Multimodal Retrieval of Genomics Data Visualizations

Huyen N. Nguyen, Sehi L'Yi, Thomas C. Smits, Shanghua Gao, Marinka Zitnik, Nils Gehlenborg

Abstract—To address the challenges of efficiently retrieving information from the vast landscape of genomics data visualizations, we introduce a database system designed for retrieving interactive genomics visualizations. The system supports multimodal retrieval to cater to diverse user needs, offering flexibility in search methods. Through a user interface, users can choose their preferred query approach: example images, natural language queries, or grammar-based queries. For each visualization, we construct a set of multimodal representations from the three corresponding modalities: a declarative specification using Gosling visualization grammar to define the structural framework, a pixel-based rendering generated from that specification, and a text description that details the visualization in natural language. To leverage both specialized knowledge from the grammar and general knowledge from a multimodal biomedical foundation model and a large language model, our approach incorporates three types of embedding methods: context-free grammar embeddings, multimodal embeddings, and textual embeddings. We designed the context-free grammar embeddings specifically tailored for grammar-based genomics visualizations, addressing previously underexplored aspects such as genomic tracks, views, and interactivity. The multimodal embeddings are derived from a state-of-the-art biomedical vision-language foundation model, while the textual embeddings are generated by our fine-tuned specification-to-text large language model; both capture generalized insights from large-scale training data. We experimented with different embedding methods across different variations of each modality to identify the strategies that maximize the top-k retrieval accuracy. The current collection comprises 3,200 visualization examples across about 50 categories, from single-view to coordinated multi-view visualizations, and covering a wide range of applications, such as single-cell epigenomics and structural variation analysis.

Index Terms—Multimodal retrieval, multimodal representation, genomic data visualization

1 Introduction

The exponential growth of genomics data has driven a rapid increase in the development of interactive visualizations, establishing new frontiers in both biology and data visualization. This expansion, while essential, has presented both new opportunities and significant challenges. The sheer number and variety of visualizations make it difficult for researchers and practitioners to efficiently search for relevant examples. Although previous work has tackled the problem of retrieving general visualizations [3, 7, 28, 42, 44], there has been limited focus on developing specialized search engines tailored to the unique demands of genomics visualization research, considering the specific data formats, grammar constructs, track and view types, among other interactive elements. The problem of finding relevant example visualizations becomes more pronounced when it comes to visualization authoring [38]. Authoring genomics data visualizations is a challenging task as it often requires researchers to spend significant time searching for relevant examples to use as templates. The development of a search engine, capable of understanding the specific characteristics of genomics representations, is essential for facilitating both the discovery and authoring of genomics data visualizations.

To address this challenge of navigating the vast landscape of genomics data visualizations, we introduce a database system that supports multimodal retrieval tailored to the genomics domain. The goal of the system is to enable users to efficiently find visualizations that meet their needs through images, text descriptions, or partial specifications. By providing access to reusable examples, we seek to empower researchers to focus on data analysis and interpretation rather than the mechanics of visualization construction. The visualizations from retrieval results can be used as *scaffolds* in the authoring process: template structures that can be filled in or modified with users' own data and visual design.

For each visualization, we construct a set of multimodal representations from the three corresponding modalities: a declarative specification using Gosling visualization grammar [22] to define the structural framework, a pixel-based rendering generated from that specification, and a text description that details the visualization in natural language. Having each modality provides a narrow slice of the understanding of the underlying concept, the multimodal representation approach attempts to provide a comprehensive understanding of the visualization. Subsequently, we developed a large document collection containing triplets of these modalities as the basis for the database system, ranging from common visualization types to real-world applications, such as breast cancer variants of human genomes and single-cell epigenomics. Specifically, we introduce a new approach to generate rich text descriptions for genomics data visualization by combining structural knowledge from automatic text description generation tool AltGosling [35] with information extracted from the specification and its classified properties and static image using a large language models (LLM).

Building upon this rich multimodal dataset, we develop and experiment with different methods to transform raw modalities into semantically meaningful vector representations: context-free grammar (CFG) embeddings, multimodal embeddings, and textual embeddings. We designed the CFG embeddings specifically tailored for grammar-based genomics visualizations, addressing previously underexplored aspects such as genomic tracks, views, and interactivity. The multimodal embeddings are derived from a state-of-the-art biomedical vision-language foundation model, while the textual embeddings are generated by our fine-tuned specification-to-text LLM; both capture generalized insights from large-scale training data. Though it does not require training, the CFG approach demonstrates comparable results with other methods.

In particular, to enable effective retrieval between natural language text and formal specifications, we leverage pretrained textual embedding models that map textual inputs into a shared semantic space. These models, trained on large-scale natural language corpora, offer strong generalization across varied text types. However, specifications introduce a distinct domain characterized by structured, rule-based representations not typically encountered in natural language. To address this domain gap, we fine-tune the embedding models on a curated text-specification dataset. This adaptation retains the models' broad semantic capabilities while aligning them more closely with the structure of specifications, enabling accurate and robust cross-modal retrieval.

In summary, the contributions of this paper are threefold:

- First, we propose a database system for multimodal querying of

• *The authors are with Harvard Medical School. E-mail: {huyen_nguyen | sehi_lyi | tsmits | shanghua_gao | marinka | nils} @hms.harvard.edu*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

interactive genomics visualizations, allowing users to search via example images, natural language, or grammar-based queries. Our collection includes 3,200 interactive genomic visualizations across around 50 categories.

- Second, we introduce multimodal representations for genomics visualizations by combining context-free grammar embeddings, multimodal embeddings, and fine-tuned textual embeddings across three modalities: text descriptions, visualization images, and declarative Gosling grammar specifications
- Finally, we evaluate and gather insights from experiments with different retrieval and embedding strategies. One notable observation is that the CFG approach applied to specifications shows promising retrieval results and is not computational heavy.

2 Motivation: Authoring Genomics Data Visualization

In genomics, visualization authoring is regarded as a challenging task considering the complexity of visualizations and interactions that are frequently used (e.g., coordinated many-view visualizations) [18, 21, 23, 38]. Our interviews with domain experts showed that they use various approaches for authoring genomics data visualizations [38]. One common approach is to use genome browsers, such as IGV.js [31] and JBrowse 2 [4]. While these tools enable the easy creation of visualizations, they offer limited customization of visualizations to meet users' different needs. Another approach is using a visualization grammar, such as Gosling [21] and Gos [24]. It provides flexibility in authoring custom visualizations while still enabling the ease-of-use for domain experts with computational skills, such as computational biologists and software engineers. More recently, the grammar has been deployed in Blace [20] by combining familiar graphical user interfaces and an LLM to further lower barriers for customizing genomics data visualizations.

Since visualization authoring is a challenging task for many domain experts, we found from our user interviews [23, 38] that domain experts frequently utilize existing visualization examples (e.g., visual and code examples) for inspiration and make their visualization process more efficient by reusing examples. For example, domain experts frequently search for research papers that use the same data types (e.g., using Google Scholar with a data description as a search keyword) and try to find data visualization figures to get an idea on what kinds of visualizations they need to create. Another example is using Q&A forums (e.g., Stack Overflow) to find reusable code examples. However, the workflow to retrieve visualization examples is highly time-consuming since it involves many trials-and-errors to find the right visualization that they are finding for. The main motivation of our work is to make this retrieval process efficient so that domain experts can more easily retrieve visualization examples that they can use in their visualization authoring workflow.

3 Related Work

3.1 Representations of Visualization

Visualization is the visual representation of data, and in this section, we explore the various forms in which visualization *itself* can be represented. These representations range from intuitive formats, such as images (pixel-based displays) and text descriptions, including alt-text [35], to more structured formats, such as vector graphics-based structures (SVG) [13] and XML-based formats [34]. Among these representation approaches, declarative grammar specifications such as Vega-Lite [32] have emerged as a particularly powerful framework for visualization design, offering a structured yet expressive paradigm that forms the central focus of our work.

Declarative grammar offers a formalized approach to describing visualizations, allowing for precise specifications of visual elements and interactions. Starting with Wilkinson's Grammar of Graphics (GoG) [40], recent work from the visualization community has shown increasing attention to visualization grammars in both theory and practice. This includes the development of grammar specification toolkits for general charts such as Vega-Lite [33] and Visualization Object Model [15], which presented a generative process of constructing a visualization through a set of concatenated high-level production rules. Beyond general-purpose tools, domain-specific toolkits have been developed, including Gosling [22] for genomics and GoTree [12] for phylogenetic trees.

