings on gender stereotypes in word embeddings. Bolukubasi *et al.* [BCZ*16a] quantify which words are closer to *he* versus *she* in the embedding space. They compile a list of profession names and project them onto two gender axes. We similarly define an attribute vector from word pairs in the *family* category of Google's analogy benchmark, including *king:queen*, *son:daughter* and *uncle:aunt*. We then view the attribute vector projection (T6).

Next, we brush corresponding regions in the projected space to explore words associated with each extreme of the attribute vector. Figure 12 shows words within the brushed region inside the convex hull of the *female* concept. Besides words that are gendered by definition (*e.g., mother*), other words (*e.g., pink*) reveal implicit stereotypes in the training corpus, as these words are considered similar due to frequent co-occurence with female names. Similarly, brushing the region around the *male* convex hull produces stereotyped words including *director*, *mayor*, *victory*, and *hero*. Words located toward each horizontal extreme but far away from the attribute vector region are strongly associated with each gender, but less relevant, for example male and female given names. While the previous approach [BCZ*16a] involved manually choosing words prior to visualizing their distribution, LSC allows us to easily explore and identify interesting words from the visualized corpus. The design choice to visualize attribute vectors on top of the projected data points (S1) facilitates quick discovery of relevant words.

The projected view provides additional insights. The mass of all words is unbalanced as the majority of the words are shifted toward the *male* concept. This imbalance suggests a prevalence of accounts associated with *male* in the training data, again indicating bias.

6.3.2. Analysis of Analogy Benchmark

We now turn to the analysis of Google's analogy dataset [MYZ13], one of the most widely used benchmarks for evaluating word embeddings. The dataset contains 14 groups, some of which are *syntactic* (*e.g.,* verbs and superlatives) while others assess *semantic* relationships (*e.g.,* countries and capitols). Each group contains several dozen pairs of words. The test works by systematically taking two pairs, performing attribute vector arithmetic on the first three words, and assessing if the fourth word is the top nearest neighbor. For example, given the pairs *king:queen* and *son:daughter*, the test computes $v = vec(king) - vec(son) + vec(queen)$ and counts the answer as correct if $vec(daughter)$ is the nearest neighbor of v. Typically, people exclude words already in the query (*i.e., king*, *queen* and *son* in the previous example) during the nearest neighbor search. The test exhausts all possible combinations of pairs and outputs the percentage of correct answers as a final score.

We use words in these analogy groups to define attribute vectors in LSC. Figure 13 shows an attribute vector for the group containing present tense verbs and participles. We view the projection onto the attribute vector axis (T6) and the distribution of pairwise cosine similarity (S2) in multiple dimensions. In the 50-dimensional embedding, the attribute vector appears less consistent visually: the convex hulls overlap and the pairs are less parallel compared to the other embeddings. The pairwise cosine distance plot further confirms this observation, as the effect size is smaller.

