3D Layered DAG Visualization with Time-Sliced Text/Word Analysis

Brian Shu, Erhan Huang, Gian-Carlo Kekoa Panoy, Xingyao He

February 2, 2026

1 Introduction

This project builds an interactive 3D visualization tool for hierarchical layered directed acyclic graphs (DAGs). Nodes are arranged into ordered layers (Layer 1 $\rightarrow$ Layer 2 $\rightarrow \cdots \rightarrow$ Layer N), and edges only point forward to later layers. The layout uses a force/spring approach to improve readability: nodes within the same layer repel each other to reduce overlap, while edges act like springs that pull connected nodes into visually coherent paths across layers.

The primary deliverable is a working tool that accepts reasonable input data for a layered DAG and produces a stable, explorable 3D interactive visualization (rotate/zoom/pan, select nodes, and drag nodes while the system re-stabilizes).

A target application is time-sliced text analysis: a 3D "word cloud through time." Each time slice (e.g., month-by-month corpora) forms a layer; nodes represent words with size proportional to word frequency in that slice. Connecting the same word across layers visualizes persistence, emergence, and disappearance; optional richer edges (e.g., co-occurrence/association) can be added when such relationship data is available (including exports from tools such as NVivo).

2 Scope and Scale

2.1 Problem Scope

- interactive 3D visualization of layered DAGs using constrained layered positioning plus force-directed refinement (repulsion within layers, spring forces along edges, damping/settling).
- ability to construct layered DAGs representing time-sliced word frequencies and connect word instances across time slices.
- camera controls (rotate/pan/zoom), node selection (inspect label/metadata), and node dragging with continuous physics-based stabilization.

2.2 Graph Model

- **Graph:** a directed acyclic graph with an explicit layer assignment for each node (integer time/layer index).
- **Layer constraint:** each layer corresponds to a plane in 3D space; layers are stacked along one axis (e.g., z).
- **Edges:** only connect from a node in layer i to a node in layer j where $j > i$ (often $j = i + 1$ but not required).
- **Node attributes (optional):** label, weight (e.g., frequency), category/group, and arbitrary metadata for inspection.

2.3 Layout and Rendering Approach

- **Within-layer repulsion:** nodes in the same layer repel to reduce overlap and distribute labels.
- **Spring edges across layers:** edges pull connected nodes toward alignment, encouraging straighter, visually salient paths.
- **Layer anchoring:** node positions are constrained/attracted to their layer plane to preserve the hierarchical ordering.
- **Stability controls:** tunable parameters such as force strengths and damping to reach stable, interpretable layouts.

3 MoSCoW Prioritization

3.1 Must-have

1. **Intended users**
   - Users needing exploratory understanding of layered dependencies and paths (e.g., researchers, analysts, engineers), without requiring polished production UX.
2. **Data inputs**
   - A layered DAG input format (e.g., JSON/CSV) specifying nodes, layer indices, and directed edges that only go forward in layer order.
3. **Core outputs**
   - Interactive 3D visualization with visible layers (stacked planes), nodes, and edges.
   - Layout produced by force/spring method: within-layer repulsion + edge springs + layer constraints.
4. **Basic interaction**
   - Load graph data.
   - Camera controls: rotate, pan, zoom.
   - Select a node to view its label and key metadata.
   - Drag a node and observe re-stabilization (physics continues to converge).
5. **Tunable forces**
   - Provide parameter controls (even if simple sliders or config values) for repulsion strength, spring strength, damping, and layer spacing/anchoring.

3.2 Should-have

1. **Time-sliced text to layered DAG**
   - Build a "word through time" dataset: each time slice is a layer; nodes are words with weights equal to frequency per slice.
   - Connect identical words across adjacent (or forward) time slices to form word "threads" showing persistence and change.
2. **Polished UI and higher-end rendering**
   - Improved controls (presets, guided tours, better labeling/occlusion handling), smoother performance, and enhanced rendering styles.
3. **Usefulness requirements**
   - Enable quick identification of continuity, emergence, splitting/merging patterns, and fading of terms across time layers.

3.3 Nice-to-have

1. **Integration with qualitative analysis tools**
    - Import/export compatibility with tools such as NVivo via export files and/or API (where feasible), enabling a workflow of: categorize texts by time → compute word frequency/relationships → visualize in 3D.
2. **Automated ingestion**
    - Optional pipelines for text ingestion/crawling and automatic time slicing, if time permits.

3.4 Won't-have

- Full game-engine-level visual polish, complex animation systems, or production-grade AR/VR delivery.
- Fully automated end-to-end crawling, cleaning, topic modeling, and qualitative coding as a complete product.
- Support for arbitrary cyclic graphs (the layout and semantics assume a layered DAG).

3.5 Can't-have

- Guaranteed perfect readability for very dense graphs without filtering or aggregation (visual clutter is unavoidable beyond certain scales).
- Real-time performance for extremely large graphs (e.g., hundreds of thousands of nodes) without specialized GPU/level-of-detail engineering.

4 SMART GOALS

Goal 1: Basic Prototype

Specific: Design a basic directed hierarchical graph to visualize word frequency change over time.

Measurable: A working prototype that displays nodes, directed edges, and at least two time layers.

Achievable: Focus only on core visualization features in the early stage.

Relevant: Prioritizes building a simple prototype rather than a full system.

Time-bound: Within 3 weeks.

Goal 2: Visualization of Change

Specific: Design and implement a visualization mechanism that clearly represents word frequency changes across discrete time layers within the DAG structure.

Measurable: The system visually distinguishes word frequency between time layers using node attributes such as size, color, or position.

Achievable: The goal builds on the existing DAG prototype and uses basic frequency data without requiring advanced linguistic analysis.

Relevant: This scope is realistic for the project timeline and aligns with the focus on qualitative analysis rather than full semantic modeling.

Time-bound: Within 5 weeks.

Goal 3: Interaction and Navigation

Specific: Integrate intuitive user interaction and navigation features into the DAG visualization to support exploratory analysis of word frequency changes over time.

Measurable: Enable users to zoom, rotate, select nodes, and switch between time layers to explore and interpret the data.

Achievable: Build on the existing visualization prototype by adding standard interaction techniques, without relying on complex or specialized UI frameworks.

Relevant: Fits the project timeline and enhances usability and interpretability while keeping the analytical scope focused and manageable.

Time-bound: Within 9 weeks.