These grammars function both as specification languages and structured frameworks for creating meaningful visualization vector embeddings. Recent chart representation advances have leveraged visualization grammars, particularly context-free grammars (CFG) through Grammar Variational Autoencoder (GVAE) applications [9]. One of the early adoptions of GVAE in visualization research is Chartseer [47], which makes use of Vega-lite [32] JSON specs to create one-hot encodings, where it captures the presence or absence of each grammar rule. A more detailed explanation of one-hot encoding will be provided in Section 6.1.3. GoTreeScape by Li and Yuan [12] applied a similar approach to tree visualization with GoTree specifications [11]. GraphScape [8] quantified transition costs among different chart specifications and visualized embeddings as nodes and the transitions as edges. ComputableViz [41] supported operations on multiple visualization specifications, enabling the generation of visualization embeddings and accessibility of visualizations.

3.2 Visualization Retrieval

Once information is represented, the next task is to retrieve it effectively. Hoque and Agrawala [7] introduced a straightforward retrieval system enabling users to search for D3 visualizations with incomplete Vega-Lite JSON grammar, leveraging the structure and explicit format of specifications. VISAtlas by Ye et al. [44] utilized embeddings in the visual interface to facilitate navigation. Recent work by Xiao et al. [42] enabled multimodal inputs to help capture user intent. Erato [37] and later, Chart2Vec [3] presented approaches to retrieve visualizations in data stories supporting multi-view visualizations, while introducing declarative visualization grammar for universal chart embeddings.

While the main focus of previous work is on finding visualizations with similar visual encodings, it is also important to find documents that exhibit similar data types. VizCommender [28] took this consideration into account for searching Tableau visualizations, and most recently, VAID by Ying et al. [45] explored this direction with a grammar-based approach. Notably, KnowledgePearls [36] enables searching analysis states independent of the underlying visualization framework.

Retrieval becomes truly beneficial when we have a substantial collection. Previous work on visualization databases and corpora [1, 2] inspired us to build our own system in the genomics domain. In particular, Olio by Setlur et al. [34] facilitates semantic search on data repositories, emphasizing the 'pre-authored' aspect of visualizations, which is aligned with our driving motivation in authoring for this work.

4 Methodology

4.1 Design Considerations

The primary goal of the proposed database system is to explore effective retrieval of genomics data visualizations by supporting diverse user needs and search strategies. Its design was shaped by insights derived from authoring genomics data visualizations, as outlined in Section 2. Our workflow draws inspiration from prior work on encoding grammar-based visualizations [12, 47] and visualization retrieval [3, 7, 28, 42, 45], each of which employs a different form of query input. Below, we present the key design considerations that guided the development of the proposed system.

D1. Provide multimodal retrieval support to accommodate diverse user needs The system needs to support multiple retrieval modalities to serve users with varying expertise and preferences. Users should be able to search the visualization collection through three distinct methods: (1) example images, where users provide visual references; (2) natural language queries, allowing intuitive text-based descriptions; and (3) grammar-based queries using Gosling specifications. This flexibility aims to engage users with searching regardless of their expertise or communication preference.

D2. Construct multimodal representations that provide comprehensive information Appropriate representations provide the basis for effective retrieval. For each visualization in the collection, it is critical

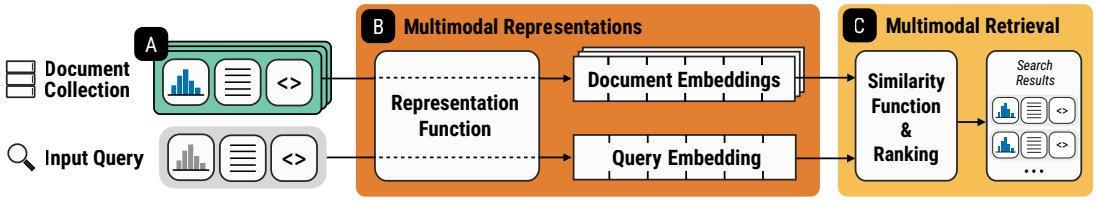

Fig. 1: Overview of the database system. Panel (A) depicts the document collection, where each document is represented by a triplet of three modalities: image, text, and specification, as discussed in Section 5. An input query can be based on one of these modalities. Panel (B), detailed in Section 6, present the multimodal representation, utilizing embedding methods as transformation functions to convert raw data modalities into corresponding embeddings. Panel (C) is covered in Section 7 and demonstrates computing similarities between query and document embeddings to rank documents by relevance, with results presented as a list of ranked triplets.

to generate a comprehensive multimodal representation comprising three components: (1) a specification component using declarative Gosling grammar; (2) an image component representing the pixel-based rendering; and (3) a text component providing natural language descriptions of the visualization. These representations capture complementary information, addressing potential gaps that might exist if relying solely on any single modality.

D3. Support investigation of multiple embeddings and retrieval strategies This exploratory work aims to establish a baseline and lay the groundwork for multimodal retrieval of genomics data visualizations. Therefore, it is crucial that the system supports experimentation with a range of approaches. Specifically, it needs to support both unimodal and cross-modal embedding and retrieval strategies to capture complementary information across modalities and to enable the identification of the most promising approaches suitable for genomics data visualizations.

4.2 System Overview

From a user's perspective, they can search for genomics data visualizations on a user interface with their preferred query approach: example images, natural language queries, or grammar-based queries. Our retrieval system is designed to efficiently handle such queries and document searches across multiple modalities. An overview of our system is presented in Figure 1. It operates as follows:

Document Collection (Panel A): We build the document collection of genomics data visualization, where each document is represented by a triplet of three modalities: image, text, and specification, in the form of (i_k, t_k, s_k). This multimodal approach facilitates comprehensive understanding of a visualization and forms the basis for subsequent retrieval processes, which will be described in detail in Section 5.

Multimodal Representation (Panel B): For each modality, we utilize embedding approaches to transform these raw data into vector representations. Our approach integrates three embedding techniques: context-free grammar embeddings, multimodal embeddings, and textual embeddings, to combine specialized knowledge from the grammar structure and general knowledge from a multimodal biomedical foundation model and a large language model. Further details of the embedding methods are discussed in Section 6.

Multimodal Retrieval (Panel C): This component involves computing similarities between the query embedding and each document's embedding. The system uses these similarity scores to rank how relevant each document is to the search query. Documents with higher scores are considered more relevant and appear higher in the search results, which ultimately presented as a ranked list of document triplets. Section 7 presents this component in greater detail.

Our integrated system allows flexible and efficient retrieval of genomics data visualizations across various query types. To evaluate the performance of this approach, we conducted experiments with top-k retrieval accuracy. These findings are outlined in detail in Section 8.

5 Document Collection

5.1 Approaches

Each Gosling specification can be represented as a parse tree, generated by a set of context-free grammar rules. This presents two possible approaches for data generation: (1) a top-down approach, deriving specifications from the grammar by substituting symbols with literal values, or (2) a bottom-up approach, constructing the dataset from existing example specifications.

The primary advantage of the top-down method is its comprehensive coverage of all grammar-possible scenarios. However, this approach presents significant drawbacks: generated specifications often don't accurately reflect real-world genomics data requirements, and it produces numerous rules—many redundant or unnecessary—requiring substantial effort to filter negative examples and resulting in sparse vector encodings where examples utilize only small portions of the possible rule space.

The bottom-up approach, starting with specific examples and extracting only rules present in the collection, grounds the system in actual specifications rather than theoretical possibilities. While this might miss some valid but unobserved constructions, we can mitigate this limitation by referencing genomic visualization taxonomy by Nusrat et al. [26] to ensure coverage of essential rules. After careful evaluation of tradeoffs, we selected the bottom-up approach for its practical advantages in our genomics visualization context.

5.2 Collecting Gosling Specifications

We collect Gosling specifications from diverse sources, including Gosling documentation and gallery examples[1], training datasets for image analysis algorithms [39], and examples from an interactive multiscale visualization for structural variation in human genomes [19]. This collection represents a wide range of examples and real-world applications that employ Gosling for genomics data visualization. In this work, we utilize the Gosling declarative grammar [21] to structure our specifications since it enables users to create expressive visualizations while offering robust file format handling, which is an essential feature in genomic data analysis pipelines.

We collected seed examples across approximately 50 different categories, ranging from basic marks (e.g., line chart, bar chart) to domain-specific visualization such as single-cell epigenomics and structural variation analysis, to real-world applications such as Breast Cancer Variants and SARS-CoV-2 (see Figure 2).