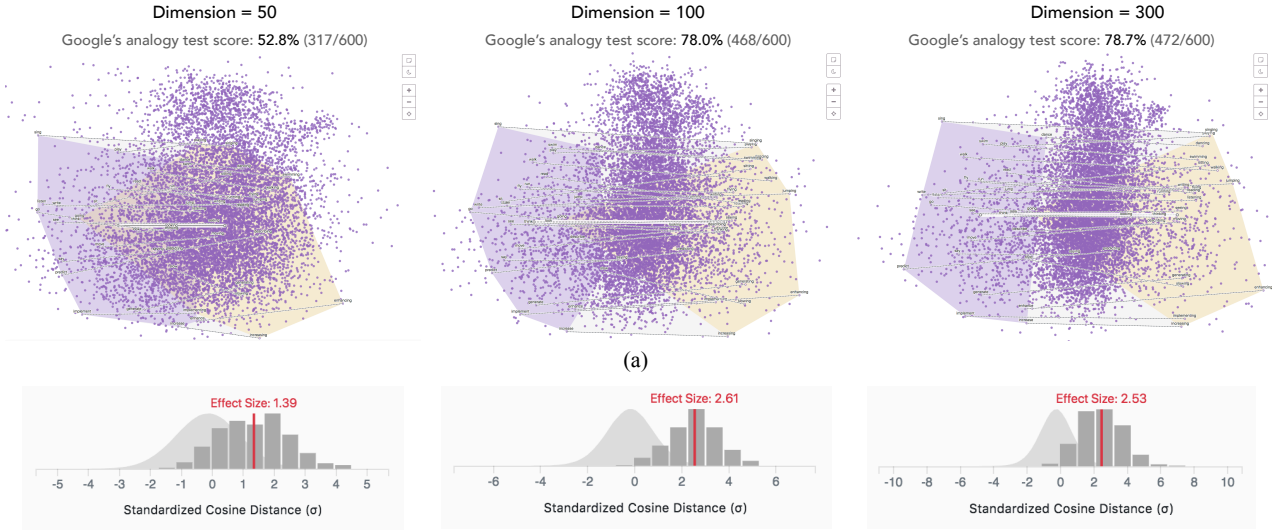


Figure 13: An attribute vector defining present tense verbs and participles in Google's analogy test dataset. Both the appearance of projected pairs (a) and the pair alignment statistics (b) agree with the analogy test scores.

Original		Answer	
1.000	slow	0.962	slow
0.874	fast	0.845	fast
0.828	faster	0.842	pace
0.825	slower	0.817	slowed
0.823	pace	0.817	slower
0.804	steady	0.807	steady
0.794	coming	0.781	turning
0.793	moves	0.779	driven
0.792	turning	0.771	faster
0.789	quick	0.766	slowing
0.784	difficult	0.762	slowly
0.782	turn	0.752	sluggish
0.774	harder	0.748	sharp
0.769	trouble	0.748	rapid
0.769	recovery	0.746	recovery
0.767	slowed	0.742	trouble
0.766	shift	0.740	weak
0.763	start	0.735	slide
0.763	hard	0.733	extremely
0.760	break	0.732	coming

(a)

Original		Answer	
1.000	dance	0.971	dance
0.907	dancing	0.909	dancing
0.872	singing	0.868	singing
0.853	music	0.836	musical
0.839	musical	0.826	music
0.814	dancers	0.795	dancers
0.798	hop	0.778	ensemble
0.788	singers	0.777	performing
0.788	pop	0.761	folk
0.781	performing	0.759	dancer
0.780	folk	0.757	hop
0.775	ensemble	0.756	performed
0.774	performed	0.750	pop
0.772	concert	0.748	singers
0.767	performers	0.746	concert
0.760	songs	0.736	performers
0.759	musicians	0.726	piano
0.750	jazz	0.726	ballet
0.748	bands	0.725	featured
0.746	song	0.720	performances

(b)

Figure 14: Original nearest neighbors of (a) *slow* and (b) *dance*, and nearest neighbors after adding the *present:participle* attribute vector (in the "answer" column).

Both the visual and statistical evaluation of attribute vector quality correlate with analogy test scores. As our quality assessments indicate how consistent individual pairs are in direction, they imply that the analogy test really measures linear regularities. Together, these analyses demonstrate the utility of our novel features on assessing relationship saliency (S2).

However, interactively exploring individual words within the attribute vector show interesting cases that are overlooked by the automated test procedure. In the following examples, we select a word, apply the *present:participle* vector (T4), and examine the nearest neighbors (T3) of both the original word and the answer. As shown in Figure 14a, the original neighborhood of *slow* does not contain the expected answer *slowing*. Applying the attribute vector brings *slowing* into the neighborhood, but because of other interfering words, it does not appear as the top choice. In contrast, *dancing* is already the first nearest neighbor of *dance* (Figure 14b). Applying the attribute vector has an insignificant effect on *dancing*, but it will be counted as correct by the automated test. Another example (figure in supplemental material) involves *sing* and *singing*. *Singing* is originally the 3rd nearest neighbor of *sing*. Adding the attribute vector improves its cosine similarity by 9% but does not change its rank in the neighborhood, so this case will be counted as incorrect. However, reversing the direction (*i.e.,* subtract the attribute vector from *singing*) makes the case correct because *sing* is the 1st nearest neighbor excluding *singing*.

