Eurographics Conference on Visualization (EuroVis) 2022
R. Borgo, G. E. Marai, and T. Schreck
(Guest Editors)

Volume 41 (2022), Number 3

PROVEE: A Flexible System Architecture for a Progressive Exploration of Embedding Spaces

Simen van Herpt[1] & Dennis Owolabi[1] & Sinie van der Ben[1] & Dong Nguyen[1] & Michael Behrisch[1]

[1]Utrecht University, Netherlands

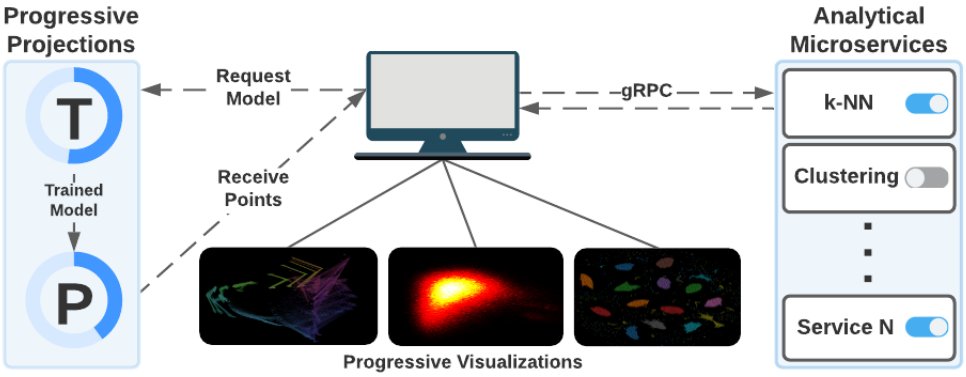


Figure 1: **Framework of PROVEE**. First, the **T**rainer process incrementally trains a dimension reduction model and sends it to the **P**rojection process. Second, while the trainer process continues, the projection process uses the received model to create low-dimensional points. In turn, these are used to render *progressive visualizations* and analytical microservices concurrently run analysis tasks on these visualizations.

Abstract

Progressive Analytics is an emerging data exploration paradigm that makes it possible to explore very large datasets in a fluent and iterative fashion. The core idea behind this paradigm is to split long computations into a succession of partial solutions arriving and improving with time. In this work, we present PROVEE, a flexible system architecture to support the development of novel visual, algorithmic, and analytical solutions for the Progressive Analytics paradigm. With PROVEE, we hope to drastically reduce the engineering-heavy setup work that needs to be done before a research contribution can even begin. PROVEE is specifically designed to support exploration of large embedding spaces. A key challenge on the visual side is supporting early exploration by displaying intermittent results and gradually displaying a more complex visualization. By reflecting on our choices for a microservice and contract-based architecture, we shine a light on the flexibility and extensibility implications of our engineering solution. Our evaluation approach is twofold: (1) We compare with state-of-the-art tools for reasoning in embedding spaces through a series of quantitative evaluations focusing on time, memory consumption, and scalability; (2) We demonstrate the potential of PROVEE's analytical features and extensibility through a case study conducted with researchers in the field of Natural Language Processing. We show that our system design allows for a fast and flexible engineering of novel Progressive Analytics techniques and is already more feature-rich than most commonly used embedding exploration tools.

CCS Concepts

• ***Human-centered computing*** → ***Visualization systems and tools; Visual analytics;*** *Information visualization;*

1 Introduction

The Information Paradox claims that *"we are drowning in information but starved for knowledge"* [NB83]. Although new hardware and software systems are constantly tweaked to endure an ever-growing demand in terms of storage, management and computational power, the support for exploratory data analysis is more and more lagging behind.

The field of machine learning has changed dramatically with the emergence of deep neural networks that achieve state-of-the-art performance on a variety of tasks [BLH21]. Deep neural networks automatically learn to represent objects as continuous, dense

vector representations, also called embeddings. Each dimension represents a latent feature of the object. For example, in Natural Language Processing (NLP), such embeddings can represent linguistic units, such as words, sentences, paragraphs or documents [DCLT19, Smi20]. Despite the popularity of deep neural networks, they have severe limitations: They are difficult to interpret [GMR*18], can reflect and even reinforce societal biases [MMS*21], and sometimes take shortcuts rather than actually modelling the problem of interest [GJM*20]. Because of the central role of embeddings, the analyses and interpretation of embeddings has received increasing interest. For example, the analysis of embedding spaces can reveal biases [BCZ*16] and shed light on neural network training dynamics [RGYSD17].

To explore and analyze embedding spaces, visualization is an important tool for the researcher [BG19]. However, a major challenge is that the human's cognitive constraints impact the knowledge generation process in Visual Analytics [MSA*19, SSS*14]. Analysts require a *fluent* and *adaptable* human-computer dialogue that enables them to flexibly change their querying behavior, visualization parameter settings, or algorithmic choices without having to wait for minutes until long computations finish.

In recent years, Progressive Visual Analytics (PVA) gained traction as a comprehensive solution for many data-intensive problems [SPG14, FP16, SASS16, ASSS18, MSA*19]. It differs from existing (machine learning) concepts related to progressiveness, such as online [BE98], streaming [Mut05], and iterative computations [Saa03] by incorporating the user into the loop during otherwise long-running and non-transparent computations by producing intermediate partial results [ASSS18].

However, as the recent Dagstuhl seminar report on Progressive Data Analysis and Visualization states: *"The widespread use of progressive data analysis has been hampered by a chicken-and-egg problem: data visualization researchers do not have online database systems to work against, and database researchers do not have tools that will display the results of their work"* [FFNS19]. Unlike important work that focus on analytical/computational [WM04, PLvdM*17, MPG*14] or (uncertainty) visualization questions [TKBH17, Fis11, FPDmcs12], we aim to address the system engineering aspect of the aforementioned problem by providing a flexible, yet generic, architectural framework on which to build and experiment with future PVA solutions. We want to relieve researchers from the engineering-heavy setup work that needs to be done before a research contribution in PVA can even begin. In this paper, we present PROVEE, a flexible system architecture for a Progressive Exploration of Embedding spaces [BTN*21]. We investigated, developed, and refined PROVEE over the course of a two-year collaboration with researchers from the field of NLP.

PROVEE features a *decoupling* of model computation, e.g., incremental dimensionality reduction, and model consumption, e.g, visualization rendering of 2D projections. This deeply ingrained *separation-of-concerns* design principle allows us to load, project, visualize, and interactively analyze datasets which normally wouldn't fit in memory. Using this method, the maximum number of high-dimensional points which can be projected, won't be bottlenecked as severely by the size of the original dataset. Furthermore, we demonstrate a microservice and contract-based component architecture within PROVEE. This flexible component solution allows us to add and remove analytical features without affecting the overall progressive system architecture. For example, a kNN service can be plugged in when needed. Since this requires a lot of RAM memory, it can easily be disabled when more memory is required for other systems. To demonstrate PROVEE's extensibility capabilities, we implemented several analytical features, like k-nearest neighbor search and vector arithmetic on the latent space, in a progressive embedding analysis scenario.

To be of practical use to the wider Visual Analytics community, our system paper contribute a critical reflection of engineering and abstraction design choices, such as "how can services communicate efficiently" and "which microservice should store which information", and the limitations of our approach. Moreover, we outline several novel progressive data pipeline concepts, showing how an alternative decoupling of model generation and consumption could effectively support the exploratory and analytical process and give rise to a range of future works.

We evaluated PROVEE in a series of experiments. First, although we are not making technical (novel algorithm) contributions per se (c.f., technique versus system paper comparison in [Mun08]), we conducted a series of performance evaluations focusing on time, memory consumption, and scalability to investigate the technical limits of our architecture on current hardware. Second, we explored the responsiveness, practicability and extensibility of our analytical features through a case study conducted with researchers in NLP. And, third, we conducted a comparative evaluation with state-of-the-art tools for reasoning in embedding spaces. We found that our system design allows for a fast and flexible engineering of novel PVA techniques and is already now more feature-rich than most commonly used embedding exploration tools.

2 Related work

In this section, we outline a selection of works that led to the development of our unifying system architecture for supporting Progressive Visual Analytics research and system development.

Progressive Visual Analytics (PVA): Increasingly large datasets, complex computations, and the expectation towards fluent interactiveness are challenging for existing Visual Analytics (VA) systems [KMS*08]. To meet these demands a progressive approach is needed, where meaningful partial results are shown to users before the computation is finished [AS13, MSA*19]. Stolper et al. define Progressive Visual Analytics (PVA) as a *"progressive process [that] produces a sequence of useful/meaningful intermediate results in contrast with pure iterative processes that inherently generate intermediate results whose utility is mainly related to monitoring, tuning, steering and explaining the process itself"* [FFNS19]. In PVA, intermediate outputs can be results, quality measures, or progress measures and their utility for the user has to increase during the sequence of visualization. While approaching the end of the visualization sequence, the intermediate outputs gradually resemble the final outputs. Micallef et al. found that displaying intermediate results too early in the process misleads users into the misinterpretation of patterns or falling for confirmation biases [MSA*19]. Consequently, feedback about the ongoing computations is key in PVA, as well as progressively improved results [FP16, GTE*20a].

