SUPPLEMENTARY MATERIAL

Show Me the Infographic I Imagine: Intent-Aware Infographic Retrieval for Authoring Support

Additional methodological details, prompt templates, system notes, and evaluation protocol summaries.

1 Overview

This supplementary document consolidates the materials deferred from the main paper and records the concrete methodological and system details needed to interpret the approach and evaluation. It focuses on four items:

1. the query-parsing prompt template and output schema referenced in Section 4;
2. the expert-specified chart-type soft-match table referenced in Section 4;
3. additional details of the interactive retrieval-and-adaptation interface described in Section 5;
4. the human-query collection and rating protocols referenced in Section 6.

For the human-written query benchmark, only the details explicitly stated in the main paper are included below. The original annotation form itself is left blank pending confirmation of the source material.

2 Query Parsing Prompt and Structured Output Schema

2.1 Five-Facet Retrieval Schema

The interactive system parses a user request into five retrieval facets: `content`, `style`, `layout`, `illustration`, and `chart_type`. The required JSON structure is:

```
<retrieval>
{
  "content": {"query": "...", "weight": 0.0},
  "style": {"query": "...", "weight": 0.0},
  "layout": {"query": "...", "weight": 0.0},
  "illustration": {"query": "...", "weight": 0.0},
  "chart_type": {"query": "...", "weight": 0.0}
}
</retrieval>
```

At inference time, the system expects exactly one `<retrieval>...</retrieval>` block, parses its JSON content, and validates that all five required keys are present. For each facet, it enforces a two-field object with `query` and `weight`, and converts `weight` to a floating-point value. Invalid or incomplete outputs are rejected and treated as parse failures.

2.2 Facet Semantics

The system prompt defines the five facets as follows.

- **content**: data subject, entities, metrics, and relationships.
- **style**: visual aesthetic, color palette, and high-level visual tone.
- **layout**: structural composition and arrangement of marks and panels.
- **illustration**: decorative elements, icons, and background graphics.
- **chart_type**: one or more coarse chart types from a fixed pool.

The chart-type pool used by the prompt and retriever contains 13 coarse labels: *Bar Chart, Line Chart, Area Chart, Radar Chart, Pie Chart, Scatterplot, Gauge Chart, Treemap, Diagram, Histogram, Range Chart, Funnel Chart, Pyramid Chart*.

2.3 Core Prompt Instructions

The main retrieval prompt gives the model three kinds of instructions.

1. **When to trigger retrieval.** Retrieval should be triggered for chart creation, redesign, layout/style inspiration, or other requests where reference examples would improve the answer quality.
2. **When not to trigger retrieval.** Retrieval should not be triggered for pure explanation, interpretation, or analysis tasks that do not need visual references.
3. **How to structure the retrieval query.** The model must output exactly one `<retrieval>` block and no extra text in the same turn.

The same prompt also defines the following weight heuristics:

- weights lie in $[0, 1]$;
- content usually receives the highest weight;
- unspecified facets use an empty query string and weight 0;
- chart type typically receives a moderate weight if specified;
- the weights do not need to sum to 1 because the retriever renormalizes them.

2.4 Prompt Template A: Initial Query Parsing

The deployed initial query-parsing prompt contains the following operative instructions.

Initial Query Parsing Prompt

You are a multimodal assistant helping users design and implement charts. You have access to an external multimodal retrieval system that can search for example chart images and SVG code based on both text and image inputs.

Your goals

- Engage with the user to understand their visualization intent.

- For direct questions about existing charts, data analysis, or chart interpretation, provide direct answers without retrieval.
- The retrieval system accepts structured query input: it requires a JSON object describing five distinct aspects of the desired chart.

When to trigger retrieval

- Trigger retrieval when the user is asking for chart creation, redesign, visual examples, style/layout inspiration, or when references would likely improve answer quality.
- Default to retrieval for ambiguous requests about how a chart should be designed or drawn.

When not to trigger retrieval

- Do not trigger retrieval when the user only needs explanation or analysis and does not need design references.
- This includes questions about existing charts, requests for data interpretation or insights, and requests for explanations of chart elements or trends.