These examples demonstrate that the analogy test does not depend solely on the "strength" of the linear relationship, but is confounded by the query word's neighborhood. They further imply that a summary score produced by the automated test oversimplifies potential issues and obscures interesting nuances. Though the analogy test result is a useful approximation of the quality of a word embedding, interactive visual analysis enables additional insights.

7. Conclusion and Future Work

In this paper, we contribute methods for mapping meaningful semantic dimensions of latent spaces. We surveyed the literature across a range of research communities, brought together common interpretation tasks, and integrated them in the LSC visual analysis system. In addition to its general utility, LSC contributes novel methods, including linear projection strategies to provide context, visual and statistical methods to assess attribute vector uncertainty, and techniques for comparing attribute vectors. With the support of our system, we challenged existing scientific findings on cancer gene expression, and shed light on nuances overlooked by the state-of-the-art word analogy benchmark in NLP.

Looking forward, our system could be extended to align the latent space to semantic axes. Currently, our system enables users to quickly discover relationships, define them as attribute vectors, verify if they manifest as consistent linear structures, and assess their quality. Given a set of orthogonal attribute vectors that encode separate causal factors underlying data variations, we might use the vectors as a basis to re-orient the space. This approach is inline with the goal of *disentangled representation* [BCV13], where the latent space allocates a separate dimension for each semantic attribute.

8. Acknowledgements

We thank Gregory Way, Casey Greene, Lang Liu, members of the UW Interactive Data Lab, and the anonymous reviewers for their helpful feedback. This work was supported by a Moore Foundation Data-Driven Discovery Investigator Award.

References

[ADvdH17] ABBASNEJAD M. E., DICK A., VAN DEN HENGEL A.: Infinite variational autoencoder for semi-supervised learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 781–790. 3

[ArX] Arxiv.org e-print archive. https://arxiv.org/ Accessed: 2018-12-01. 3

[BCV13] BENGIO Y., COURVILLE A., VINCENT P.: Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence 35*, 8 (2013), 1798–1828. 10

[BCZ*16a] BOLUKBASI T., CHANG K.-W., ZOU J., SALIGRAMA V., KALAI A.: Quantifying and reducing stereotypes in word embeddings. *arXiv preprint arXiv:1606.06121* (2016). 3, 9

[BCZ*16b] BOLUKBASI T., CHANG K.-W., ZOU J. Y., SALIGRAMA V., KALAI A. T.: Man is to computer programmer as woman is to homemaker? Debiasing word embeddings. In *Advances in Neural Information Processing Systems* (2016), pp. 4349–4357. 3

[BTB14] BRUNI E., TRAN N.-K., BARONI M.: Multimodal distributional semantics. *Journal of Artificial Intelligence Research 49* (2014), 1–47. 2, 5

[CLNL87] CARR D. B., LITTLEFIELD R. J., NICHOLSON W., LITTLEFIELD J.: Scatterplot matrix techniques for large N. *Journal of the American Statistical Association 82*, 398 (1987), 424–436. 2

[CN17] CARTER S., NIELSEN M.: Using artificial intelligence to augment human intelligence. *Distill* (2017). https://distill.pub/2017/aia. 1, 3, 4, 5, 8

[CR74] CATMULL E., ROM R.: A class of local interpolating splines. In *Computer Aided Geometric Design*. Elsevier, 1974, pp. 317–326. 7

[DTD*18] DAI H., TIAN Y., DAI B., SKIENA S., SONG L.: Syntax-directed variational autoencoder for structured data. In *Proceedings of the International Conference on Learning Representations* (2018). 3

[Emo] Emojipedia: Home of emoji meanings. https://emojipedia.org/ Accessed: 2018-12-01. 5

[FGM*01] FINKELSTEIN L., GABRILOVICH E., MATIAS Y., RIVLIN E., SOLAN Z., WOLFMAN G., RUPPIN E.: Placing search in context: The concept revisited. In *Proceedings of the International Conference on World Wide Web* (2001), pp. 406–414. 2, 5

[FSBL17] FRONTERA-PONS J., SUREAU F., BOBIN J., LE FLOC'H E.: Unsupervised feature-learning for galaxy SEDs with denoising autoencoders. *Astronomy & Astrophysics 603* (2017), A60. 3