Two orthogonal data management approaches for PVA have been presented [FP16, ASSS18]: divide the computation into steps and dividing the data into chunks. For example, in ProgressiVis [FP16] a scheduler keeps track of the computation progress and run time for all assigned modules, i.e., the computational units that need to deliver results in periodic intervals. If the compute module is not finished with its calculation after a predefined update interval, it is pushed back to a queue and the next module continues its operation. More interesting is the second "interactive" scheduling mode implemented into ProgressiVis: Whenever settings of interaction parameters changes, the compute modules have to pause, adapt internally, and continue their computation according to the new parameter settings. Another system in this category is PIVE, short for Per-Iteration Visualization Environment, presented by Choo et al. [CLK*14, KCL*17] or the prototype demonstrated by Schulz et al. in [SASS16]. Alternatively, data batching approaches, such as in DimXplorer [TKBH17], ensure temporal constraints by processing large chunks of data piece-by-piece in a sequential manner. To adapt to the user interactivity, PVA systems in this category use adaptive batch sampling strategies, predictive caching [CXGH08] or aggregated binning [LJH13]. With PROVEE, we give flexibility for both approaches, providing an extendable framework. We discuss these system choices more comprehensively in Section 5.

Progressive Machine Learning: A plethora of concepts for dealing with progressive algorithms have been presented in the past, mostly driven by the machine learning and database communities. Terms such as *online* [BE98, HHW97, Alb03], *incremental* [Car02, THSW15], or *iterative* [Saa03], or *streaming* [PVG*11] computations subsume approaches for designing or re-thinking algorithmic designs to serve the progressive computation idea. The interested reader should be deferred to the work of Fekete and Primet, who give in [FP16] a more detailed summary of the mentioned computation approaches. Independent of the terms used for describing the concept, PROVEE provides a flexible component architecture allowing researchers and system engineers to incorporate a wide range of machine learning and visualization approaches.

Visualization Latency in PVA systems: Visualization of complex computations on larger datasets can lead to latency in the display of results [SPG14]. A too long latency leads to an interrupted focus of the user and the likely urge to perform other tasks while waiting for the computations to finish [Mil68]. This is why a so-called *bounded latency* is essential for PVA. Latencies of at most 10 seconds are crucial to keep the user engaged in the task. These 10 seconds are considered as attention preserving latency [FP16] and date back to research of Miller [Mil68] and Shneiderman [Shn84]. Accordingly, approaches to implement PVA, like in PROVEE, strongly rely on multi-threading architectures, where we separate the visualization thread from the main application thread [MPG*14, SASS16]. Separating these two processes allows for a concurrent handling of large datasets, allows for progressive provision of approximate visualizations, but also requires a thorough event handling mechanism to respond to the human's interactive input.

Progressive Embeddings: Interaction with progressively provided information is an important aspect of Visual Analytics in general, but especially for systems in which users need to understand the relationships between (manipulated) parameters and (changed) results. Several works have addressed this VA problem for high-dimensional projection algorithms and the exploration of latent space embeddings. To progressively produce these high-dimensional projections, William and Munzner presented MD-Steer, an user-adaptive Multidimensional Scaling (MDS) computation. Their approach uses hierarchical data structures and progressive layout computation to steer the computation towards user-defined areas of interest in the dataset [WM04]. A similar goal is pursued by Pezzotti et al. providing an approximated and user-steerable t-SNE algorithm [PLvdM*17] or Ko et al. targeting a progressive Uniform Manifold Approximation (UMAP) [KJS20].

Visual Analytics for Embeddings: Studies to understand embedding spaces often involve large datasets with high-dimensional embeddings (e.g. 1024 dimensions). We need to separate between visualization tools that allow exploration of embedding spaces through standard projection methods, such as PCA and t-SNE, and tools that allow users to analyze embedding space beyond these projection methods. We make this distinction because the result of the tools is different: providing visualization only gives the users freedom of own interpretation, while accommodating analysis methods provides the users with a more focused result. Existing tools such as Parallax [MWZ19], Whatlies [WKT20] and Tensorflow Embedding Projector (EP) [STN*16] are considered visualization tools. Parallax allows users to explicitly define the axes on which to project an embedding space [MWZ19], while Whatlies is a visualization library that incorporates the idea of explicit definition and adds 2D interactive charts for the exploration of local neighbourhoods [WKT20]. Interestingly, the Tensorflow EP allows for a local or remote computation of embedding projections displayed in 2D or 3D [STN*16]. Examples of visual analytical tools are the Embedding Comparatora, a visual interactive analysis tool to compare local neighbourhoods of embeddings [BCS19] and Latent space cartography, a VA system to explore relationships in the embedding space [LJLH19]. These are presented, e.g., by Turkay et al. [TKBH17], Angelini et al. [AS13], or Kim et al. in PIVE [CLK*14, KCL*17]. With PROVEE, we not only satisfy the basic computational aspects of PVA systems but also demonstrate how to enable advanced algorithmic functionality on top of this.

3 PROVEE's Architecture for Progressive Microservices

PROVEE's progressive analytics pipeline is structured as shown in Figure 2. It consists of a set of four cohesively grouped, but loosely coupled *microservices* (see Section 3.1); i.e. independent processes that execute isolated tasks. These microservices work incrementally by passing down partial output as they work on their partial input. This creates a stream of data flowing through the pipeline allowing for incremental visualization rendering, as well as progressive analysis tasks to increasingly supply information.

Central to PROVEE's analytic capabilities is the strong separation-of-concerns between model production/training and model consumption/inference. In our pipeline, these processes are happening simultaneously and are separated by thread- and/or microservice-boundaries. As one example, we allow the data projection to *consume* a partial model that is *produced* in a separate process. Whenever a updated model is ready, PROVEE will just

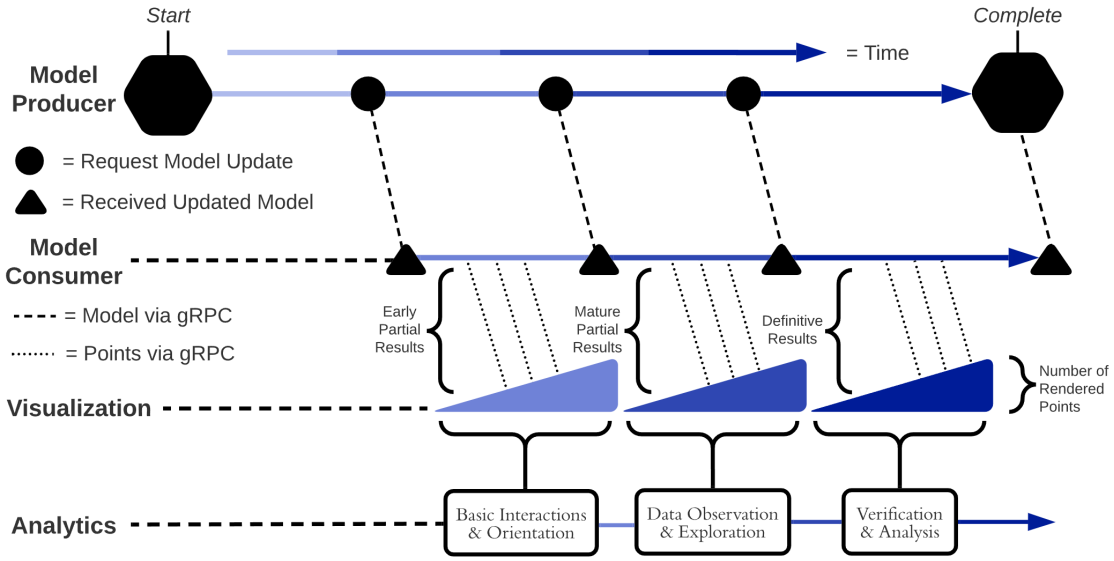


Figure 2: **PROVEE's pipeline structure:** Each of the four lines represents one of the four decoupled systems. As described in Section 3.1, the microservices build upon the partial output of other microservices in parallel, creating interactive incremental visualizations.

have to update the "model consumption pointer" to work with the new knowledge source and can then continue concurrently.

Whenever scalability is a focus point, even with sophisticated storage structures the use of systems can be hindered if features require all data to be stored. We go a different and more flexible route in PROVEE. In PROVEE's framework opt-in functionality is defined by optional microservices as described in Section 3.1, allowing users to disable more demanding microservices.

PROVEE's functionality is not fixed but can be adapted by flexibly adding, removing, and altering the pipeline's microservices. The only restriction during these changes is that microservices must comply with the **interservice API**, i.e. the Application Programming Interface, defined for each type of service. Our paper presents how a multitude of **features** can be created in this manner.

3.1 Microservice pipeline

We divide our PROVEE's pipeline implementation into four types of microservices: (1) Model Producer, (2) Model Consumer, (3) Visualization and (4) Analytics. The pipeline begins with dimension reduction on the embedding set, split into the phases Model Producer and Model Consumer.

Model Producer: The *Model Producer microservice* receives a precomputed embedding dataset and uses it to create a dimension reduction model. PROVEE uses incremental dimensionality projection techniques, such as the incremental PCA [RLLY08]. We facilitate progressive exploration by making intermediate versions of this model continuously available to the system. The power of PROVEE is that we allow different projection microservices, each with its own dimension reduction algorithm or parameter settings, to run in parallel. The user can freely decide which algorithm (and or setting) should be used by "activating" one model instantiation. Furthermore, due to the model production-consumption separation in the core framework, PROVEE naturally allows for substitution of dimension reduction models. This means that previously generated models can be loaded and used again when saved. Models can even be shared with others. In this way, users can continue where they left off without first needing to fit their model. Models can also be substituted for models trained on different datasets and incrementally trained further on the new dataset. These functionalities are lacking in other tools, as is shown in Table 3.