Output requirements

- Output a single `<retrieval>` tag containing a valid JSON object with exactly the five aspects `content`, `style`, `layout`, `illustration`, and `chart_type`.
- Assign weights between 0.0 and 1.0.
- If an aspect is not mentioned, set `query` to an empty string and `weight` to 0.0.
- After outputting the retrieval block, stop and do not produce additional explanation.

2.5 Refinement Prompt

When a user is unsatisfied with the initial retrieval, a refinement prompt reuses the same five-facet schema. Its input contains:

- the conversation history,
- the previous retrieval query,
- and the user's refinement feedback.

The prompt requires the model to keep user-constrained facets specific while relaxing uncertain facets instead of over-constraining them.

2.6 Prompt Template B: Retrieval Refinement

The deployed refinement prompt contains the following operative instructions.

Retrieval Refinement Prompt

You are an expert at refining chart-retrieval queries.
The user was not satisfied with previous retrieval results. You will receive:

- conversation history,
- the previous retrieval query,

- and the user's refinement feedback.

Task

- Output an improved retrieval query in the same five-aspect schema used by the retrieval system.

Strict output requirements

- Output exactly one `<retrieval>...</retrieval>` block.
- Inside it, output valid JSON with exactly these keys: `content`, `style`, `layout`, `illustration`, and `chart_type`.
- Each key must contain a `query` string in English and a `weight` between 0.0 and 1.0.
- Do not output any text before or after the retrieval block.
- Keep user-constrained aspects specific; relax uncertain aspects instead of over-constraining them.

3 Chart-Type Soft Matching Table

Section 4 of the paper defines a chart-type kernel $\kappa(t, t')$ for soft matching among coarse chart types. Exact matches have similarity 1. Pairs not listed below default to 0.

Chart-type pair	Score	Rationale
Bar Chart – Histogram	0.9	visually very similar
Bar Chart – Funnel Chart	0.6	funnel as center-aligned bar variant
Bar Chart – Pyramid Chart	0.6	pyramid as bar-like variant
Line Chart – Area Chart	0.8	area as filled line
Line Chart – Scatterplot	0.4	trend vs. point cloud overlap
Line Chart – Radar Chart	0.3	radar as polar line chart
Area Chart – Radar Chart	0.3	filled radar resembles area chart
Area Chart – Scatterplot	0.6	used to cover bubble-like proportional-area cases
Pie Chart – Gauge Chart	0.7	circular proportion/progress similarity
Pie Chart – Diagram	0.2	some circular charts are annotated as diagrams

Chart-type pair	Score	Rationale
Funnel Chart – Pyramid Chart	0.8	inverted-shape relationship

3.1 Scoring Details

At retrieval time, the above table is converted into a symmetric similarity matrix and combined with the facet-based embedding scores. The relevant scoring details are:

- query chart types are read from a comma-separated string in the `chart_type.query` field;
- exact matches receive score 1 through the identity matrix;
- the zero-th row/column is reserved for "unknown" or missing chart types and forced to similarity 0;
- the chart-type score is normalized together with the other facet weights before final fusion.

3.2 Chart-Type Tree Construction and Organization

The interface exposes chart types through a hierarchy with 13 root categories, matching the 13 coarse chart types used by the retriever. This hierarchy is not inferred automatically from the corpus. Instead, it is manually constructed from the original chart-type statistics using semantic grouping and naming patterns, and then materialized as a tree together with a flat lookup table. Conceptually, this taxonomy has three levels: coarse root categories, intermediate grouping nodes, and fine-grained chart-type leaves. In the stored hierarchy data, the first two levels are explicit tree nodes, while the third level is recorded as the `chart_types` list attached to each grouping node.

The construction process has four steps:

1. **Start from the original chart-type inventory.** Each fine-grained chart type is first associated with its corpus count from the chart-type statistics.
2. **Define a manual coarse-to-fine hierarchy.** A hand-designed hierarchy groups fine-grained chart types under the 13 coarse roots used by the retriever. For example, bar-like variants are grouped under *Bar Chart*, while treemap-like variants are grouped under *Treemap*.
3. **Recursively aggregate node statistics.** For every tree node, the construction procedure stores a display name, tree path, depth level, the directly mapped fine-grained chart types, the aggregated corpus count, the percentage in the full corpus, and its child nodes. Counts for internal nodes are obtained by summing over descendants.
4. **Build and validate a flat mapping.** After the tree is built, every fine-grained chart type is mapped to its tree path and ancestor list. The construction step then verifies whether any original chart type was left unmapped.