For each category, we augment the seed example to create variations with different visual channels. Augmentations include: permutations of mark types (such as area, bar, line, point, triangle for gene annotation, arc for genome interaction), color (categorical and continuous color maps), arrangement (vertical, horizontal, serial, parallel), layout (circular, linear), and different permutations of views within a multiple-view visualization. This resulted in 3,200 visualizations.

After collection of specifications, we generated corresponding texts and images, as shown in Figure 3. Having the specification as the single source of truth has several advantages. First, as all generated texts and images derive their content from the same specification, it guarantees consistency and uniformity between all outputs. Second, this enables efficient updates once the specification is changed, minimizing discrepancies. The details of generating texts and images are outlined in the following Sections 5.3 and 5.4.

[1] https://gosling-lang.org/examples/

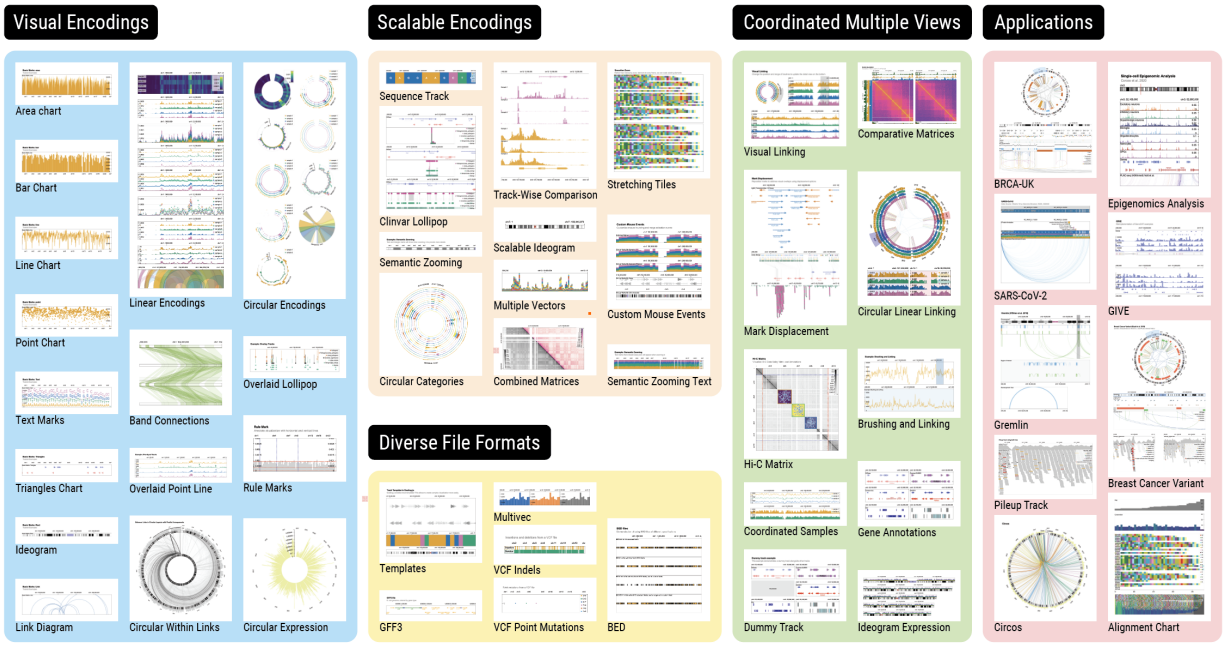

Fig. 2: The approximately 50 categories of genomics data visualizations that we collected. These categories range from common visualization types to real-world applications, such as breast cancer variants of human genomes [19].

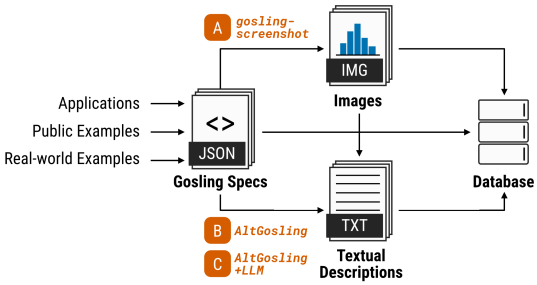

Fig. 3: The collection of Gosling specifications in three modalities. Gosling specifications from a wide range of examples and real-world applications are collected. (A) We generated images from the Gosling specifications using an open source script, i.e., `gosling-screenshot`. Two versions of textual descriptions are constructed using (B) AltGosling [35] and (C) in combination with a LLM.

5.3 Generating Text Descriptions

We generated text descriptions from Gosling specifications using two approaches (Figure 3B–C). First, we used AltGosling [35], a grammar-based tool which creates alt text descriptions for Gosling visualizations, aimed at improving accessibility for people who are blind and visually impaired. Furthermore, we generated a second set of descriptions with an LLM based on the AltGosling descriptions, the specifications, and images.

5.3.1 AltGosling

We used AltGosling (v0.2.4) to generate text descriptions for all visualizations. AltGosling fetches both an initial alt text and subsequently updates this with retrieved data. Because the purpose of our text descriptions is to query the visualization primarily, we opted to retrieve the initial alt text without fetching the underlying data of the visualization. We set up a pipeline for bulk generation of processed specifications and alt texts. The lengths of the texts varied with the number of tracks and views (mean 1691 characters, standard deviation 2169, median 563).

5.3.2 LLM-integrated Approach

For the task of visualization querying, we hypothesized that combining AltGosling texts with an LLM would create more robust descriptions, i.e., having better query similarity for a range of queries. AltGosling has knowledge of the underlying structure of the visualization, and in an earlier evaluation outperformed state-of-the-art LLM models in accuracy. LLMs excel in structured writing and diversity of text usage.

We queried GPT-4o [27] with the Gosling specification, a processed version of the specification, the automatic alt text from AltGosling, and the image. To provide context on the specifications, we also classified all properties of the specification, extracted from the CFG extraction process in Section 6, in one or more of the following classes: 1) data-related, 2) visual-encoding, 3) interaction-related, 4) styling, and 5) metadata, and added this classification to the query.

To improve consistency among text descriptions, we used a few-shot learning approach [29], for which we constructed four 'ideal' example descriptions (Supplementary Material), for which we added the specifications, AltGosling text, and ideal description to each query (omitting the image due to exceeding the maximum query length).

The lengths of the generated texts were shorter and more consistent than the AltGosling-only approach (mean 839 characters, standard deviation 298, median 775).

5.4 Generating Images

For efficient generation of images, we developed a batch image generation pipeline, based on a previously built open-source tool for single image input (`gosling-screenshot`[2]). This pipeline automates the process of converting multiple Gosling JSON specifications into image files. For each file, the script uses a headless browser to render the visualization in a virtual environment and captures a screenshot of the result. For technical details, see the Supplementary Material.

6 Multimodal Representations & Embedding Methods

Now that we have the raw data for each modality, we will discuss three primary embedding approaches to transform these raw modalities into vector representations. To formalize our approach, let $\mathcal{D}$ be the document collection:

$$\mathcal{D} = \{v_k \mid k = 1, 2, \ldots, N\}$$

where each visualization $v_k \in \mathcal{D}$ is represented as a triplet:

$$v_k = (i_k, t_k, s_k),$$

[2] https://github.com/gosling-lang/gosling-screenshot

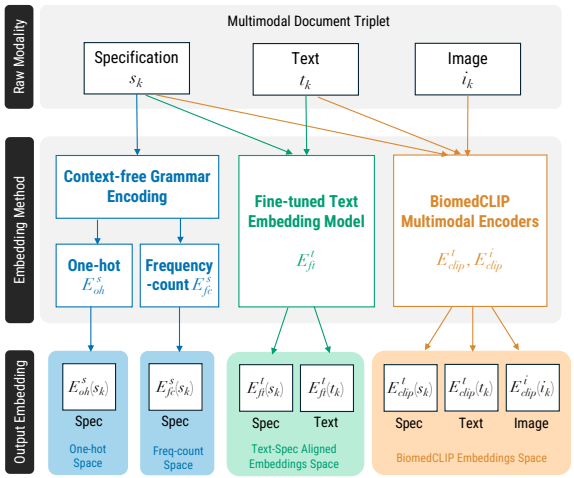

Fig. 4: Different strategies for creating embeddings from raw data for each modality. Three approaches are: Context-free grammar encoding applied to specifications, fine-tuning a general text embedding model applied to specifications and text descriptions, and a state-of-the-art foundational model BiomedCLIP for creating embeddings for all three modalities. These three approaches are described in Sections 6.1, 6.2, and 6.3, respectively.

consisting of an image $i_k \in \mathcal{I}$, text $t_k \in \mathcal{T}$, and specification $s_k \in \mathcal{S}$ for the k-th item. These modalities are semantically aligned, all describing the same underlying concept of v_k.