Model Consumer: In our typical workflow, the resulting (partial) models will be consumed by the *Model Consumer microservice* to create low-dimensional representations for high-dimensional embedding vectors. Like the Model Producer phase, this step happens incrementally by calculating the points in batches and passing them along the pipeline before continuing with the next batch. Different model consumption strategies are possible here, see 'Partial Fits' and 'Comparing Algorithms' in Section 5. By handling the processing in this manner, points are not unnecessarily stored and output is created earlier. By working with partially trained models, we can create low-dimensional representatives faster, albeit less accurately, i.e., our experimental validation shows latency improvements of up to 463% to comparable projection systems, such as TensorFlow's Embedding projector [STN*16]. When receiving a better model, the transformation can restart and create more accurate points using the new model. Because of this constant improvement of the model, the final difference in accuracy is less than 1% compared to the non-incremental batch-mode PCA algorithm [ZYK06]. In this way, initialisation time and memory usage are greatly reduced at the cost of little accuracy. When it is not possible to split the Model Producer and Model Consumer, such as e.g., in t-SNE [VdMH08], a specialized microservice can be implemented that handles the Model Producer and Model Consumer steps monolithically. However, in this case, intermediate results can still be used. If at an early stage the dimension reduction model already behaves similar to its

final version, results created after relatively few iterations can nevertheless be informative, as e.g. with progressive UMAP [KJS20].

Visualization: We visualize the low-dimensional points with the *Visualization microservice*, which is also concerned with user interaction and animations. When it receives new points, stemming from an updated model, it can either discard the old set of points or use them to visualize the difference between the models. Different experiments can call for different visualization methods; each specializes in their own way of embedding exploration. Therefore, interchangeable rendering methods are critical to the versatility of PROVEE. To showcase this ability we created multiple renderers, e.g., 2D and 3D renderer and a heatmap renderer. The heatmap renderer, shown in Figure 1, allows users to understand the underlying distribution, effectively mediating the overplotting issue, i.e., a large number of overlapping points [ED02, Ber11]. Because each renderer operates independently in PROVEE, they can be changed on-the-fly without affecting other rendering or analytical functionalities. However, they all operate in the same microservice to prevent redundantly storing low-dimensional points across multiple.

Analytics: In contrast to the preceding types of microservices, *Analytical microservices* do not have a fixed position in the pipeline, but are used as extensions. Their task can range from k-Nearest Neighbor (kNN) search to calculating projection accuracy depending on the application domain requirements. These microservices can be based on input from the embeddings dataset, other microservices, or both. In essence, they build new features upon the pipeline by using the data contained within. The incremental usage of the data-flow also applies to analytical microservices. For instance, local neighborhoods, see Section 3.3, can be displayed with intermediate projections. In this way, output can be delivered as quickly as possible. However, analytical microservices are often more memory consuming, because they require the high-dimensional embeddings. Therefore, in cases where the analytical microservices consume too much memory, they can effortlessly be disabled without impacting the projection functionalities, due to PROVEE's loose coupling. With versatile microservices like these, PROVEE aims to empower the community to conduct research in progressive analytics by providing it with a flexible and extendable framework in which novel experiments can be conducted.

3.2 Interservice communication with gRPC

Quintessential for the progressive processing pipeline outlined above are synchronized, as well as asynchronous, fast, predictable, and (in the long run) extendable communication paths between PROVEE and its microservice components. One challenge here is that these services might be hosted within one machine or even in the cloud. We therefore use a state-of-the-art, lightweight communication protocol, called gRPC [WZZ93]. gRPC (Group Remote Procedure Call) is Google's open-source RPC framework. It supplies (bi-directional) communication methods for distributed systems, as well as efficient serialization [WZZ93]. While this communication could also be achieved with HTTP or event-based communication schemes, gRPC creates a long-lasting link that pays off when intensely used [Lu20].

At its core, gRPC consists of *remote procedure calls*, i.e., a procedure (subroutine) will be transparently executed in a different address space (commonly on another computer on a shared network) [Wik21]. A "contract" has to be set up between the services, so that the clients know which functions they can call, their parameter requirements, and which return values to expect. These contracts consist of a server's services and data types, and are stored in .PROTO files, as shown for the projector microservice in Listing 1. These contracts were created to be versatile. For example, the models are transmitted in raw byte format. This allows the developer to serialize the model objects themselves, simplifying the implementation for the model producer and consumer.

Listing 1: Protocol buffer for Model Producer microservices. Protobuf descriptors allow us to connect services in a descriptive and incremental manner.

```
// Interface exported by the server.
service ModelProducer {
  rpc getModel(google.protobuf.Empty) returns (model) {}
  rpc getProgress(google.protobuf.Empty) returns (progressModelProducer) {}
  //... Start and stop procedures ...
}

message progressModelProducer{
  int32 progress = 1;
}

message model{   bytes model = 1;  }
```

Our choice for gRPC has both advantages and disadvantages compared to having all microservices in one process. Important for this choice is the fact that PROVEE has been implemented with Python, due to its extensive support and usage in the AI and VA fields [NG19]. However, Python has poor performance compared to other programming languages, primarily due to the fact that its core interpreter is inherently single threaded. It is, consequently, unfit for parallel computation-intensive microservice frameworks. One way to deal with this challenge is by using optimized libraries, such as Cython [BBC*11]. Alternatively, crucial sections could be optimized in a different programming language (such as C/C++) and using dedicated Python extension interfaces to execute the code. Over the course of this project we have explored many of these solutions, all having their own limitations and approach to user-friendliness. To increase support and user-friendliness, we chose to separate processes on a service-level. gRPC supports many languages [grp] and is easy to use. This makes it easy to write different microservices in different languages, each having a language that excels at their task. Additionally, this service-level segregation improves PROVEE's scalability by supporting the offloading of computations to (external) servers. However, a disadvantage of using separate processes is that data is not directly accessible between them. Therefore one should carefully consider where data is stored to avoid redundant requests or duplicate data.

3.3 PROVEE's Analytic Capabilities

Both PROVEE's microservices pipeline and interservice communication are focused on achieving high flexibility and loose coupling. Although our architectural design facilitates the implemen-

tation of a wide range of functionalities, we focus on embedding exploration tasks as our first use case to study Progressive Analytics architectures. We implemented multiple features to showcase our system's extensibility and performance. First, an incremental PCA and a more memory reliant non-incremental UMAP have been implemented as **Model Producer** and **Model Consumer** microservices. Thus, even though PROVEE focuses on incremental algorithms, we also support memory intensive processing algorithms. Second, several features for analysis tasks have been implemented, which we will discuss in this section.

PROVEE is designed in collaboration with researchers in the field of NLP. They work with embeddings that represent texts, e.g. whole documents, sentences or individual words. Because these embeddings are opaque, there is an increasing interest in interpreting and analysing embeddings [BG19]. These interests vary in nature and can be categorized using different user roles. The *Searcher* user role analyses with a concrete, often isolated, question in mind. On the contrary, the *Explorer* aims to gain a comprehensive understanding of the whole embedding dataset. Lastly, *Observers* use their knowledge of VA to understand the resulting projection [MSA*19]. PROVEE aims to support a broad spectrum of users by employing features for these different perspectives.

Current visualization tools to analyze embeddings in NLP include Whatlies [WKT20], Tensorflow EP [STN*16] and Parallax [MWZ19]. Frequent analysis tasks are: local embedding inspection [BDS14, HLJ16], context differences [HLJ16], bias detection [BCZ*16, GSJZ18] and analogy search [MYZ13, GDM16]. Different analysis tasks can be performed with different tools, depending on the information of interest (see Table 3). However, as described in [FFNS19], lack of PVA causes tools to: (1) perform worse under stricter time and resource constraint, (2) take more time to detect wrong assumptions, e.g. projections with faulty parameters, and (3) result in a too high latency on large datasets for an effective work environment (see Section 4). PROVEE therefore incorporates progressive features for the following use cases:

Local Embedding Inspection: Crucial to embedding exploration is local neighborhood inspection [SGB*19] to gain insight into which points are close to, and therefore similar to, a specific point. We implemented the *kNN (k-Nearest Neighbor) microservice* to support this use case, as an example analytical microservice. It works parallel to the pipeline and starts as soon as an input file is selected. The kNN allows for searching in the high-dimensional embedding space. However, holding the full high-dimensional embedding space in memory is not desirable. We therefore implemented kNN search using the distributed GPU-enabled Faiss library [JDJ21]. Faiss allows to perform kNN searches with billions of data points by, e.g., using compression techniques and having the possibility to distribute the data over several physical machines. Thus, if the kNN service exceeds RAM limits we can distribute its index over several compute nodes. To measure how accurately a projection preserves the local neighborhood structure, the *kNN-Validator microservice* has been added. This analytical microservice compares the low-dimensional neighborhood of projected points to the actual neighborhood in high-dimensional space. It does so by constructing an undemanding low-dimensional kNN, using sklearn [PVG*11], and calculates how accurately these low-dimensional neighborhoods corresponds to the high-dimensional neighborhoods retrieved from the kNN microservice. This demonstrates how inter-dependencies between (analytical) microservices allow PROVEE to execute more complex tasks. By providing more insight in the progressive visualization, this feature is especially relevant to *Observers* [MSA*19].

Context differences: Context differences are analysed by comparing embeddings of different datasets. For example, researchers interested in language change can compare word embeddings trained on texts from different time periods. Local embedding inspection can then be carried out to understand how the meaning of an individual word has changed. Such an analysis could reveal, for example, how the word *broadcast*, initially used to describe scattering seeds, has now also acquired the meaning of transmitting signals [HLJ16]. PROVEE makes this possible by allowing the addition of datasets to the visualization, whilst using the Model Producer model on all added datasets. Furthermore, PROVEE makes it easy to differentiate between these datasets using highlighting and other color differences. This way we increase user-friendliness. PROVEE's awareness of the usage of multiple datasets can also be used to build additional features upon, as is demonstrated in Section 4.2 from the *Explorer's* perspective and discussed in Section 5.