The tree serves two complementary purposes:

- **Retrieval-time chart-type specification.** The model predicts one or more coarse chart types in the `chart_type` facet. These are the same 13 root categories used in retrieval scoring.

- **Interface-time result filtering.** The UI exposes a tree for narrowing retrieved candidates by category while preserving the relationship between coarse and fine-grained types.

Structurally, the hierarchy mixes two cases.

- Some roots directly serve as final chart types without additional children, such as *Histogram*, *Range Chart*, *Funnel Chart*, and *Pyramid Chart*.
- Other roots expand into level-2 groups that collect fine-grained variants. For example, *Bar Chart* is split into groups such as *Vertical Bar Chart*, *Horizontal Bar Chart*, *Circular Bar Chart*, and *Radial Bar Chart*; *Area Chart* includes *Basic Area Chart* and *Proportional Area Chart*; and *Treemap* includes variants such as *Voronoi Treemap*.

To support filtering and result display, the hierarchy also stores a flat mapping from each fine-grained chart type to its path in the tree. This makes it possible to recover a parent category for a retrieved leaf type and display both its fine-grained label and its coarse parent label in the interface. The current hierarchy statistics indicate 13 roots, 70 mapped chart types, and no unmapped original types.

4 Additional Details of the Interactive Interface

4.1 Visible Retrieval Controls

The interface exposes retrieval state directly to users. The current interaction design includes:

- an editable structured retrieval specification;
- chart-type tree filtering for narrowing the candidate gallery;
- manual image selection before final generation;
- persistent reference images carried forward across later conversation turns;
- a history of generated SVG versions extracted from assistant messages.

These choices match the interface claims in Section 5: retrieval is exposed as an inspectable intermediate state rather than a hidden process, and committed exemplars persist during downstream adaptation.

4.2 SVG Compression, Placeholders, and On-Demand Inspection

The adaptation pipeline uses three mechanisms to keep SVG handling tractable.

1. **Compact structural summaries.** Reference SVGs are transformed into compressed tree-like summaries with stable node identifiers.
2. **Image payload placeholders.** Embedded base64 image payloads are replaced by placeholders of the form `[IMAGE_DATA_N]` to reduce prompt size.
3. **On-demand expansion.** When needed, the system can expand a selected compressed subtree and return its detailed SVG content for inspection.

This is the concrete mechanism behind the "compact structural summary" and on-demand sanitized SVG inspection described in Section 5 of the main paper.

Figure 1: Card-based overview of the chart-type taxonomy used in the interface. Each card corresponds to one coarse root type, followed by its grouping nodes and fine-grained chart types. Standalone roots such as *Histogram*, *Range Chart*, *Funnel Chart*, and *Pyramid Chart* terminate directly without additional grouping nodes.

4.3 Detailed SVG Handling Mechanism

The SVG handling pipeline operates in three stages.

Stage 1: Preprocess embedded image payloads. Before structural summarization, the system scans the SVG for long embedded image payloads carried by `href="data:image/..."` fields. Each unique payload is replaced by a placeholder of the form `[IMAGE_DATA_N]`. If the same image payload appears multiple times in the SVG, the same placeholder is reused. This prevents large base64 strings from dominating the prompt while preserving a reversible mapping for later restoration.

Stage 2: Build a compressed structure tree. The SVG is parsed into a recursive tree in which each node stores a unique node identifier, tag name, optional `id`/`class`, text content when present, and whether the node has children. For every node, the system also stores the corresponding SVG subtree so that the node can later be expanded exactly.

To further reduce prompt length, the compressor detects repeated sibling patterns. When at least three siblings share the same structural signature, the group is summarized as a repeated pattern rather than being fully expanded. In that case, the summary keeps:

- the total repeat count;
- the first repeated element as a template instance;
- several "important examples" selected from the repeated set, including first/last elements and salient value examples when textual numeric values are available.

The resulting text summary begins with the root SVG attributes and then prints a compressed tree. Each summarized or leaf node is annotated with a `[SHOW:node_id]` marker, which acts as an expansion handle.