[GBWD*18] GÓMEZ-BOMBARELLI R., WEI J. N., DUVENAUD D., HERNÁNDEZ-LOBATO J. M., SÁNCHEZ-LENGELING B., SHEBERLA D., AGUILERA-IPARRAGUIRRE J., HIRZEL T. D., ADAMS R. P., ASPURU-GUZIK A.: Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science 4*, 2 (2018), 268–276. 3

[Gle13] GLEICHER M.: Explainers: Expert explorations with crafted projections. *IEEE Transactions on Visualization and Computer Graphics*, 12 (2013), 2042–2051. 3

[GPAM*14] GOODFELLOW I., POUGET-ABADIE J., MIRZA M., XU B., WARDE-FARLEY D., OZAIR S., COURVILLE A., BENGIO Y.: Generative adversarial nets. In *Advances in Neural Information Processing Systems* (2014), pp. 2672–2680. 1

[HG18] HEIMERL F., GLEICHER M.: Interactive analysis of word vector embeddings. In *Computer Graphics Forum* (2018), vol. 37, pp. 253–265. 2, 3

[HKPC18] HOHMAN F. M., KAHNG M., PIENTA R., CHAU D. H.: Visual analytics in deep learning: An interrogative survey for the next frontiers. *IEEE Transactions on Visualization and Computer Graphics* (2018), 1–1. 3

[HLJ16] HAMILTON W. L., LESKOVEC J., JURAFSKY D.: Diachronic word embeddings reveal statistical laws of semantic change. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics* (2016), vol. 1, pp. 1489–1501. 1, 3

[HNP17] HADJERES G., NIELSEN F., PACHET F.: GLSR-VAE: Geodesic latent space regularization for variational autoencoder architectures. In *IEEE Symposium Series on Computational Intelligence* (2017), pp. 1–7. 3

[HRK15] HILL F., REICHART R., KORHONEN A.: Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics 41*, 4 (2015), 665–695. 2, 5

[HSSQ17] HOU X., SHEN L., SUN K., QIU G.: Deep feature consistent variational autoencoder. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision* (2017), pp. 1133–1141. 3

[ID90] INSELBERG A., DIMSDALE B.: Parallel coordinates: A tool for visualizing multi-dimensional geometry. In *Proceedings of the 1st Conference on Visualization* (1990), pp. 361–378. 2

[JBJ18] JIN W., BARZILAY R., JAAKKOLA T.: Junction tree variational autoencoder for molecular graph generation. In *Proceedings of the International Conference on Machine Learning* (2018), vol. 80, pp. 2323–2332. 3

[Jol11] JOLLIFFE I.: Principal component analysis. In *International Encyclopedia of Statistical Science*. Springer, 2011, pp. 1094–1096. 2

[JSL*17] JOHNSON M., SCHUSTER M., LE Q. V., KRIKUN M., WU Y., CHEN Z., THORAT N., VIÉGAS F., WATTENBERG M., CORRADO G., HUGHES M., DEAN J.: Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Annual Meeting of the Association for Computational Linguistics 5* (2017), 339–351. 3

[JYY*16] JI S., YUN H., YANARDAG P., MATSUSHIMA S., VISHWANATHAN S. V. N.: WordRank: Learning word embeddings via robust ranking. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (2016), pp. 658–668. 3

[JZS17] JAIN U., ZHANG Z., SCHWING A. G.: Creativity: Generating diverse questions using variational autoencoders. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 5415–5424. 3

[KCPE16] KIM H., CHOO J., PARK H., ENDERT A.: InterAxis: Steering scatterplot axes via observation-level interaction. *IEEE Transactions on Visualization and Computer Graphics 22*, 1 (2016), 131–140. 3

[KMRW14] KINGMA D. P., MOHAMED S., REZENDE D. J., WELLING M.: Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems* (2014), pp. 3581–3589. 3

[KPHL17] KUSNER M. J., PAIGE B., HERNÁNDEZ-LOBATO J. M.: Grammar variational autoencoder. In *Proceedings of the International Conference on Machine Learning* (2017), vol. 70, pp. 1945–1954. 3

[KW13] KINGMA D. P., WELLING M.: Auto-encoding variational Bayes. In *Proceedings of the International Conference on Learning Representations* (2013). 1, 2