Bias detection: Machine learning models can reflect or even enlarge societal biases. Such biases are sometimes also encoded directly in the embeddings themselves [BCZ*16, GSJZ18]. To support bias exploration, PROVEE allows users to define embedding axes. In the resulting projections, points are then plotted based on their distance to the embedding on the custom axis. For example, if the X-axis is *man* and the Y-axis is *woman*, points are projected as $p(X,Y) = (distance(p,Man), distance(p,Woman))$. Thus, one can expect *boy* to appear with a lower X value and a higher Y value than *girl*, exposing their relatedness. The current implementation uses the Euclidean distance using Faiss, via the kNN microservice, to support large requests. Support for cosine similarity, which is commonly used in NLP, could be added in the future.

Analogy search: To explore the structure of embedding spaces *embedding arithmetic* can be used [MYZ13]. For example, the quality of embeddings is sometimes evaluated by executing analogy queries in the form of `Paris - France + Germany` and testing whether the nearest neighbor of the resulting embedding is the embedding for `Berlin` [MYZ13]. Such analogies can cover a range of word relations [GDM16]. They have also been used to investigate biases in embedding spaces [BCZ*16] (but such results should be carefully interpreted, see [NvNvdG20]). We enable these analyses by allowing for simple embedding arithmetic, and performing a kNN request with the resulting custom embedding.

4 Evaluation

We conducted PROVEE's evaluation in two parts. First, we compare PROVEE with existing tools to demonstrate functional coverage and investigate performance and scalability. Second, we evaluate PROVEE in a real-world application using a natural language processing (NLP) case-study. In this way, we also show how the framework can be flexibly be adapted to desired uses.

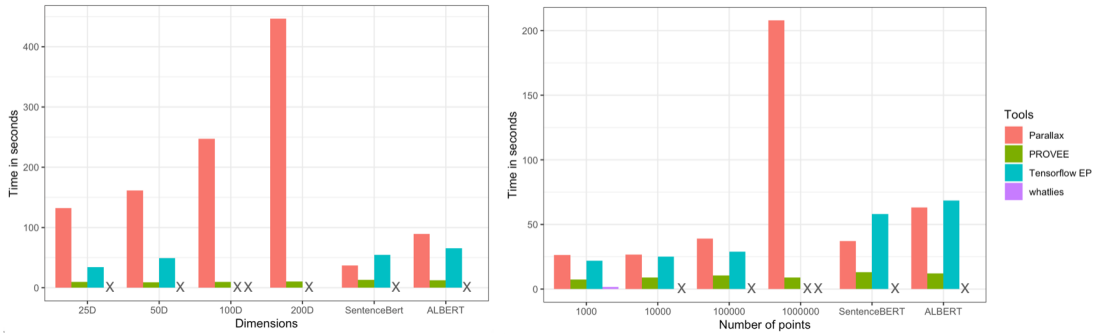


Figure 3: **Initialisation time comparison (sec)**, i.e. time it takes to show first results on the screen, of respectively Parallax, PROVEE, Tensorflow EP and Whatlies. **Left:** PROVEE outperforms all other tools in our experiments in which we vary the number of dimensions. With a dataset of 200-dimensional embeddings, PROVEE is roughly 43.8 times faster than Parallax. With 50D embeddings, PROVEE is 5.7 times faster than Tensorflow EP and 18.5 times faster than Parallax. **Right:** PROVEE also outperforms the other tools when varying the number of points in a dataset. On the dataset with 1 million points, PROVEE is almost 24 times faster than Parallax. On the dataset with 100.000 points, PROVEE outperforms Tensorflow EP and Parallax, by being almost 3 and 4 times faster respectively. In both figures SentenceBERT and ALBERT initialisation times are displayed, as these sets contain a typical number of dimensions for NLP embeddings and serve as a standard dataset in this figure.

4.1 Comparison with other tools

We compare PROVEE to three other tools that are similar in the way of embeddings visualization: Whatlies [WKT20], Parallax [MWZ19] and Tensorflow EP [STN*16]. The three tools and PROVEE are compared on initialisation time, full loading time, memory consumption and features. We use datasets with a varying number of embeddings and embedding dimensions, which are projected using PCA, as PROVEE currently does not hold any other incremental dimensionality reduction methods. This comparison reveals the scalability of not only PROVEE but also the other tools. Although the Tensorflow EP has a local and remote usage, we compare the local version. Important to note is that this tool downsamples PCA to visualize 100.000 points, a hard coded limit. It can be changed in package files, which did not work in our case. This indicates that measurements of the latter do not providing the full picture for datasets > 100.000 points and show the lower bound concerning time. We opted for the local version, because it shows similar workings to Parallax and PROVEE. Whatlies works differently from the other tools, providing interactive usage through Jupyter Notebook. As the tool can be executed through the various cells, loading dependencies and packages can be separated from the loading of the data and workings of the tool. In our comparison we separate these processes, measuring the initialisation and full loading time starting from the cells that load in the dataset and perform necessary computations that allow for visualization. We opted for this separation as one of the main differences between PROVEE and the other tools is the separation of loading the tool and the data. Both Tensorflow EP's local version and Parallax load the tool and the data at the same time, while PROVEE first displays the tool with no data and asks the user to load a dataset. This results in additional seconds for the full loading and initialisation time of the first two.

We create different embedding datasets to examine the performance of the tools along different numbers of dimensions and data points. First, we use 100-dimensional GloVe embeddings trained on Twitter data [PSM14]. To examine the performance of the tools in relation to the number of points, we randomly sample 1000, 10.000, 100.000 and 1.000.000 embeddings from the full dataset. Second, to experiment with embeddings with a larger number of dimensions, we use two state-of-the-art NLP models to generate embeddings of questions from the Quora dataset [IDC17]. We generate a dataset with 40.000 768D SentenceBERT embeddings [RG19] and a dataset with 30.000 1024D ALBERT embeddings [LCG*20].

Initialisation time: The first measure is initialisation time, i.e. the duration starting from loading the dataset until the moment the first visualization appears in a graph in seconds. Initialisation time is latency related to the display of the first results, after the user has requested results for the first time, while latency can also occur in later stages of visualization.

The left graph of Figure 3 shows the initialisation times along a varying number of dimensions (1.2M embeddings). As shown with the X, Whatlies is excluded from the measurements with more than 5000 points. In all cases, PROVEE has a lower initialisation time than Parallax and Tensorflow EP, despite loading the whole dataset instead of 100.000 points. One of PROVEE's aims is to stay under the 10 seconds latency, which is crucial to keep the user engaged in the task [FP16]. This figure shows that PROVEE's initialisation time stays close to 10 seconds, regardless of the size of the dataset.

The right graph of Figure 3 shows the initialisation times along a varying number of points. PROVEE has the lowest initialisation time overall. Whatlies is not able to show more than 5000 points and is therefore excluded from measurements with datasets above 1000 points. Whatlies has the fastest initialisation time for the first set of points (100). A possible explanation is that it runs through Jupiter Notebook, which takes less time then starting an application in a new screen, as the other tools do. Parallax is slightly slower than Tensorflow EP, but both are slower than PROVEE, which is in line with the aims of bounded latency.

Full loading time The second benchmark is full loading time, which is the time needed to fully load the whole projection in sec-

Tool	10K and 100D	GloVe Twitter 50D
PROVEE	8.1	196.3
Parallax	26.6	161.5
Tensorflow EP	27.9	51.6
Whatlies	-	-

Table 1: Examples full loading time (in seconds) by the four tools for two datasets: the sampled 100-dimensional 10K embeddings and GloVe Twitter dataset with 50-dimensional 1.2M points. PROVEE is outperformed by Parallax and Tensorflow Embedding Projector in set with more than a million points with 50 dimensions, but outperforms the other tools with 10.000 points with 100 dimensions.

Tool	10K and 100D	GloVe Twitter 50D
PROVEE	505.3	1528.7
Parallax	632.9	2914
Tensorflow EP	707	2352.3
Whatlies	-	-

Table 2: Examples of memory consumption by the four tools for the samples 10K embeddings with 100D and for the GloVe Twitter set of 50 dimensions (1.2M points). PROVEE outperforms Parallax and Tensorflow Embedding Projector both in the number of embeddings and in the number of dimensions.

onds. PROVEE's goal is to show early partial results, but not necessarily to speed up the process of loading the full dataset. Table 1 shows that PROVEE is not as fast as Parallax or the Tensorflow EP when more than a million points have to be displayed. Tensorflow EP appears to be the fastest for datasets with a larger number of points, but note that it only displays 100.000 points. This makes the comparison different. Parallax uses webGL for the scatterplot and HTML Canvas for the display of the labels. The authors mention themselves that HTML Canvas becomes slow when more than 10.000 points are used.

Memory consumption Lastly, the memory consumption in megabytes (MB) of the tools was measured during their performance. Parallax and Tensorflow EP both depend on visualization in the browser, while PROVEE and Whatlies do not. Whatlies is executed through Jupyter Notebook in Visual Studio Code. All tools are dependent on Python during their execution. To make a correct comparison, all necessary dependencies are taken into account when evaluating the RAM impact. For Parallax and Tensorflow EP this is Python (inside Visual Studio Code) and the browser (Chrome), for PROVEE Python only and for Whatlies Python and Jupyter notebook through Visual Studio Code. PROVEE consumes the least amount of memory during execution given different datasets of varying dimensions and number of points. Two situations are shown as an example in Table 2. It is worth mentioning that Whatlies outperforms PROVEE with a small number of points (1000) in terms of memory consumption.