We propose three embedding approaches, as described in Figure 4: Context-free grammar embeddings for specifications, a fine-tuned text embedding model for texts and specifications, and BiomedCLIP multimodal foundation model for all three modalities. These three approaches are described in the following Sections 6.1, 6.2, and 6.3.

A general notation for an embedding function E transforming raw data into an embedding can be defined as follows:

$$\mathrm{E}^{modality}_{method}(input) \in \mathbb{R}^{dimension} \tag{1}$$

where *modality* specifies the types of input (image, text, or specification), *method* refers to the specific technique used to create embedding, *input* is the actual data being transformed, and *dimension* is the dimensionality of the resulting vector space.

6.1 Context-free Grammar Embeddings for Specifications

To vectorize Gosling specifications, we use a context-free grammar (CFG) parsing approach, which is a formal system in which production rules define how symbols combine recursively. A production rule has a left-hand side (LHS) representing a single element, and a right-hand side (RHS) defining how that element can be expanded, which applies regardless of the surrounding symbols (hence the name "context-free"). This approach, adapted from ChartSeer [47] and the Grammar Variational Autoencoder (GVAE) concept [10], effectively captures the hierarchical structure (e.g., a `track` containing `mark` and `size`) while enabling systematic extraction of grammar patterns.

The CFG approach for creating embeddings from Gosling specifications is presented in Figure 5. Panel (A) shows a highly abstracted Gosling specification. The full specification can be seen as the Cytoband example[3] on the Gosling editor. The corresponding image of the full specification is shown in Panel (B). In this example, there are five parallel chromosome views, listed from one to five in Panel (A). There are two tracks in view (1), each with a different dataset. The first track features a multivec dataset, and the second track includes a CSV dataset containing cytogenetic band data. Under the second track, there is a nested track for band visualization, including three different mark types: `rect`, `triangleRight` and `triangleLeft` for visualizing the ideogram.

[3] https://gosling.js.org/?example=CYTOBANDS

The adapted algorithm effectively handled hierarchical object nesting but was not able to process arrays of unnamed elements, which is crucial for representing collections like tracks or views in Gosling specifications. This limitation is significant because genomics visualizations typically comprise multiple views containing multiple tracks. While traditional CFG rules work with nested objects using named properties, array elements lack explicit identifiers to serve as non-terminals in subsequent rules. As shown in Figure 3(A), a singular 'view' within an array has no name to reference it in following production rules. Therefore, we developed an improved algorithm to explicitly address this structural characteristic, described below.

6.1.1 Internal Node Insertion

We propose a simple yet effective method for adapting CFG extraction to handle arrays of enumerable elements. Our improved algorithm introduces *internal nodes* that represent individual, unnamed elements within arrays like `views`, `tracks`, and `dataTransforms`. Starting with `root`, we establish the first rule: `root` → `views` + `layout` + `arrangement`. When encountering the `views` array, our algorithm applies internal node insertion as illustrated in Panel (C) of Figure 5, where named nodes appear in solid blue color and previously unnamed nodes appear hollowed. An unnamed node (Panel C1) is assigned the name *view* (Panel C2), using the singular form of its parent node's name (`views`). This newly named node (in solid green) is then inserted into the list of non-terminal symbols and continues in the production rule chain, starting with `view` → `tracks` + `xDomain`, for example, as in Panel C2.

6.1.2 Extracting CFG Rules

The extraction of CFG rules follows a systematic approach to convert visualization specifications into structured grammatical rules while preserving semantic meaning. Panel (D) of Figure 5 represents a snippet of the CFG rules extracted from the specification in Panel (A). For production rules where the RHS is a defined value accepted by the LHS in the grammar, the RHS is kept intact, e.g., `arrangement` → `"parallel"`. Otherwise, the RHS is substituted by a common token: `NUMBER`, `STRING`, similar to the approach used in [47], with a new addition of `ARRAY`. Our method parses through the entire structure of the specification, especially `data` field (which was excluded in ChartSeer [47]), since it encodes important information about file format, a crucial aspect of genomics visualizations. Additionally, we extract production rules for *interactions*, which form an important aspect in interactive visualizations but were less addressed in previous work. These interactions include brushing via linking ID, zooming, and coordinated multiple views.

In total, we extracted 707 unique CFG rules from our document collection $\mathcal{D}$, with 205 distinct LHSs. The most complex specification in our collection contained 1633 rules. Gosling introduces 32 foundational rules to support construction of visualizations according to the taxonomy by Nusrat et al. [26], which form the basis for diverse genomics data visualizations, including data types, marks, arrangements, orientations, alignments and layouts. Our extracted CFG rules from collection $\mathcal{D}$ fully cover these 32 rules. The Supplementary Material provides a detailed description of these rules.

6.1.3 One-hot and Frequency-count Encoding

Panels (E) and (F) demonstrate two encoding methods using the CFG: one-hot and frequency-count. One-hot encoding captures the presence or absence of each CFG rule, while frequency-count encoding represents the prevalence of each rule by its frequency of occurrence. Previous work [3, 12, 47], including the original GVAE [10], used one-hot encoding with binary values. Additionally, we propose the use of frequency-count encoding to represent the distribution of CFG rules within the specification, better characterizing the nested structure and common patterns of these highly-nested specifications.

For example, in Figure 5: Rule `arrangement` → `"parallel"` occurs once, resulting in a value of 1 in both encoding approaches. However, rule `view` → `tracks` + `xDomain` occurs once in *each* of the five views, leading to a value of 1 in one-hot encoding but a value of 5 in

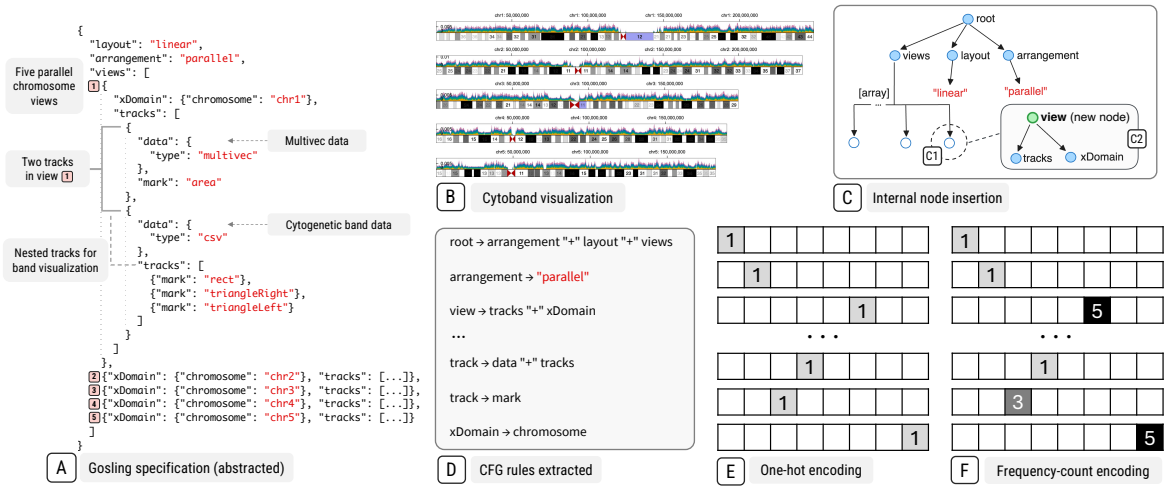

Fig. 5: The CFG approach for creating embeddings from Gosling specifications. (A) Presents abstract Gosling specification with 5 parallel chromosome views (1-5), with view (1) containing two tracks (multivec and CSV formats) and a nested track for band visualization. Displays the visualization of the complete specification. (C) Illustrates algorithm improvement for handling unnamed element arrays like `views`, showing before (C1) and after (C2). (D) Represents a snippet of the CFG rules extract from (A). Panel (E) and (F) demonstrate the two encoding methods with CFG: One-hot and frequency-count. While the former captures the presence or absence of each CFG rule, the latter represents the prevalence by frequency of each CFG rule. Example: `arrangement → "parallel"` gets value 1 in both methods, while `view → tracks + xDomain` gets value 1 in one-hot but value 5 in frequency-count (reflecting its occurrence in each view).

frequency-count encoding. To formalize this, using the notation from notation 1 for a specification s_k and our extracted set of CFG rules:

One-hot encoding:

$$\mathrm{E}^{s}_{oh}(s_k) = [u_1, u_2, ..., u_{d_{oh}}] \in \{0,1\}^{d_{oh}},$$

where u_i indicates the presence (1) or absence (0) of the i^{th} rule in s_k.