[LBT*18] LIU S., BREMER P.-T., THIAGARAJAN J. J., SRIKUMAR V., WANG B., LIVNAT Y., PASCUCCI V.: Visual exploration of semantic relationships in neural word embeddings. *IEEE Transactions on Visualization and Computer Graphics 24*, 1 (2018), 553–562. 2, 3

[LNH*18] LI Q., NJOTOPRAWIRO K. S., HALEEM H., CHEN Q., YI C., MA X.: EmbeddingVis: A visual analytics approach to comparative network embedding inspection. *IEEE Transactions on Visualization and Computer Graphics* (2018), 1–1. 2, 3

[LSC*18] LIU M., SHI J., CAO K., ZHU J., LIU S.: Analyzing the training processes of deep generative models. *IEEE Transactions on Visualization and Computer Graphics 24*, 1 (2018), 77–87. 3

[LSL*17] LIU M., SHI J., LI Z., LI C., ZHU J., LIU S.: Towards better analysis of deep convolutional neural networks. *IEEE Transactions on Visualization and Computer Graphics 23*, 1 (2017), 91–100. 3

[MDP*11] MAAS A. L., DALY R. E., PHAM P. T., HUANG D., NG A. Y., POTTS C.: Learning word vectors for sentiment analysis. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics* (2011), vol. 1, pp. 142–150. 2

[MH18] MCINNES L., HEALY J.: UMAP: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426* (2018). 2

[MKS*15] MNIH V., KAVUKCUOGLU K., SILVER D., RUSU A. A., VENESS J., BELLEMARE M. G., GRAVES A., RIEDMILLER M., FIDJELAND A. K., OSTROVSKI G., ET AL.: Human-level control through deep reinforcement learning. *Nature 518*, 7540 (2015), 529. 3

[MNG17] MESCHEDER L., NOWOZIN S., GEIGER A.: Adversarial variational Bayes: Unifying variational autoencoders and generative adversarial networks. In *Proceedings of the International Conference on Machine Learning* (2017), vol. 70, pp. 2391–2400. 3

[MSC*13] MIKOLOV T., SUTSKEVER I., CHEN K., CORRADO G. S., DEAN J.: Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems* (2013), pp. 3111–3119. 2

[MSSW16] MAALØE L., SØNDERBY C. K., SØNDERBY S. K., WINTHER O.: Auxiliary deep generative models. In *Proceedings of the International Conference on Machine Learning* (2016), vol. 48, pp. 1445–1454. 3, 4, 5

[MYZ13] MIKOLOV T., YIH W.-T., ZWEIG G.: Linguistic regularities in continuous space word representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (2013), pp. 746–751. 2, 5, 9

[PHL*16] PEZZOTTI N., HÖLLT T., LELIEVELDT B., EISEMANN E., VILANOVA A.: Hierarchical stochastic neighbor embedding. In *Computer Graphics Forum* (2016), vol. 35, pp. 21–30. 2

[PHVG*18] PEZZOTTI N., HÖLLT T., VAN GEMERT J., LELIEVELDT B. P., EISEMANN E., VILANOVA A.: DeepEyes: Progressive visual analytics for designing deep neural networks. *IEEE Transactions on Visualization and Computer Graphics 24*, 1 (2018), 98–108. 3

[PLVDM*17] PEZZOTTI N., LELIEVELDT B. P. F., VAN DER MAATEN L., HÖLLT T., EISEMANN E., VILANOVA A.: Approximated and user steerable tSNE for progressive visual analytics. *IEEE Transactions on Visualization and Computer Graphics 23*, 7 (2017), 1739–1752. 2

[PMH*18] PEZZOTTI N., MORDVINTSEV A., HÖLLT T., LELIEVELDT B. P. F., EISEMANN E., VILANOVA A.: Linear tSNE optimization for the web. *arXiv preprint arXiv:1805.10817* (2018). 2

[PSM14] PENNINGTON J., SOCHER R., MANNING C. D.: GloVe: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (2014), pp. 1532–1543. 2, 9