Features Analytical tasks are discussed in Section 3.3. These tasks are enabled by several features. An example is the kNN search which supports the local embedding inspection. There are additional features aside from those that support analytical tasks. The additional features that could not be found in every evaluated tool were the loading of pre-built dimension reduction models and, the core of PROVEE, progressive visualization. To load pre-built models means to use already computed dimension reduction models, e.g. PCA or tSNE, on an embedding set. In addition, the ability to save and load pre-built models makes it possible to exchange models between users as well as improve time-efficiency, with no need to train the same model twice. As embedding sets can contain thousands of embeddings with many dimensions, a projector should be able to handle large datasets. PROVEE, Parallax and Tensorflow EP are able to load and display large datasets, although the Tensorflow EP does cut off the display at 100.000 points. Whatlies lacks this feature, but the tool was not intended to visualize large datasets, as it focusses on local embedding inspection. The second feature, progressive visualization, is one of the core aims of PROVEE. PROVEE is specifically designed to show meaningful intermediate results, which is a feature other tools lack.

One additional feature that has not yet been implemented in PROVEE is the web version of the tool. The Tensorflow EP can be accessed remotely, which enables the user to upload the dataset with corresponding metadata online. This provides an easily accessible version for users, which displays the embedding sets faster than the local Tensorflow EP version. Another additional feature that supports embedding visualization touches upon metadata visualization. PROVEE displays embedding labels only, but Tensorflow EP and Parallax can show more. Tensorflow EP is able to use images as metadata and Parallax can display PoS (Part of Speech) labels, such as nouns or adjectives, aside from the words.

Summary Combining the performance results and implemented features, PROVEE is a tool suited for basic analytical tasks performed on large amounts of data. PROVEE shows successful implementation of bounded latency, by outperforming other tools with respect to the initialisation time. Besides the lowest consumption of RAM by Whatlies, PROVEE outperforms the other two tools in terms of memory consumption. Noteworthy is the fact that the memory necessary for PROVEE does not increase linearly with the number of points or dimensions. As shown in Table 3, PROVEE is able to perform several analysis tasks in a progressive way, which yields fast meaningful partial results. Besides features supporting analytical tasks, PROVEE has other features that are beneficial in use cases where large information exchange is desired. PROVEE is build with flexibility that allows the incorporation of new features.

4.2 Case Study: Natural Language Processing Neural Networks

To display PROVEE's usefulness and extensibility we performed a case study on the training of a state-of-the-art NLP neural network model. Studies have shown that resulting embeddings can be sensitive to small changes (e.g., [AM18, WKM18]) and that even models trained with only different random seeds can vary substantially in their behavior [SYW*21]. Visualizing embeddings from neural networks trained with minor differences (e.g., training objectives or random seeds) can complement work that analyses resulting embeddings quantitatively (e.g. [SL19]).

The NLP researchers fine-tuned a set of *bert-large-uncased* models [DCLT19] on the Microsoft Research Paraphrase Corpus (MRPC) [DB05] using a learning rate of 5e-05, a batch size of 16, and 5 training epochs. The only difference between the mod-

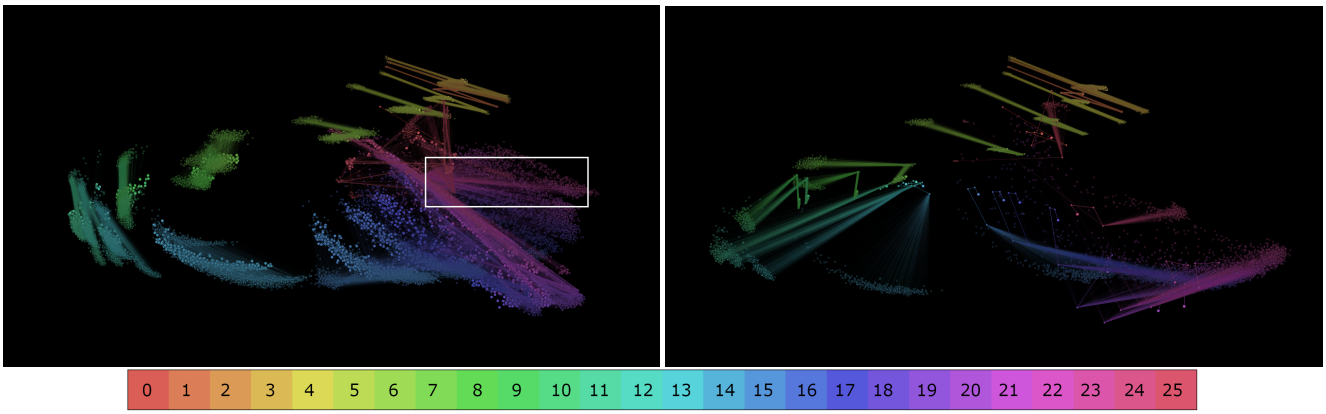


Figure 4: **Training trajectory** of a more successful (**left**) and less successful (**right**) BERT model [DCLT19] fine-tuned on the Microsoft Research Paraphrase Corpus (MRPC) [DB05]. Visualized using incremental PCA on layer embeddings at update steps: 152, 382, 536, 842 and 1148. The embedding datasets represent the 408 instances of the MRPC development set across the different layers, resulting in 53,040 points in each image. Gradient with numbering displays the layer index of each color. Highly concentrated layer point clouds in the projection could signify information loss. The white rectangle in the more successful model's figure highlights how the points of layers 19–22 from step 152 move leftwards and get more concentrated until step 382.

Analysis task	**Whatlies**	**Tensorflow EP**	**Parallax**	**PROVEE**
Local embedding inspection	X	X	X	X
Context differences	X		X	X
Bias detection	X	X	X	X
Analogy search	X		X	X
Supporting feature				
Loading pre-built models				X
Web access		X		
Display large datasets > 1M			X	X
Metadata beyond labels		X	X	
Progressive visualization				X

Table 3: Analysis task comparison of intended use of Whatlies, Tensorflow Embedding Projector, Parallax and PROVEE with overview of features further supporting users analytical tasks.

Step	**More Successful Model**	**Less Successful Model**
152	0.775	0.721
382	0.723	0.684
536	0.788	0.684
842	0.804	0.684
1148	0.875	0.684

Table 4: Accuracy of two fine-tuned BERT models at different update steps. The less successful model does not improve after iteration 382 and its overall performance is low.

els are the random seed. From this set, two models were selected for visualization: one where the performance was higher than average (more successful model) and one where the performance was lower than average (less successful model). PROVEE is then used to project the embeddings of each instance in the development set of MRPC at different layers. Embeddings were saved for different update steps. For the projection we use incremental PCA, fitting the PCA model on the embeddings of both models.

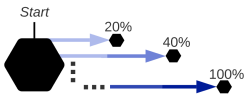
To visualize the change trajectories between steps more clearly, we expanded PROVEE by adding a new renderer to the Visualization microservice that draws lines between embeddings of the same instance across different steps. We added several options to PROVEE for multiple dataset visualizations; for instance highlighting a single dataset and adjusted color schemes to improve usability. These changes were rapidly implemented and received positive responses from the researchers involved with the case study.

The resulting visualizations are shown in Figure 4. The lower layers (0–∼5) behave similarly in both the more successful and less successful model. In contrast, the higher layers (∼6–25) differ remarkably: in the less successful model the higher layers converge sooner and become more concentrated. The early convergence of the less successful model can also be seen in its performance across different update steps (Table 4): The performance of the less successful model barely differs after step 382. With the successful model, layer point clouds never get as focused as the less successful model. However, the points of higher layers (∼14–25) are more concentrated at step 382, after which they dissipate again. This is seen for example in how the layers 19–22 at step 382 (marked by the white rectangle in Figure 4) move leftwards and get more clustered. In turn, the performance of the more successful model is lowest at step 382, see Table 4. Thus, differences in performance seem to be visible in the projections of the neural networks.

These interesting insights highlight the importance of embedding VA tools. With its seamless adaptability, PROVEE can be adapted to accommodate use cases in large variety of domains.

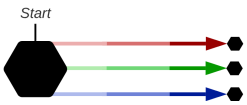
5 Discussion and Future Work

PROVEE builds on several important PVA(-related) works. For instance, Schulz et al.'s *Enhanced Visualization Process Model* [SASS16], the *Seven Design Goals* for PVA applications gathered by Stolper et al. [SPG14], or the definition of *Progressive Computations* (and their intertwined working) by Fekete [FP16]. These works center their discussions around the analysis of a single dataset, one applied ML algorithm, or one analysis scenario. However, with a flexible compute infrastructure, PVA is not limited to these choices. We identified the following extension possibilities.

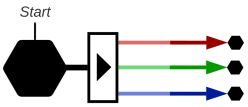


Partial Fits: Currently, the models are updated manually on user request, after an update period, or gradually as new data arrives, i.e., as *sequential* fixed-size batch updates or, in the extreme case, as *sequential* per-item updates. The order of these inputs matters in model training. To complement the options, we propose a percentage-based model which computes projections for, e.g., 25%, ..., 75%, 100% of the data sampled individually per subdivision. This randomness factor can lead to more qualitative and quantifiable insights on how projections evolve and provides a better understanding how the pipeline's algorithms are impacted from the quality of the supplied data. Due to the influence of input order, more research can and should be devoted into quality-aware data scheduling and management. PROVEE has shown already now to be a flexible backend for these experiments.

Comparing Algorithms: Comparing projection algorithms is a general challenge. In PROVEE, we can open multiple independent data processing streams per algorithm and parameter settings in parallel and compare their results. Seeing these individual projection results converge, i.e., build a consensus, can give more analytical certainty.