Stage 3: Expand nodes on demand and restore payloads. When the model needs more detail about a region of the SVG, the system can return the exact stored SVG subtree for the selected node. Because image payloads have already been replaced by placeholders, this detailed view remains substantially shorter than the original raw SVG. After downstream editing is complete, the placeholder-to-image mapping can be used to restore the original embedded payloads into the final SVG.

Global node and placeholder consistency. When multiple reference SVGs are used in the same interaction, the system renumbers node identifiers globally and continues placeholder numbering across SVGs. This avoids collisions between different references and ensures that node references and image placeholders remain unique throughout the session.

5 Human-Written Query Collection Protocol

5.1 Target Sampling

We randomly sampled 300 target infographics from the fixed evaluation corpus $\mathcal{X}$. For each target infographic, annotators wrote two free-form text queries:

- a **short query**, intended to capture only the most salient cues needed to identify the target exemplar;
- a **long query**, intended to provide a fuller design-intent specification, including additional layout, style, or illustration cues when appropriate.

5.2 Draft Annotation Prompt and Instructions

Draft Annotation Prompt and Instructions

Goal

- You will be shown one target infographic.
- Write two retrieval queries for this infographic: one short query and one long query.

Short query

- Write a brief query containing only the most salient cues needed to identify the target infographic.

Long query

- Write a more complete query that includes additional cues such as layout, style, or illustration when these help specify the same target more precisely.

Instructions

- Describe the infographic in your own words.
- Avoid copying visible on-image text verbatim whenever possible.
- If textual content is important, paraphrase rather than quote.
- Make each query self-contained, as if it were written by a user searching for a similar infographic.
- Focus on retrievable cues such as content, chart type, layout, illustration, and visual style.

Output

- **Short query:** one sentence or short phrase.
- **Long query:** a fuller natural-language description of the intended infographic.

This draft is included to satisfy the supplementary description in the main paper and should be replaced by the original annotation form if available.

5.3 Intended Difference Between Short and Long Queries

The short/long pairing is designed to separate two retrieval regimes.

- **Short queries** test whether the retriever can recover the target from sparse but high-salience intent cues.
- **Long queries** test whether the retriever can leverage richer multi-facet descriptions without collapsing them into a single generic similarity signal.

6 Human Judgment Rubric for Retrieved Results

For the human evaluation of retrieved top-5 lists, the paper reports a paired rating study using two raters per query–method pair. The rating rule described in Section 6 can be summarized as follows.

- Each returned top-5 list is rated on a 1–5 Likert scale for overall intent match and usefulness as an exemplar.
- A score of 1 indicates an irrelevant result list.
- A score of 5 indicates a near-perfect match.
- To anchor the top end of the scale, raters are instructed to assign a score of 5 whenever the intended exemplar already appears at rank 1.

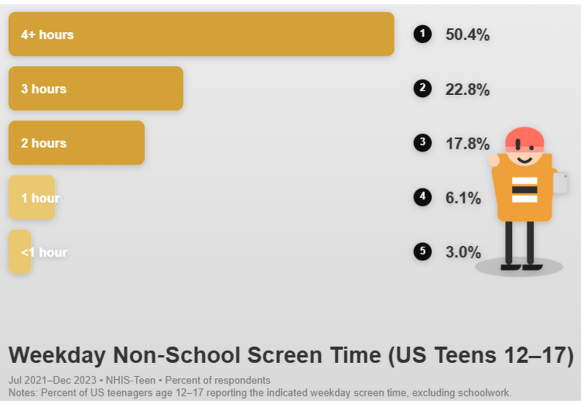
The main paper reports the average of the two raters' scores for each query–method pair, together with win/tie/loss comparisons and inter-rater agreement statistics.

7 Selected Authoring System Cases

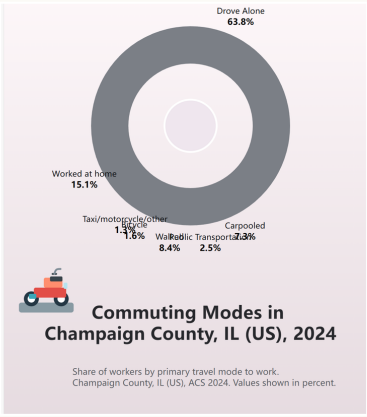
We show four selected case pairs from the blind artifact review dataset used in the authoring-system evaluation. Each row corresponds to one authoring task. Within each row, the left artifact was produced with Ours and the right artifact was produced with the Baseline system.