Frequency-count encoding:

$$\mathrm{E}^{s}_{fc}(s_k) = [w_1, w_2, ..., w_{d_{oh}}] \in \mathbb{N}_0^{d_{fc}},$$

where w_i is the count of occurrences of the i^{th} rule in s_k.

Both resulting vector spaces have $d_{oh} = d_{fc} = 707$ dimensions, as we extracted 707 CFG rules as per the section above. This represents a significant expansion in dimensionality compared to established visualization systems that employ CFG-based approaches. For instance, ChartSeer [47], GoTree [12], and Chart2Vec [3] each work with approximately 60 CFG rules, demonstrating their effective optimization for their respective domains. The substantial number of rules in our work reflects both the inherent complexity and expressiveness of the Gosling grammar, which was specifically designed to accommodate the rich diversity of genomics visualizations, as well as the variety present in our dataset.

6.2 Fined-tuned Text Embedding Model

6.2.1 Overview

To facilitate retrieval across text and specifications (**D1**), we employ textual embedding models pretrained on extensive natural language datasets. These embedding models convert textual data into semantic embeddings, enabling retrieval based on embedding similarity. Trained on large-scale textual data, these models exhibit strong generalization capabilities across diverse text types. Leveraging the semantic knowledge encapsulated in embeddings learned from vast datasets, we propose utilizing textual embedding models for effective retrieval between text and specifications.

However, in this new domain, specifications represent a novel set of rules explicitly designed for visualization, resulting in a significant domain gap relative to textual data handled by existing embedding models. To bridge this gap, we further fine-tune pretrained textual embedding models using our collected text-specification dataset. This strategy enables us to utilize the embedding models' general semantic understanding while effectively reducing the domain gap, making them suitable for accurate text-specification retrieval.

6.2.2 Encoder

Specifically, we utilize the gte-Qwen2-1.5B-instruct model [14], an embedding model derived from the Qwen2-1.5B large language model [43]. The base model employs a 28-layer Transformer architecture, where multi-head self-attention layers alternate with Mixture-of-Experts (MoE) feed-forward networks (FFNs). The query text is first tokenized and then fed into the model. Tokens are processed in an autoregressive manner, with the model generating one token at a time, each conditioned on the previously generated tokens. The final token output, derived from the full sequence of input tokens, is used as the embedding representation of the query text. The embeddings are then compared to the preprocessed embeddings to identify the most relevant data.

6.2.3 Loss Function

We fine-tune the model using the multiple negatives ranking loss [6] with paired text-specification data. Positive pairs are extracted from our dataset by pairing each specification with its corresponding descriptive text. The multiple negatives ranking loss for a single example is defined as:

$$\mathcal{L} = -\log\left(\frac{\exp(sim(\mathbf{a},\mathbf{p}))}{\sum_j \exp(sim(\mathbf{a},\mathbf{p}_j))}\right),$$

where $\mathbf{a}$ is the embedding of the *anchor* (e.g., a sentence), and $\mathbf{p}$ is the embedding of its corresponding *positive* (e.g., a specification). The index j ranges over all positive samples in the batch. The similarity function $sim(\mathbf{a},\mathbf{p})$ is the cosine similarity:

$$sim(\mathbf{a},\mathbf{p}) = \frac{\mathbf{a}\cdot\mathbf{p}}{\|\mathbf{a}\|\,\|\mathbf{p}\|} \tag{2}$$

This loss encourages each anchor to be more similar to its own positive than to any other positive in the batch, which are treated as implicit negatives. As a result, it improves the model's ability to retrieve the correct specification given a query sentence.

6.2.4 Training Settings

We load the pretrained model weights from gte-Qwen2-1.5B-instruct model and further fine-tune the model with our collected text-specification data pairs. We train the model with a batch size of 256 for 8 epochs. The training uses the AdamW optimizer [16]and a cosine

learning rate scheduler with a warmup ratio of 10%. The initial learning rate is 2e-5.

The fine-tuned text embedding model provides a unified function $\mathrm{E}^{t}_{ft}(x) \in \mathbb{R}^{d_{ft}}$, where x can be either text t_k or specification s_k, with $d_{ft} = 1536$ embedding both text and specifications into the same vector space, allowing direct comparison between these two modalities.

6.3 BiomedCLIP Multimodal Encoder

In this section, we utilize the multimodal biomedical foundation model BiomedCLIP [46] to generate embeddings for all three modalities: images, text, and specifications, within a shared semantic space (**D2**). The BiomedCLIP embedding space is trained on a vast dataset of 15 million biomedical image-text pairs, extracted from 4.4 million articles. BiomedCLIP uses a vision encoder for images and a text encoder for both text and specifications (which are treated as raw text), resulting in a unified embedding space where representations of all three modalities are directly comparable.

The image encoder transforms an input image, i_k, into the corresponding embedding:

$$\mathrm{E}^{i}_{clip}(i_k) \in \mathbb{R}^{d_{clip}}$$

The text encoder processes both text inputs, t_k, and specification inputs, s_k, generating their respective embeddings:

$$\mathrm{E}^{t}_{clip}(t_k) \in \mathbb{R}^{d_{clip}}, \text{ and } \quad \mathrm{E}^{s}_{clip}(s_k) = \mathrm{E}^{t}_{clip}(s_k) \in \mathbb{R}^{d_{clip}}$$

where $d_{clip} = 512$.

As stated in design consideration **D3**, we aim to experiment with different strategies. Through approaches such as a fine-tuned text embedding model or the BiomedCLIP vision-language model, we seek to test whether the raw format of specification is a promising approach compared to the more feature-engineered approach used in CFG. For an intuitive view of the embedding spaces, UMAP visualizations of all embeddings are included in the Supplementary Material.

7 Multimodal Retrieval

7.1 Retrieval Process: Overview

Our retrieval process involves two main steps. First, we embed the input query $q^{(m)}_j$ using the appropriate embedding function for its modality $m \in \{i,t,s\}$, as demonstrated in Section 6. We then compute similarities between this query embedding and document components using the similarity modeling strategies outlined in Section 7.2. These similarities are used to rank all N documents in collection $\mathcal{D}$, with higher scores indicating greater relevance.

A key feature of our system is that it always retrieves a ranked list of complete triplets for each result, regardless of the modality or embedding space used for similarity computation. This approach allows us to maximize the information available to users while leveraging the strengths of multimodal representation. Our multimodal retrieval system also enables both within-modality and cross-modality comparison strategies, as illustrated in Figure 6. This figure shows the possible combinations of modalities for query input (rows) and document (columns) comparison.

7.2 Similarity Modeling

7.2.1 Within-modality Similarity

The most straightforward approach occurs when the modality of the query matches one of the document components. In these cases, we compute similarity directly between embeddings of the same type $m \in \{i,t,s\}$, with the cosine similarity function:

$$\mathrm{sim}_{within}(q^{(m)}_j, m_k) = \frac{E^m(q^{(m)}_j) \cdot E^m(m_k)}{\|E^m(q^{(m)}_j)\| \, \|E^m(m_k)\|}, \tag{3}$$

where E^m represents the appropriate embedding function for modality m. For example, a specification query is compared directly with the specification document components using the same embedding method E^s for direct matching; in our case, it can be either: one-hot embeddings E^s_{oh}, frequency-count embeddings E^s_{fc}, fine-tuned model $E^s_{ft} = E^t_{ft}$, or BiomedCLIP text encoder $E^s_{clip} = E^t_{clip}$.

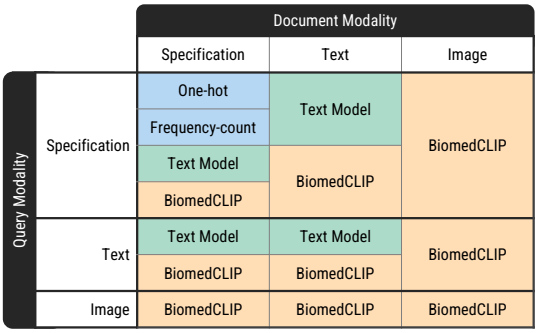

Fig. 6: Modality pairings for query (rows) and document (columns). For specification, four different embedding methods can be applied: One-hot encoding, Frequency-count encoding, Fine-tuned Text Model, and BiomedCLIP, with the latter two processing specifications as raw text. For text, the applicable methods are the Fine-tuned Text Model and BiomedCLIP, while for images, BiomedCLIP is used.