Multiple Datasets: Comparing multiple datasets is an important use case for many researchers interested in analyzing and improving machine learning models. In NLP, for example, researchers like to compare embeddings of models trained on different datasets (e.g. texts from different time periods [NGSP*21]), with different training objectives [YM21], or before and after applying "debiasing" techniques [BCZ*16]. As we have shown in our expert study in Section 4.2, we already support basic comparative analyses on various datasets (and characteristics). Future work could extend PROVEE to support more fine-grained comparative analyses.



Predictive Computation: If we could predict which data (aspect or sampling) contributes most to the patterns in the embedding space, we could display better results at an earlier time. The database domain (with predictive prefetching) [GTE*20b] and analytics domain (with interaction prefetching) [MHH19] have shown that these ideas can work in practice. We plan to experiment with these concepts in PROVEE.

Limitations: PROVEE solely focuses on architectural considerations and implements current visualization approaches (2D projections, lists, heatmaps) ML techniques (dimension reduction, clustering) and embedding analysis (kNN, embedding arithmetic, custom axes). Our goal is to demonstrate our design and advance PVA rather than to make information visualization or ML contributions.

In streaming systems or when using animated transitions in general, careful consideration has to be paid to the perceptual and cognitive limits, so that the analysts can still conduct effective pattern recognition and understanding [PMMG20, CDF14]. Systems like *Progressive Insights* [SPG14] or dynamic projection techniques like PCD-tSNE [VCT21] however, demonstrate the effective use of advanced (uncertainty) visualizations for these tasks. PROVEE has yet to explore dynamic projection methods in which users steer the dimension reduction process, which could deepen the user's understanding of the projection and help preserve the cognitive map.

Another current limitation in PROVEE is algorithmic support. While we have not yet hit a fundamental conceptual limit within PROVEE's architectural bounds, only a few dimension reduction algorithms have been implemented. We hope that in the future, more visualizations and incremental algorithms will be added by the visual analytics open-source community to further progress PVA.

PROVEE's strongly enforced separation-of-concerns also has disadvantages: (1) During the design it became clear that we have to pay minuscule attention to the question *"which microservice can and should store which information"*. Separating knowledge between microservices has the disadvantage that an interrelated collaboration between microservices is only possible through shared network memory. The fast compute, but slow update/propagation cycles, for this shared memory renders collaborative processes ultimately sequential in nature. (2) Relying on networked microservices creates an overhead in terms of RAM usage and latency, too. Although gRCP relieves PROVEE from many nuances of this process, e.g., deadline enforcement or retry management, data marshalling and unmarshalling are still considerable computation efforts, compared to single machine PVA systems.

6 Conclusion

With PROVEE, we present a scalable microservice-based architecture for Progressive Visual Analytics (PVA) to support system engineering and research. So far, the landscape has been fragmented along engineering-heavy solutions for distinct research questions. Our system enables a rapid and logically structured pathway towards complex PVA systems. In our systems paper, we describe prototypical implementations of several core and analytical functionalities. We showcase PROVEE's performance, analytical value and flexibility with a quantitative evaluation and a case study in NLP. In the future, we expect that our PVA architecture will lead to a more generic and structurized way to design PVA systems.

References

[Alb03] ALBERS S.: Online algorithms: a survey. *Math. Program. 97*, 1-2 (2003), 3–26. URL: https://doi.org/10.1007/s10107-003-0436-0, doi:10.1007/s10107-003-0436-0.3

[AM18] ANTONIAK M., MIMNO D.: Evaluating the stability of embedding-based word similarities. *Transactions of the Association for Computational Linguistics 6* (2018), 107–119. URL: https://aclanthology.org/Q18-1008, doi:10.1162/tacl_a_00008. 8

[AS13] ANGELINI M., SANTUCCI G.: Modeling incremental visualizations. In *4th International EuroVis Workshop on Visual Analytics, EuroVA@EuroVis 2013, Leipzig, Germany, June 17-18, 2013* (2013), Pohl M., Schumann H., (Eds.), Eurographics Association. URL: https://doi.org/10.2312/PE.EuroVAST.EuroVA13.013-017, doi:10.2312/PE.EuroVAST.EuroVA13.013-017. 2, 3

[ASSS18] ANGELINI M., SANTUCCI G., SCHUMANN H., SCHULZ H.: A review and characterization of progressive visual analytics. *Informatics 5*, 3 (2018), 31. URL: https://doi.org/10.3390/informatics5030031, doi:10.3390/informatics5030031. 2, 3

[BBC*11] BEHNEL S., BRADSHAW R., CITRO C., DALCIN L., SELJEBOTN D. S., SMITH K.: Cython: The best of both worlds. *Computing in Science Engineering 13*, 2 (2011), 31–39. doi:10.1109/MCSE.2010.118. 5

[BCS19] BOGGUST A., CARTER B., SATYANARAYAN A.: Embedding comparator: Visualizing differences in global structure and local neighborhoods via small multiples. *arXiv preprint arXiv:1912.04853* (2019). 3

[BCZ*16] BOLUKBASI T., CHANG K.-W., ZOU J., SALIGRAMA V., KALAI A.: Man is to computer programmer as woman is to homemaker? Debiasing word embeddings. In *NIPS'16* (2016), p. 4356–4364. 2, 6

[BDS14] BAMMAN D., DYER C., SMITH N. A.: Distributed representations of geographically situated language. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)* (Baltimore, Maryland, June 2014), Association for Computational Linguistics, pp. 828–834. URL: https://aclanthology.org/P14-2134, doi:10.3115/v1/P14-2134. 6

[BE98] BORODIN A., EL-YANIV R.: *Online computation and competitive analysis*. Cambridge University Press, 1998. 2, 3

[Ber11] BERTINI E.: Quality metrics in high-dimensional data visualization: An overview and systematization. *IEEE Trans. Vis. Comput. Graph. 17*, 12 (2011), 2203–2212. doi:10.1109/TVCG.2011.229. 5

[BG19] BELINKOV Y., GLASS J.: Analysis Methods in Neural Language Processing: A Survey. *Transactions of the Association for Computational Linguistics 7* (04 2019), 49–72. URL: https://doi.org/10.1162/tacl_a_00254, doi:10.1162/tacl_a_00254. 2, 6

[BLH21] BENGIO Y., LECUN Y., HINTON G.: Deep learning for AI. *Commun. ACM 64*, 7 (jun 2021), 58–65. URL: https://doi.org/10.1145/3448250, doi:10.1145/3448250. 1

[BTN*21] BEHRISCH M., TELEA A., NGUYEN D., VAN HERPT S., OWOLABI D., VAN DER BEN S.: Provee local projector, 2021. URL: https://git.science.uu.nl/vig/provee/provee-local-projector. 2

[Car02] CARLSSON M.: Monads for incremental computing. In *Proceedings of the Seventh ACM SIGPLAN International Conference on Functional Programming (ICFP '02), Pittsburgh, Pennsylvania, USA, October 4-6, 2002* (2002), Wand M., Jones S. L. P., (Eds.), ACM, pp. 26–35. URL: https://doi.org/10.1145/581478.581482, doi:10.1145/581478.581482. 3

[CDF14] CHEVALIER F., DRAGICEVIC P., FRANCONERI S.: The not-so-staggering effect of staggered animated transitions on visual tracking. *IEEE Trans. Vis. Comput. Graph. 20*, 12 (2014), 2241–2250. URL: https://doi.org/10.1109/TVCG.2014.2346424, doi:10.1109/TVCG.2014.2346424.

[CLK*14] CHOO J., LEE C., KIM H., LEE H., REDDY C. K., DRAKE B. L., PARK H.: PIVE: per-iteration visualization environment for supporting real-time interactions with computational methods. In *9th IEEE Conference on Visual Analytics Science and Technology, IEEE VAST 2014, Paris, France, October 25-31, 2014* (2014), Chen M., Ebert D. S., North C., (Eds.), IEEE Computer Society, pp. 241–242. URL: https://doi.org/10.1109/VAST.2014.7042510, doi:10.1109/VAST.2014.7042510. 3

[CXGH08] CHAN S., XIAO L., GERTH J., HANRAHAN P.: Maintaining interactivity while exploring massive time series. In *3rd IEEE Symposium on Visual Analytics Science and Technology, IEEE VAST 2008, Columbus, OH, USA, October 19-24, 2008* (2008), IEEE Computer Society, pp. 59–66. URL: https://doi.org/10.1109/VAST.2008.4677357, doi:10.1109/VAST.2008.4677357. 3

[DB05] DOLAN W. B., BROCKETT C.: Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)* (2005). URL: https://aclanthology.org/I05-5002. 8, 9

[DCLT19] DEVLIN J., CHANG M.-W., LEE K., TOUTANOVA K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (Minneapolis, Minnesota, June 2019), Association for Computational Linguistics, pp. 4171–4186. URL: https://aclanthology.org/N19-1423, doi:10.18653/v1/N19-1423. 2, 8, 9

[ED02] ELLIS G. P., DIX A. J.: Density control through random sampling: an architectural perspective. In *International Conference on Information Visualisation, IV 2002, London, England, UK, July 10-12, 2002* (2002), IEEE Computer Society, p. 82. URL: https://doi.org/10.1109/IV.2002.1028760, doi:10.1109/IV.2002.1028760. 5

[FFNS19] FEKETE J.-D., FISHER D., NANDI A., SEDLMAIR M.: Progressive Data Analysis and Visualization (Dagstuhl Seminar 18411). *Dagstuhl Reports 8*, 10 (2019), 1–40. URL: http://drops.dagstuhl.de/opus/volltexte/2019/10346, doi:10.4230/DagRep.8.10.1. 2, 6