Case 1

Ours

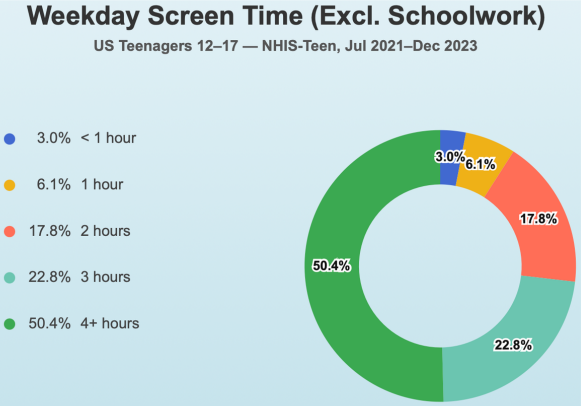


Baseline system

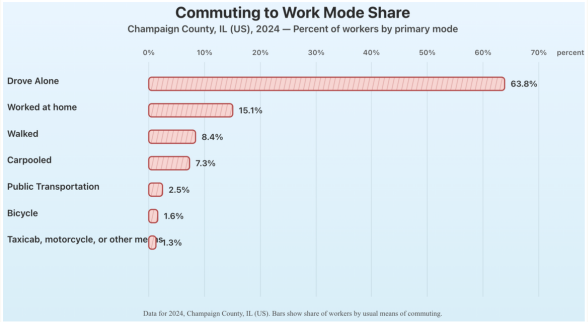


Case 2

Ours

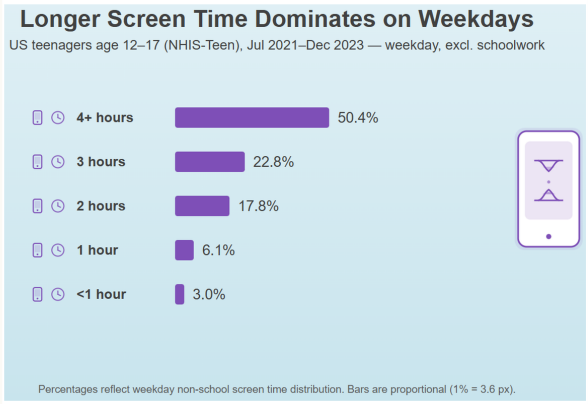


Baseline system

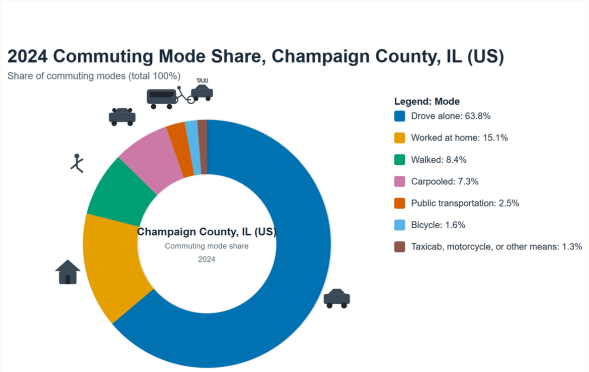


Case 3

Ours

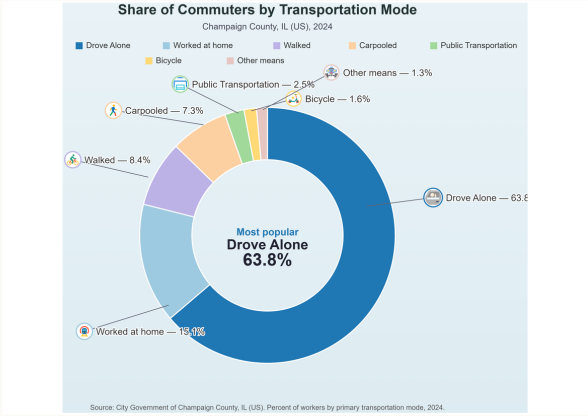


Baseline system



Case 4

Ours



Baseline system