[RAL*17] REN D., AMERSHI S., LEE B., SUH J., WILLIAMS J. D.: Squares: Supporting interactive performance analysis for multiclass classifiers. *IEEE Transactions on Visualization and Computer Graphics 23*, 1 (2017), 61–70. 3

[RMC15] RADFORD A., METZ L., CHINTALA S.: Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434* (2015). 2, 4

[SBMN13] SOCHER R., BAUER J., MANNING C. D., NG A. Y.: Parsing with compositional vector grammars. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics* (2013), vol. 1, pp. 455–465. 2

[SRM*16] SØNDERBY C. K., RAIKO T., MAALØE L., SØNDERBY S. K., WINTHER O.: Ladder variational autoencoders. In *Advances in Neural Information Processing Systems* (2016), pp. 3738–3746. 3

[STN*16] SMILKOV D., THORAT N., NICHOLSON C., REIF E., VIÉGAS F. B., WATTENBERG M.: Embedding projector: Interactive visualization and interpretation of embeddings. In *NIPS Workshop on Interpretable ML in Complex Systems* (2016). 2, 3

[TRB10] TURIAN J., RATINOV L., BENGIO Y.: Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics* (2010), pp. 384–394. 2

[TWR*17] TRAN D. L., WALECKI R., RUDOVIC O., ELEFTHERIADIS S., SCHULLER B. W., PANTIC M.: DeepCoder: Semi-parametric variational autoencoders for automatic facial action coding. In *Proceedings of the IEEE International Conference on Computer Vision* (2017), pp. 3190–3199. 3

[UFDR16] UPADHYAY S., FARUQUI M., DYER C., ROTH D.: Cross-lingual models of word embeddings: An empirical comparison. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics* (2016), vol. 1, pp. 1661–1670. 3

[VDM14] VAN DER MAATEN L.: Accelerating t-SNE using tree-based algorithms. *The Journal of Machine Learning Research 15*, 1 (2014), 3221–3245. 2

[VdMH08] VAN DER MAATEN L., HINTON G.: Visualizing data using t-SNE. *Journal of Machine Learning Research 9* (2008), 2579–2605. 2

[VdMH12] VAN DER MAATEN L., HINTON G.: Visualizing non-metric similarities in multiple maps. *Machine learning 87*, 1 (2012), 33–55. 2

[Wet17] WETZEL S. J.: Unsupervised learning of phase transitions: From principal component analysis to variational autoencoders. *Physical Review E 96*, 2 (2017), 022140. 3

[WG18] WAY G. P., GREENE C. S.: Extracting a biologically relevant latent space from cancer transcriptomes with variational autoencoders. In *Proceedings of Pacific Symposium on Biocomputing* (2018), vol. 23, pp. 80–91. 2, 3, 8

[WSW*18] WONGSUPHASAWAT K., SMILKOV D., WEXLER J., WILSON J., MANÉ D., FRITZ D., KRISHNAN D., VIÉGAS F. B., WATTENBERG M.: Visualizing dataflow graphs of deep learning models in tensorflow. *IEEE Transactions on Visualization and Computer Graphics 24*, 1 (2018), 1–12. 3

[YHSBK17] YANG Z., HU Z., SALAKHUTDINOV R., BERG-KIRKPATRICK T.: Improved variational autoencoders for text modeling using dilated convolutions. In *Proceedings of the International Conference on Machine Learning* (2017), vol. 70, pp. 3881–3890. 3

[YSD*18] YAO Z., SUN Y., DING W., RAO N., XIONG H.: Dynamic word embeddings for evolving semantic discovery. In *Proceedings of the ACM International Conference on Web Search and Data Mining* (2018), pp. 673–681. 3

[YYSL16] YAN X., YANG J., SOHN K., LEE H.: Attribute2Image: Conditional image generation from visual attributes. In *Proceedings of the European Conference on Computer Vision* (2016), pp. 776–791. 3

[ZF14] ZEILER M. D., FERGUS R.: Visualizing and understanding convolutional networks. In *Proceedings of the European Conference on Computer Vision* (2014), pp. 818–833. 3

[ZSE17] ZHAO S., SONG J., ERMON S.: Learning hierarchical features from generative models. In *Proceedings of the International Conference on Machine Learning* (2017), vol. 70, pp. 4091–4099. 3