[Fis11] FISHER D.: Incremental, approximate database queries and uncertainty for exploratory visualization. In *IEEE Symposium on Large Data Analysis and Visualization, LDAV 2011, Providence, Rhode Island, USA, 23-24 October, 2011* (2011), Rogers D. H., Silva C. T., (Eds.), IEEE Computer Society, pp. 73–80. URL: https://doi.org/10.1109/LDAV.2011.6092320, doi:10.1109/LDAV.2011.6092320. 2

[FP16] FEKETE J.-D., PRIMET R.: Progressive analytics: A computation paradigm for exploratory data analysis. *arXiv preprint arXiv:1607.05162* (2016). 2, 3, 7

[FPDmcs12] FISHER D., POPOV I. O., DRUCKER S. M., M. C. SCHRAEFEL: Trust me, I'm partially right: incremental visualization lets analysts explore large datasets faster. In *CHI Conference on Human Factors in Computing Systems, CHI '12* (2012), Konstan J. A., Chi E. H., Höök K., (Eds.), ACM, pp. 1673–1682. URL: https://doi.org/10.1145/2207676.2208294, doi:10.1145/2207676.2208294. 2

[GDM16] GLADKOVA A., DROZD A., MATSUOKA S.: Analogy-based detection of morphological and semantic relations with word embeddings: what works and what doesn't. In *Proceedings of the NAACL Student Research Workshop* (San Diego, California, June 2016), Association for Computational Linguistics, pp. 8–15. URL: https://aclanthology.org/N16-2002, doi:10.18653/v1/N16-2002. 6

[GJM*20] GEIRHOS R., JACOBSEN J.-H., MICHAELIS C., ZEMEL R., BRENDEL W., BETHGE M., WICHMANN F. A.: Shortcut learning in deep neural networks. *Nature Machine Intelligence 2*, 11 (2020), 665–673. 2

[GMR*18] GUIDOTTI R., MONREALE A., RUGGIERI S., TURINI F., GIANNOTTI F., PEDRESCHI D.: A survey of methods for explaining black box models. *ACM Comput. Surv. 51*, 5 (aug 2018). URL: https://doi.org/10.1145/3236009, doi:10.1145/3236009. 2

[grp] Supported languages. URL: https://grpc.io/docs/languages/. 5

[GSJZ18] GARG N., SCHIEBINGER L., JURAFSKY D., ZOU J.: Word embeddings quantify 100 years of gender and ethnic stereotypes. *Proceedings of the National Academy of Sciences 115*, 16 (2018), E3635–E3644. 6

[GTE*20a] GOGOLOU A., TSANDILAS T., ECHIHABI K., BEZERIANOS A., PALPANAS T.: Data series progressive similarity search with probabilistic quality guarantees. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data* (New York, NY, USA, 2020), SIGMOD '20, Association for Computing Machinery, p. 1857–1873. URL: https://doi.org/10.1145/3318464.3389751, doi:10.1145/3318464.3389751. 2

[GTE*20b] GOGOLOU A., TSANDILAS T., ECHIHABI K., BEZERIANOS A., PALPANAS T.: Data series progressive similarity search with probabilistic quality guarantees. In *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020* (2020), Maier D., Pottinger R., Doan A., Tan W., Alawini A., Ngo H. Q., (Eds.), ACM, pp. 1857–1873. URL: https://doi.org/10.1145/3318464.3389751, doi:10.1145/3318464.3389751.

[HHW97] HELLERSTEIN J. M., HAAS P. J., WANG H. J.: Online aggregation. In *SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data, May 13-15, 1997, Tucson, Arizona, USA* (1997), Peckham J., (Ed.), ACM Press, pp. 171–182. URL: https://doi.org/10.1145/253260.253291, doi:10.1145/253260.253291. 3

[HLJ16] HAMILTON W. L., LESKOVEC J., JURAFSKY D.: Diachronic word embeddings reveal statistical laws of semantic change. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (Berlin, Germany, Aug. 2016), Association for Computational Linguistics, pp. 1489–1501. URL: https://aclanthology.org/P16-1141, doi:10.18653/v1/P16-1141. 6

[IDC17] IYER S., DANDEKAR N., CSERNAI K.: First quora dataset release: Question pairs - data @ quora, 2017. URL: https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs. 7

[JDJ21] JOHNSON J., DOUZE M., JÉGOU H.: Billion-scale similarity search with gpus. *IEEE Transactions on Big Data 7*, 3 (2021), 535–547. doi:10.1109/TBDATA.2019.2921572. 6

[KCL*17] KIM H., CHOO J., LEE C., LEE H., REDDY C. K., PARK H.: PIVE: per-iteration visualization environment for real-time interactions with dimension reduction and clustering. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA* (2017), Singh S. P., Markovitch S., (Eds.), AAAI Press, pp. 1001–1009. URL: http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14381. 3

[KJS20] KO H.-K., JO J., SEO J.: Progressive Uniform Manifold Approximation and Projection. In *EuroVis 2020 - Short Papers* (2020), Kerren A., Garth C., Marai G. E., (Eds.), The Eurographics Association. doi:10.2312/evs.20201061. 3, 5

[KMS*08] KEIM D. A., MANSMANN F., SCHNEIDEWIND J., THOMAS J., ZIEGLER H.: Visual analytics: Scope and challenges. In *Visual data mining*. Springer, 2008, pp. 76–90. 2

[LCG*20] LAN Z., CHEN M., GOODMAN S., GIMPEL K., SHARMA P., SORICUT R.: Albert: A lite BERT for self-supervised learning of language representations. In *ICLR 2020* (2020). 7

[LJH13] LIU Z., JIANG B., HEER J.: *imMens*: Real-time visual querying of big data. *Comput. Graph. Forum 32*, 3 (2013), 421–430. URL: https://doi.org/10.1111/cgf.12129, doi:10.1111/cgf.12129. 3

[LJLH19] LIU Y., JUN E., LI Q., HEER J.: Latent space cartography: Visual analysis of vector space embeddings. *Computer Graphics Forum 38*, 3 (2019), 67–78. 3

[Lu20] LU Z.: A case study about different network architectures in federated machine learning, 2020. URL: http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-425193. 5

[MHH19] MORITZ D., HOWE B., HEER J.: Falcon: Balancing interactive latency and resolution sensitivity for scalable linked visualizations. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, CHI 2019, Glasgow, Scotland, UK, May 04-09, 2019* (2019), Brewster S. A., Fitzpatrick G., Cox A. L., Kostakos V., (Eds.), ACM, p. 694. URL: https://doi.org/10.1145/3290605.3300924, doi:10.1145/3290605.3300924.

[Mil68] MILLER R. B.: Response time in man-computer conversational transactions. In *American Federation of Information Processing Societies: Proceedings of the AFIPS '68 Fall Joint Computer Conference, December 9-11, 1968, San Francisco, California, USA - Part I* (1968), vol. 33 of *AFIPS Conference Proceedings*, AFIPS / ACM / Thomson Book Company, Washington D.C., pp. 267–277. URL: https://doi.org/10.1145/1476589.1476628, doi:10.1145/1476589.1476628. 3

[MMS*21] MEHRABI N., MORSTATTER F., SAXENA N., LERMAN K., GALSTYAN A.: A survey on bias and fairness in machine learning. *ACM Comput. Surv. 54*, 6 (jul 2021). URL: https://doi.org/10.1145/3457607, doi:10.1145/3457607. 2

[MPG*14] MÜHLBACHER T., PIRINGER H., GRATZL S., SEDLMAIR M., STREIT M.: Opening the black box: Strategies for increased user involvement in existing algorithm implementations. *IEEE Trans. Vis. Comput. Graph. 20*, 12 (2014), 1643–1652. URL: https://doi.org/10.1109/TVCG.2014.2346578, doi:10.1109/TVCG.2014.2346578. 2, 3

[MSA*19] MICALLEF L., SCHULZ H., ANGELINI M., AUPETIT M., CHANG R., KOHLHAMMER J., PERER A., SANTUCCI G.: The human user in progressive visual analytics. In *21st Eurographics Conference on Visualization, EuroVis 2019 - Short Papers, Porto, Portugal, June 3-7, 2019* (2019), Johansson J., Sadlo F., Marai G. E., (Eds.), Eurographics Association, pp. 19–23. URL: https://doi.org/10.2312/evs.20191164, doi:10.2312/evs.20191164. 2, 6

[Mun08] MUNZNER T.: Process and pitfalls in writing information visualization research papers. In *Information Visualization - Human-Centered Issues and Perspectives*, Kerren A., Stasko J. T., Fekete J., North C., (Eds.), vol. 4950 of *Lecture Notes in Computer Science*. Springer, 2008, pp. 134–153. URL: https://doi.org/10.1007/978-3-540-70956-5_6, doi:10.1007/978-3-540-70956-5_6. 2

[Mut05] MUTHUKRISHNAN S.: Data streams: Algorithms and applications. *Found. Trends Theor. Comput. Sci. 1*, 2 (2005). URL: https://doi.org/10.1561/0400000002, doi:10.1561/0400000002. 2

[MWZ19] MOLINO P., WANG Y., ZHANG J.: Parallax: Visualizing and understanding the semantics of embedding spaces via algebraic formulae. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations* (Florence, Italy, July 2019), Association for Computational Linguistics, pp. 165–180. URL: https://aclanthology.org/P19-3028, doi:10.18653/v1/P19-3028. 3, 6, 7

[MYZ13] MIKOLOV T., YIH W.-T., ZWEIG G.: Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (Atlanta, Georgia, June 2013), Association for Computational Linguistics, pp. 746–751. URL: https://aclanthology.org/N13-1090. 6

[NB83] NAISBITT J., BISESI M.: Megatrends: Ten new directions transforming our lives. *Sloan Management Review (pre-1986) 24*, 4 (1983), 69. 1

[NG19] NAGPAL A., GABRANI G.: Python for data analytics, scientific and technical applications. In *2019 Amity International Conference on Artificial Intelligence (AICAI)* (2019), pp. 140–145. doi:10.1109/AICAI.2019.8701341. 5

[NGSP*21] NEWMAN-GRIFFIS D., SIVARAMAN V., PERER A., FOSLER-LUSSIER E., HOCHHEISER H.: TextEssence: A tool for interactive analysis of semantic shifts between corpora. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Demonstrations* (Online, June 2021), Association for Computational Linguistics, pp. 106–115. URL: https://aclanthology.org/2021.naacl-demos.13, doi:10.18653/v1/2021.naacl-demos.13.