7.2.2 Cross-modality Similarity

While within-modality matching is intuitive, many real-world scenarios require comparing across different modalities (**D1**). For cross-modal retrieval, we define the comparison between query modality m and target modality m' within the shared space as follows:

$$\mathrm{sim}_{cross}(q^{(m)}_j, m'_k) = \frac{E^m_{shared}(q^{(m)}_j) \cdot E^{m'}_{shared}(m'_k)}{\|E^m_{shared}(q^{(m)}_j)\| \, \|E^{m'}_{shared}(m'_k)\|}, \tag{4}$$

where E^m_{shared} and $E^{m'}_{shared}$ denote the embedding functions that map content from modalities m and m' respectively into a common shared embedding space. For example, when comparing a text query with specification components in a document, we can use embedding models that handle both modalities, such as our fine-tuned model E^t_{ft} or BiomedCLIP E^t_{clip}, both of which process specifications as raw text.

7.2.3 Ensemble Approach

Given the diversity of embedding methods and similarity computation strategies, we recognize that no single approach is optimal for all query types and documents. To leverage the complementary strengths of different methods, we compute multiple similarity scores and select the highest. As shown in Figure 6, for each input modality (row), we compare all similarity values across that row to determine the most effective retrieval approach.

$$sim_{\text{ensemble}}(q^{(m)}_j, v_k) = \max\{\mathrm{sim}_{within}(q^{(m)}_j, m_k), \mathrm{sim}_{cross}(q^{(m)}_j, m'_k)\},$$

where each components in the max function corresponds to equation (3) and (4) in the previous sections. This ensemble approach allows us to select the most effective similarity measure for each query-document pair. For instance, when processing a text query, the system may either compare it directly with document text components via within-modality similarity, or leverage cross-modal mappings to match against image or specification components, whichever produces the highest confidence match.

7.3 User Interface

We designed and implemented a prototype user interface that enables users to retrieve genomics data visualization examples using the three modalities. The overall interface consists of two panels (Figure 7A–B).

Table 1: The quantitative comparison of top-k retrieval accuracies. The highest accuracies for given query modalities are denoted with a bold font. The document modalities with asterisk symbols (*) indicate the within-modality search.

Query Modality	Document Modality	Embedding Method	k=1	k=2	k=3	k=4	k=5
Spec	**Spec***	**Frequency Count**	**0.6127**	**0.7205**	**0.7647**	**0.8235**	**0.8382**
		One Hot	0.6127	0.6617	0.7500	0.8039	0.8137
		BiomedCLIP	0.3333	0.4068	0.4509	0.4852	0.5147
		Text Model	0.5882	0.6029	0.6225	0.6373	0.6373
	Text (AltGosling)	BiomedCLIP	0.0294	0.0294	0.0294	0.0294	0.0294
		Text Model	0.2353	0.3186	0.4069	0.4461	0.4559
	Text (AltGosling + LLM)	BiomedCLIP	0.0294	0.0588	0.0735	0.0882	0.1029
		Text Model	0.3627	0.4706	0.5294	0.5637	0.5784
	Image	BiomedCLIP	0.0343	0.0343	0.0441	0.0441	0.0441
Text (AltGosling)	Spec	BiomedCLIP	0.0245	0.0441	0.0441	0.0490	0.0490
		Text Model	0.1667	0.2059	0.2353	0.2500	0.2549
	Text (AltGosling)*	**BiomedCLIP**	**0.4362**	**0.5343**	**0.5784**	**0.6323**	**0.6519**
		Text Model	0.2696	0.3578	0.4608	0.4804	0.4951
	Text (AltGosling + LLM)	BiomedCLIP	0.2696	0.3186	0.3725	0.4117	0.4362
		Text Model	0.1961	0.2745	0.3186	0.3529	0.3824
	Image	BiomedCLIP	0.0784	0.1176	0.1666	0.1666	0.1715
Text (AltGosling + LLM)	Spec	BiomedCLIP	0.0392	0.0637	0.0784	0.0931	0.0980
		Text Model	0.3529	0.3824	0.4069	0.4412	0.4510
	Text (AltGosling)	BiomedCLIP	0.2205	0.2794	0.3235	0.3382	0.3480
		Text Model	0.2598	0.3627	0.4510	0.4657	0.4706
	Text (AltGosling + LLM)*	**BiomedCLIP**	**0.4509**	**0.5539**	**0.6127**	**0.6421**	**0.6764**
		Text Model	0.3824	0.4755	0.5147	0.5637	0.5980
	Image	BiomedCLIP	0.2254	0.2745	0.3088	0.3333	0.3627
Image	Spec	BiomedCLIP	0.0147	0.0196	0.0196	0.0196	0.0294
	Text (AltGosling)	BiomedCLIP	0.1421	0.1715	0.1862	0.1862	0.1911
	Text (AltGosling + LLM)	BiomedCLIP	0.2107	0.3137	0.3725	0.4264	0.4656
	Image*	**BiomedCLIP**	**0.6519**	**0.6862**	**0.7107**	**0.7303**	**0.7401**

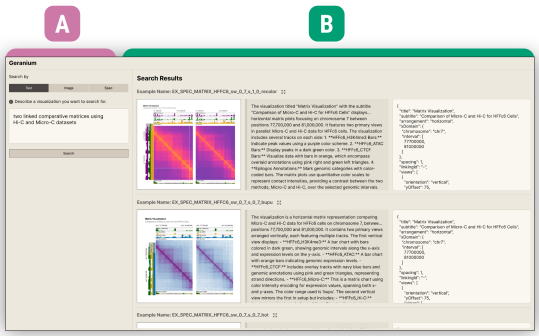

Fig. 7: The overall user interface. (A) Users can search for examples of genomics data visualization using three modalities: textual descriptions, images, and Gosling [21] specifications. (B) The search results are shown as a gallery, displaying examples with all three modalities.

The panel on the left enables users to provide three input modalities (i.e., textual descriptions, images, and the Gosling [21] specifications). For example, users can describe a visualization that they wish to search for, such as data types used in visualizations or visualization types (e.g., "`two linked comparative matrices using Hi-C and Micro-C datasets`"). Using the image modality as an input, users can alternatively use an actual image, such as a publication figure. Users can also directly upload (or copy and paste) a Gosling specification, such as a partial specification that contains a visualization that users wish to use in addition to other visualizations. Once users press a Search button, the panel on the right displays the visualization examples based on the input query. The results are shown as a gallery with visualization thumbnails, enabling easy visual exploration [17]. All other modalities, Textual Descriptions and JSON Specifications, are displayed together, which helps users to find the right visualization example that they are looking for. For example, the textual description provide additional details of a given visualization, such as datasets used and user interactions supported. The three modalities can be shown in detail in a dedicated pop-up view on demand (i.e., upon clicking on a single example of users' interest, the corresponding image, textual description, and specification are shown in detail on a maximized view. Once users found a visualization example, they can export its Gosling specification and use them in existing visualization authoring ecosystems (e.g., the Gosling online editor[4] or the Blace graphical user interface [20]). The landing page also includes a gallery to browse the dataset (Supplementary Material).

7.3.1 Implementation

The prototype system implements front-end for user interfaces and back-end for using ML models and storing data. The front-end is implemented in JavaScript using React 18 [25], and the back-end is implemented in Python using Flask [5]. The server stores all genomics data visualizations we collected in the three modalities, as well as their embeddings. The input textual descriptions, images, and specifications from the user interface is sent to the server via Flask, and the server finds the most similar visualization examples. The back-end uses BiomedCLIP [46] and also implements the CFG rule extraction to generate the embedding of the given input. The source code is publicly available at `https://github.com/huyen-nguyen/geranium`.

8 Evaluation

To evaluate the effectiveness of our system, we created a test suite and applied standard information retrieval metrics that measure the accuracy of the top-*k* retrieved items.

We define a set of unimodal queries as follows:

[4] `https://gosling.js.org`

$$\mathcal{Q}^{(m)} = \left\{ q_j^{(m)} \mid j = 1, 2, ..., Q \right\},$$

where Q is the total query count. Each query $q_j^{(m)}$ retrieves its relevant target triplets (ground truth). With document collection $\mathcal{D}$, the ground truth for query $q_j^{(m)}$ is defined as: $\mathcal{G}_j \subseteq \mathcal{D}$, including the original triplet and semantically similar documents.

From collection $\mathcal{D}$, we selected representative visualizations for querying. For singular visualizations, we removed titles and applied one random modification when possible: 1) removing the last category, 2) doubling `binSize`, 3) changing axis position, 4) removing axis, or 5) removing legend. For composite visualizations, we included only one or multiple subsets. From these modified specifications, we generated text descriptions and images as in Section 5, creating 204 queries per modality.