[NvNvdG20] NISSIM M., VAN NOORD R., VAN DER GOOT R.: Fair is better than sensational: Man is to doctor as woman is to doctor. *Computational Linguistics 46*, 2 (June 2020), 487–497. URL: https://aclanthology.org/2020.cl-2.7, doi:10.1162/coli_a_00379. 6

[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 Trans. Vis. Comput. Graph. 23*, 7 (2017), 1739–1752. URL: https://doi.org/10.1109/TVCG.2016.2570755, doi:10.1109/TVCG.2016.2570755. 2, 3

[PMMG20] PEREIRA T., MOREIRA J., MENDES D., GONÇALVES D.: Evaluating animated transitions between contiguous visualizations for streaming big data. In *31st IEEE Visualization Conference, IEEE VIS 2020 - Short Papers, Virtual Event, USA, October 25-30, 2020* (2020), IEEE, pp. 161–165. URL: https://doi.org/10.1109/VIS47514.2020.00039, doi:10.1109/VIS47514.2020.00039.

[PSM14] PENNINGTON J., SOCHER R., MANNING C.: GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (Doha, Qatar, Oct. 2014), Association for Computational Linguistics, pp. 1532–1543. URL: https://aclanthology.org/D14-1162, doi:10.3115/v1/D14-1162. 7

[PVG*11] PEDREGOSA F., VAROQUAUX G., GRAMFORT A., MICHEL V., THIRION B., GRISEL O., BLONDEL M., PRETTENHOFER P., WEISS R., DUBOURG V., VANDERPLAS J., PASSOS A., COURNAPEAU D., BRUCHER M., PERROT M., DUCHESNAY E.: Scikit-learn: Machine learning in python. *J. Mach. Learn. Res. 12* (2011), 2825–2830. URL: http://dl.acm.org/citation.cfm?id=2078195. 3, 6

[RG19] REIMERS N., GUREVYCH I.: Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (Hong Kong, China, Nov. 2019), Association for Computational Linguistics, pp. 3982–3992. URL: https://aclanthology.org/D19-1410, doi:10.18653/v1/D19-1410. 7

[RGYSD17] RAGHU M., GILMER J., YOSINSKI J., SOHL-DICKSTEIN J.: SVCCA: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Red Hook, NY, USA, 2017), NIPS'17, Curran Associates Inc., p. 6078–6087. 2

[RLLY08] ROSS D. A., LIM J., LIN R.-S., YANG M.-H.: Incremental learning for robust visual tracking. *International journal of computer vision 77*, 1 (2008), 125–141. 4

[Saa03] SAAD Y.: *Iterative methods for sparse linear systems*. SIAM, 2003. URL: https://doi.org/10.1137/1.9780898718003, doi:10.1137/1.9780898718003. 2, 3

[SASS16] SCHULZ H., ANGELINI M., SANTUCCI G., SCHUMANN H.: An enhanced visualization process model for incremental visualization. *IEEE Trans. Vis. Comput. Graph. 22*, 7 (2016), 1830–1842. URL: https://doi.org/10.1109/TVCG.2015.2462356, doi:10.1109/TVCG.2015.2462356. 2, 3

[SGB*19] STROBELT H., GEHRMANN S., BEHRISCH M., PERER A., PFISTER H., RUSH A. M.: Seq2seq-vis: A visual debugging tool for sequence-to-sequence models. *IEEE Trans. Vis. Comput. Graph. 25*, 1 (2019), 353–363. URL: https://doi.org/10.1109/TVCG.2018.2865044, doi:10.1109/TVCG.2018.2865044. 6

[Shn84] SHNEIDERMAN B.: Response time and display rate in human performance with computers. *ACM Comput. Surv. 16*, 3 (1984), 265–285. URL: https://doi.org/10.1145/2514.2517, doi:10.1145/2514.2517. 3

[SL19] SAPHRA N., LOPEZ A.: Understanding learning dynamics of language models with SVCCA. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (Minneapolis, Minnesota, June 2019), Association for Computational Linguistics, pp. 3257–3267. URL: https://aclanthology.org/N19-1329, doi:10.18653/v1/N19-1329. 8

[Smi20] SMITH N. A.: Contextual word representations: Putting words into computers. *Commun. ACM 63*, 6 (may 2020), 66–74. URL: https://doi.org/10.1145/3347145, doi:10.1145/3347145. 2

[SPG14] STOLPER C. D., PERER A., GOTZ D.: Progressive visual analytics: User-driven visual exploration of in-progress analytics. *IEEE Transactions on Visualization and Computer Graphics 20*, 12 (2014), 1653–1662. 2, 3

[SSS*14] SACHA D., STOFFEL A., STOFFEL F., KWON B. C., ELLIS G. P., KEIM D. A.: Knowledge generation model for visual analytics. *IEEE Trans. Vis. Comput. Graph. 20*, 12 (2014), 1604–1613. URL: https://doi.org/10.1109/TVCG.2014.2346481, doi:10.1109/TVCG.2014.2346481. 2

[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 2016 Workshop on Interpretable Machine Learning in Complex Systems* (2016). 3, 4, 6, 7

[SYW*21] SELLAM T., YADLOWSKY S., WEI J., SAPHRA N., D'AMOUR A., LINZEN T., BASTINGS J., TURC I., EISENSTEIN J., DAS D., TENNEY I., PAVLICK E.: The multiberts: Bert reproductions for robustness analysis, 2021. arXiv:2106.16163. 8

[THSW15] TANGWONGSAN K., HIRZEL M., SCHNEIDER S., WU K.: General incremental sliding-window aggregation. *Proc. VLDB Endow. 8*, 7 (2015), 702–713. URL: http://www.vldb.org/pvldb/vol8/p702-tangwongsan.pdf, doi:10.14778/2752939.2752940. 3

[TKBH17] TURKAY C., KAYA E., BALCISOY S., HAUSER H.: Designing progressive and interactive analytics processes for high-dimensional data analysis. *IEEE Trans. Vis. Comput. Graph. 23*, 1 (2017), 131–140. URL: https://doi.org/10.1109/TVCG.2016.2598470, doi:10.1109/TVCG.2016.2598470. 2, 3

[VCT21] VERNIER E. F., COMBA J. L. D., TELEA A. C.: Guided Stable Dynamic Projections. *Computer Graphics Forum* (2021). doi:10.1111/cgf.14291.

[VdMH08] VAN DER MAATEN L., HINTON G.: Visualizing data using t-SNE. *Journal of Machine Learning Research 9*, 86 (2008), 2579–2605. URL: http://jmlr.org/papers/v9/vandermaaten08a.html. 4

[Wik21] WIKIPEDIA CONTRIBUTORS: Remote procedure call — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Remote_procedure_call&oldid=1052084053, 2021. [Online; accessed 28-October-2021]. 5

[WKM18] WENDLANDT L., KUMMERFELD J. K., MIHALCEA R.: Factors influencing the surprising instability of word embeddings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)* (New Orleans, Louisiana, June 2018), Association for Computational Linguistics, pp. 2092–2102. URL: https://aclanthology.org/N18-1190, doi:10.18653/v1/N18-1190. 8

[WKT20] WARMERDAM V., KOBER T., TATMAN R.: Going beyond T-SNE: Exposing whatlies in text embeddings. In *Proceedings of Second Workshop for NLP Open Source Software (NLP-OSS)* (Online, Nov. 2020), Association for Computational Linguistics, pp. 52–60. URL: https://aclanthology.org/2020.nlposs-1.8, doi:10.18653/v1/2020.nlposs-1.8. 3, 6, 7

[WM04] WILLIAMS M., MUNZNER T.: Steerable, progressive multidimensional scaling. In *10th IEEE Symposium on Information Visualization (InfoVis 2004), 10-12 October 2004, Austin, TX, USA* (2004), Ward M. O., Munzner T., (Eds.), IEEE Computer Society, pp. 57–64. URL: https://doi.org/10.1109/INFVIS.2004.60, doi:10.1109/INFVIS.2004.60. 2, 3

[WZZ93] WANG X., ZHAO H., ZHU J.: GRPC: a communication cooperation mechanism in distributed systems. *SIGOPS Oper. Syst. Rev. 27*, 3 (July 1993), 75–86. URL: https://doi.org/10.1145/155870.155881, doi:10.1145/155870.155881. 5

[YM21] YAUNEY G., MIMNO D.: Comparing text representations: A theory-driven approach. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing* (Online and Punta Cana, Dominican Republic, Nov. 2021), Association for Computational Linguistics, pp. 5527–5539. URL: https://aclanthology.org/2021.emnlp-main.449.

[ZYK06] ZHAO H., YUEN P. C., KWOK J. T.: A novel incremental principal component analysis and its application for face recognition. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 36*, 4 (2006), 873–886. 4