With mapping function ϕ where visualization v_k maps to query index $\phi(k) \in \{1, 2, ..., Q\}$, for each original visualization $v_k = (i_k, t_k, s_k)$ in $\mathcal{D}$, we created queries in three modalities: $(q_{\phi(k)}^{(i)}, q_{\phi(k)}^{(t)}, q_{\phi(k)}^{(s)})$ representing image, text, and specification queries. This preserves connections between queries and source visualizations.

To extend query ground truth, we identified semantically similar visualizations. If v_k derives from seed example $v_{seed} \in \mathcal{D}$, we extend ground truth for queries $q_{\phi(k)}^{(m)}$ by including v_{seed} and any $v_j \in \mathcal{D}$ derived from v_{seed} where i_j is visually similar to i_k.

8.1 Top-k Retrieval Accuracy

For a given query $q_j^{(m)}$, if the correct answer is found within the top k returned items, then the retrieval is considered accurate. Top-k retrieval accuracy measures the ratio of instances for which the true or relevant item is included within the top k predicted results. This metric allows us to evaluate how often users will find relevant results within a reasonable number of items to review, which directly relates to the usability of the retrieval system.

To formalize this, given Q as the total number of queries evaluated, top-k retrieval accuracy is defined as:

$$\mathrm{A_k} = \frac{1}{Q} \sum_{j=1}^{Q} \mathrm{I}(\mathrm{g_j} \in \mathrm{R_{k,j}}),$$

where g_j specifies the ground truth label for query $q_j^{(m)}$, $R_{k,j}$ represents the top k items retrieved for that query, and $\mathrm{I}(\cdot)$ is the indicator function, which returns 1 if the condition inside it is true, and 0 otherwise. In this case, it checks if the ground-truth label g_j is within the top k predictions $(R_{k,j})$.

8.2 Quantitative Comparison

Table 1 summarizes the top-k retrieval accuracy with different embedding methods. Overall, within-modality queries showed higher top-k accuracies compared to cross-modality queries. The image-to-image retrieval using BiomedCLIP showed the highest accuracy when $k = 1$ (0.65), while spec-to-spec retrieval using the frequency count encoding showed the best performance with higher k, reaching 0.84 with $k = 5$. Comparing two types of encoding for the context-free grammar (CFG) rules, the frequency count encoding showed slightly better performance than the one hot encoding for all five different k options, indicating that frequency count encoding better captures the characteristics of genomics data visualization (e.g., multiview aspects). Compared to the CFG approach, treating specifications as textual descriptions showed much lower accuracies using both BiomedCLIP and trained ML text model. This isn't surprising given the complexity of the JSON specifications, as illustrated in prior studies [30]. Between two versions of textual descriptions, we found that the version with a LLM showed overall better accuracies, likely due to flexibility in word choice leading to robuster embeddings. Interestingly, BiomedCLIP outperformed our fine-tuned text model. This is potentially related to how Text Model depends on specs in the raw input format, compared to using context-free grammar rules.

For similarity score comparisons, see Supplementary Material. Multiple embeddings can have the same similarity score, in which case there is no intuitive ordering for the documents. They are returned ordered based on previous ordering, e.g. alphabetical. This influenced the top-k accuracy, and is a caveat in the document retrieval strategy. In 30.2% of queries an arbitrary ordering affected the top-k accuracy score (with similar percentages for each query modality).

9 Discussion

This paper introduced and evaluated a multimodal database system designed to address the challenge of retrieving relevant genomics data visualizations, using multimodal representations. The ensemble similarity modeling combine the cross-modality and within-modality approaches, showing a complete picture of the combined effort. Our evaluation measured by top-k accuracy reveals several key trends. Within-modality retrieval consistently outperformed cross-modality searches. Notably, specification-to-specification search was most effective with frequency-count encoding ($A_5 = 0.84$), followed closely by one-hot encoding ($A_5 = 0.81$). The two CFG-based encodings do not require training, but demonstrates comparable results with other methods using language model, comparable and better to the fine-tuned text embedding model ($A_5 = 0.64$), and BiomedCLIP ($A_5 = 0.51$), where specifications are treated as plain text. Generally, the foundational pre-trained BiomedCLIP model outperformed our fine-tuned text embedding model. For text queries, LLM-aid descriptions offered an advantage, where the delineation of specific classes (data, visual-encoding, interaction, styling, and metadata) also help steer the LLM to the right direction. Comparing our results to Chart2Vec [3] ($A_2 = 0.63$, $A_5 = 0.73$), our approach ($A_2 = 0.72, A_3 = 0.76$) with frequency-count encoding performs slightly better, but does not require training effort.

The prototype implementation described in this work relies on visualizations rendered with their original published data. A practical deployment or broader application of this system, however, might require ingesting visualizations paired with potentially synthetic datasets or representative examples rather than specific data from source publications. The specification results from this search can be used as scaffolds for authoring, where the users can directly modify the specifications and fill in their own data and styles (color maps, fonts, etc.). Overall, our multimodal approach demonstrates promising results for visualization retrieval in genomics, offering a robust foundation for future visualization recommendation and authoring systems.

10 Conclusion

In this paper, we presented a multimodal retrieval system for genomics data visualizations, providing flexible visualization retrieval. To explore ways to support effective visualization retrieval, we adopted and extended different ways to represent genomics data visualizations, such as using a context-free grammar (CFG) and a state-of-the-art multimodal foundational model (BiomedCLIP). Using such approaches, we collected 3,200 interactive genomic visualizations—spanning over 50 categories and covering a wide range of applications and design variations—and built a prototype user interface that allow users to search for genomics data visualizations using three modalities. We performed evaluation using top-k retrieval accuracy, and our notable observation is that, while specification did not translate well as rich text in machine learning models, the CFG approach applied to JSON specifications showed promising retrieval results and is not computational heavy. Although built primarily for genomics data visualizations, the approach proposed in this work can be extended and transferred to other application domains. This transfer could be applied to other visualization representations, such as Vega-Lite [32] with similar JSON structure, Tableau's exported XML as explored previously [28], or other text formats that characterize visualizations.

Acknowledgments

The authors wish to thank Etowah Adams and Viet Lai for their support. With gratitude, the first author wishes to thank Professor Susan Mengel for her insightful teaching on Information Retrieval. This work was supported in part by the National Institutes of Health (R01HG011773, K99HG013348, and UM1HG011536).

References

[1] C. Chen, H. K. Bako, P. Yu, J. Hooker, J. Joyal, S. C. Wang, S. Kim, J. Wu, A. Ding, L. Sandeep, et al. Visanatomy: An svg chart corpus with fine-grained semantic labels. *arXiv preprint arXiv:2410.12268*, 2024. 2

[2] C. Chen and Z. Liu. The state of the art in creating visualization corpora for automated chart analysis. In *Computer Graphics Forum*, vol. 42, pp. 449–470. Wiley Online Library, 2023. 2

[3] Q. Chen, Y. Chen, R. Zou, W. Shuai, Y. Guo, J. Wang, and N. Cao. Chart2vec: A universal embedding of context-aware visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 2024. 1, 2, 5, 6, 9

[4] C. Diesh, G. J. Stevens, P. Xie, T. De Jesus Martinez, E. A. Hershberg, A. Leung, E. Guo, S. Dider, J. Zhang, C. Bridge, et al. Jbrowse 2: a modular genome browser with views of synteny and structural variation. *Genome biology*, 24(1):74, 2023. 2

[5] M. Grinberg. *Flask web development: developing web applications with python*. O'Reilly Media, Inc., USA, 2018. 8

[6] M. Henderson, R. Al-Rfou, B. Strope, Y.-H. Sung, L. Lukács, R. Guo, S. Kumar, B. Miklos, and R. Kurzweil. Efficient natural language response suggestion for smart reply. *arXiv preprint arXiv:1705.00652*, 2017. 6

[7] E. Hoque and M. Agrawala. Searching the visual style and structure of d3 visualizations. *IEEE transactions on visualization and computer graphics*, 26(1):1236–1245, 2019. 1, 2

[8] Y. Kim, K. Wongsuphasawat, J. Hullman, and J. Heer. Graphscape: A model for automated reasoning about visualization similarity and sequencing. In *Proceedings of the 2017 CHI conference on human factors in computing systems*, pp. 2628–2638, 2017. 2

[9] M. J. Kusner, B. Paige, and J. M. Hernández-Lobato. Grammar variational autoencoder. In D. Precup and Y. W. Teh, eds., *Proceedings of the 34th International Conference on Machine Learning*, vol. 70 of *Proceedings of Machine Learning Research*, pp. 1945–1954. PMLR, 2017. 2

[10] M. J. Kusner, B. Paige, and J. M. Hernández-Lobato. Grammar variational autoencoder. In *International conference on machine learning*, pp. 1945–1954. PMLR, 2017. 5

[11] G. Li, M. Tian, Q. Xu, M. J. McGuffin, and X. Yuan. Gotree: A grammar of tree visualizations. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pp. 1–13, 2020. 2

[12] G. Li and X. Yuan. Gotreescape: Navigate and explore the tree visualization design space. *IEEE Transactions on Visualization and Computer Graphics*, 29(12):5451–5467, 2022. 2, 5, 6

[13] H. Li, Y. Wang, A. Wu, H. Wei, and H. Qu. Structure-aware visualization retrieval. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, pp. 1–14, 2022. 2

[14] Z. Li, X. Zhang, Y. Zhang, D. Long, P. Xie, and M. Zhang. Towards general text embeddings with multi-stage contrastive learning. *arXiv preprint arXiv:2308.03281*, 2023. 6

[15] Z. Liu, C. Chen, F. Morales, and Y. Zhao. Atlas: Grammar-based procedural generation of data visualizations. In *2021 IEEE Visualization Conference (VIS)*, pp. 171–175. IEEE, 2021. 2

[16] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. 6

[17] S. L'Yi, Y. Chang, D. Shin, and J. Seo. Toward understanding representation methods in visualization recommendations through scatterplot construction tasks. In *Computer Graphics Forum*, vol. 38, pp. 201–211. Wiley Online Library, 2019. 8

[18] S. L'Yi and N. Gehlenborg. Multi-view design patterns and responsive visualization for genomics data. *IEEE transactions on visualization and computer graphics*, 29(1):559–569, 2022. 2

[19] S. L'Yi, D. Maziec, V. Stevens, T. Manz, A. Veit, M. Berselli, and others. Chromoscope: interactive multiscale visualization for structural variation in human genomes. *Nature Methods*, 2023. doi: 10.1038/s41592-023-02056-x 3, 4

[20] S. L'Yi, A. van den Brandt, E. Adams, H. N. Nguyen, and N. Gehlenborg. Learnable and expressive visualization authoring through blended interfaces. *IEEE Transactions on Visualization and Computer Graphics*, 2024. 2, 8

[21] S. LYi, Q. Wang, F. Lekschas, and N. Gehlenborg. Gosling: A grammar-based toolkit for scalable and interactive genomics data visualization. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):140–150, 2021. 2, 3, 8

[22] S. LYi, Q. Wang, F. Lekschas, and N. Gehlenborg. Gosling: A Grammar-based Toolkit for Scalable and Interactive Genomics Data Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):140–150, Jan. 2022. doi: 10.1109/TVCG.2021.3114876 1, 2

[23] S. L'Yi, Q. Wang, and N. Gehlenborg. The role of visualization in genomics data analysis workflows: The interviews. In *2023 IEEE Visualization and Visual Analytics (VIS)*, pp. 101–105. IEEE, 2023. 2

[24] T. Manz, S. L'Yi, and N. Gehlenborg. Gos: a declarative library for interactive genomics visualization in python. *Bioinformatics*, 39(1), Jan. 2023. doi: 10.1093/bioinformatics/btad050 2

[25] Meta Platforms, Inc. A React JavaScript library for building user interfaces. `https://github.com/facebook/react`. Accessed: 2025-3-31. 8

[26] S. Nusrat, T. Harbig, and N. Gehlenborg. Tasks, Techniques, and Tools for Genomic Data Visualization. *Computer Graphics Forum*, 38(3):781–805, June 2019. doi: 10.1111/cgf.13727 3, 5

[27] OpenAI, J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, and others. Gpt-4 technical report, 2024. 4

[28] M. Oppermann, R. Kincaid, and T. Munzner. Vizcommender: Computing text-based similarity in visualization repositories for content-based recommendations. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):495–505, 2020. 1, 2, 9

[29] A. Parnami and M. Lee. Learning from few examples: A summary of approaches to few-shot learning, 2022. 4

[30] G. Poesia, O. Polozov, V. Le, A. Tiwari, G. Soares, C. Meek, and S. Gulwani. Synchromesh: Reliable code generation from pre-trained language models. *arXiv preprint arXiv:2201.11227*, 2022. 9

[31] J. T. Robinson, H. Thorvaldsdottir, D. Turner, and J. P. Mesirov. igv. js: an embeddable javascript implementation of the integrative genomics viewer (igv). *Bioinformatics*, 39(1):btac830, 2023. 2

[32] A. Satyanarayan, D. Moritz, K. Wongsuphasawat, and J. Heer. Vega-Lite: A grammar of interactive graphics. *IEEE Trans. Vis. Comput. Graph.*, 23(1):341–350, Jan. 2017. doi: 10.1109/TVCG.2016.2599030 2, 9

[33] A. Satyanarayan, D. Moritz, K. Wongsuphasawat, and J. Heer. Vega-Lite: A grammar of interactive graphics. *IEEE transactions on visualization and computer graphics*, 23(1):341–350, Jan. 2017. doi: 10.1109/TVCG.2016.2599030 2

[34] V. Setlur, A. Kanyuka, and A. Srinivasan. Olio: A semantic search interface for data repositories. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pp. 1–16, 2023. 2

[35] T. C. Smits, S. L'Yi, A. P. Mar, and N. Gehlenborg. Altgosling: automatic generation of text descriptions for accessible genomics data visualization. *Bioinformatics*, 40(12):btae670, 11 2024. doi: 10.1093/bioinformatics/btae670 1, 2, 4

[36] H. Stitz, S. Gratzl, H. Piringer, T. Zichner, and M. Streit. Knowledgepearls: Provenance-based visualization retrieval. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):120–130, 2018. 2

[37] M. Sun, L. Cai, W. Cui, Y. Wu, Y. Shi, and N. Cao. Erato: Cooperative data story editing via fact interpolation. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):983–993, 2022. 2

[38] A. van den Brandt, S. L'Yi, H. N. Nguyen, A. Vilanova, and N. Gehlenborg. Understanding visualization authoring techniques for genomics data in the context of personas and tasks. *IEEE Transactions on Visualization and Computer Graphics*, 2024. 1, 2

[39] Q. Wang, X. Liu, M. Q. Liang, S. L'Yi, and N. Gehlenborg. Enabling multimodal user interactions for genomics visualization creation. *In Proc. IEEE VIS*, 2023. 3

[40] L. Wilkinson. *The Grammar of Graphics*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. doi: 10.1007/978-3-642-21551-3_13 2

[41] A. Wu, W. Tong, H. Li, D. Moritz, Y. Wang, and H. Qu. Computableviz: Mathematical operators as a formalism for visualisation processing and analysis. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, pp. 1–15, 2022. 2

[42] S. Xiao, Y. Hou, C. Jin, and W. Zeng. Wytiwyr: A user intent-aware framework with multi-modal inputs for visualization retrieval. In *Computer Graphics Forum*, vol. 42, pp. 311–322. Wiley Online Library, 2023. 1, 2

[43] A. Yang, B. Yang, B. Hui, B. Zheng, B. Yu, C. Zhou, C. Li, C. Li, D. Liu,

F. Huang, and others. Qwen2 technical report, 2024. 6

[44] Y. Ye, R. Huang, and W. Zeng. Visatlas: An image-based exploration and query system for large visualization collections via neural image embedding. *IEEE Transactions on Visualization and Computer Graphics*, 2022. 1, 2

[45] L. Ying, A. Wu, H. Li, Z. Deng, J. Lan, J. Wu, Y. Wang, H. Qu, D. Deng, and Y. Wu. Vaid: Indexing view designs in visual analytics system. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, pp. 1–15, 2024. 2

[46] S. Zhang, Y. Xu, N. Usuyama, H. Xu, J. Bagga, R. Tinn, S. Preston, R. Rao, M. Wei, N. Valluri, C. Wong, A. Tupini, Y. Wang, M. Mazzola, S. Shukla, L. Liden, J. Gao, A. Crabtree, B. Piening, C. Bifulco, M. P. Lungren, T. Naumann, S. Wang, and H. Poon. A multimodal biomedical foundation model trained from fifteen million image–text pairs. *NEJM AI*, 2(1):AIoa2400640, 2025. doi: 10.1056/AIoa2400640 7, 8

[47] J. Zhao, M. Fan, and M. Feng. Chartseer: Interactive steering exploratory visual analysis with machine intelligence. *IEEE Transactions on Visualization and Computer Graphics*, 28(3):1500–1513, 2020. 2, 5, 6