NERVIS: An Interactive System for Graph-Based Exploration and Editing of Named Entities

Uroš Šmajdek[†], Ciril Bohak*,[†]

University of Ljubljana, Faculty of Computer and Information Science, Večna pot 113, SI-1000 Ljubljana, Slovenia

Abstract

We present an interactive visualization system for exploring named entities and their relationships across document collections. The system is designed around a graph-based representation that integrates three types of nodes: documents, entity mentions, and entities. Connections capture two key relationship types: (i) identical entities across contexts, and (ii) co-locations of mentions within documents. Multiple coordinated views enable users to examine entity occurrences, discover clusters of related mentions, and explore higher-level entity group relationships. To support flexible and iterative exploration, the interface offers fuzzy views with approximate connections, as well as tools for interactively editing the graph by adding or removing links, entities, and mentions, as well as editing entity terms. Additional interaction features include filtering, mini-map navigation, and export options to JSON or image formats for downstream analysis and reporting. This approach contributes to human-centered exploration of entity-rich text data by combining graph visualization, interactive refinement, and adaptable perspectives on relationships.

Keywords

Named Entity Visualization, Graph Exploration, Interactive Visualization

1. Introduction

Textual data continues to grow at an unprecedented scale across domains such as scientific publishing, journalism, and social media, but also with the digitization of historical text corpora. Extracting meaning from these large corpora requires more than keyword search or frequency analysis; it requires understanding the entities mentioned in the text and the relationships between them. named entity recognition (NER) and related information extraction methods provide a way to identify people, organizations, locations, and other key entities. However, the resulting collections of mentions and links are often complex, redundant, and difficult to navigate without appropriate visualization and interaction techniques.

Graph-based visualization offers a natural way to represent entities and their relationships. Nodes can represent documents, entity mentions, or consolidated entities, while edges capture relationships such as co-location of mentions or equivalence of entities across contexts. Yet, static graph layouts alone are insufficient for the exploration of large and ambiguous entity networks. What is needed are interactive approaches that allow users to flexibly explore different levels of granularity, filter and refine connections, and iteratively adjust the structure of the graph to reflect domain knowledge and emerging insights.

In this paper, we introduce an interactive visualization system for entity-centric text exploration. The system combines graph representations with multiple coordinated views, supporting fuzzy and precise connections, user-driven graph editing, and export capabilities for integration with downstream analysis pipelines. By emphasizing exploratory flexibility and human-centered control, our approach bridges automatic information extraction with interactive sensemaking, enabling users to uncover patterns and relationships in complex entity-rich text collections.

2. Related Work

A number of prior works provide foundations for designing interactive visualization systems that support exploration of large and complex text collections. Paul et al. [1] introduce TexTonic, a visual analytics system that enables exploration through hierarchical clustering and direct manipulation of terms. Their approach highlights how interactive interfaces can support users in dynamically engaging with data to discover patterns and relationships among entities and mentions. In a complementary direction, Lee et al. [2] expand design considerations for information visualization interactions beyond the traditional mouse and keyboard. Their work underscores the importance of multimodal interaction, which can broaden the accessibility and flexibility of visualization systems.

Scalability and performance are equally critical aspects of interactive visualization. Tao et al. [3] present Kyrix, a system designed for large-scale web-based visualization, demonstrating techniques for efficient pan and zoom interactions at scale. Their emphasis on performance optimization ensures responsiveness when managing very large datasets, an aspect directly relevant to maintaining smooth exploration of entity-rich corpora. Such techniques align well with our use of fuzzy views and approximate connections to facilitate exploratory analysis without sacrificing interactivity.

The need for robust navigation and filtering has also been well established. Al Nasar et al. [4] discuss principles of interactive personal information management systems, emphasizing personalized interfaces to help users manage large information collections. While their focus is on photographs and videos, the principles are transferable to text-based entity exploration. Similarly, Sedig and Parsons [5] propose a pattern-based framework for interaction design that supports complex cognitive activities, providing conceptual guidance for structuring interactions within entity exploration systems.

Editing and refinement capabilities form another important dimension of related work. Satyanarayan et al. [6] propose Reactive Vega, a declarative framework for specifying interactive visualizations. Their approach demonstrates how flexible editing, specification, and interaction can empower users to tailor visualizations to their analytical needs. This perspective informs our focus on enabling users to add, remove, and refine entities and connections within the graph representation.

Recent work has advanced domain-specific analytical systems enhanced with graph-based visualizations. Rahman et al. [7] present DiaVis, a dashboard for iterative exploration of diabetes-related data, highlighting the benefits of refinement cycles for sustaining engagement—an insight we adopt for entity-level analysis. Heberle et al. [8] propose a web-based system for biological networks that employs rule-based filtering and automated layout generation to support scalable exploration. Similarly, Kwak et al. [9] develop a visual analytics framework for stroke care networks, using multiple coordinated views to capture patterns across different levels of analysis, from national to community scale. In cybersecurity, a recent system by Razbelj et al. [10] introduces a graph-based approach to visualizing honeypot-captured attacks, enabling interactive exploration of large, heterogeneous datasets to detect recurring patterns and uncover hidden connections. Collectively, these systems demonstrate how interactive visualization, filtering, and scalable layouts can support domain-specific analysis.

Finally, Mongioví and Gangemi [11] propose GRAAL, a graph-based retrieval system for collecting related passages across multiple documents. By emphasizing semantic interactions and subgraph representations, GRAAL directly addresses the challenge of identifying co-occurring entity mentions in large text collections. Their methodology provides valuable inspiration for incorporating semantic relationships into our proposed interactive visualization system.

Prior research has contributed foundational techniques in multimodal interaction, scalability, navigation, iterative exploration, and semantic graph analysis. Building on these works, our system integrates and extends these ideas into a unified framework for interactive exploration, refinement, and visualization of named entities across large text corpora.

3. Design Principles

In this section, we highlight key limitations of existing named entity visualization systems and draw from the broader visualization literature to derive five core design principles for our approach:

- **Integrated data exploration and visualization** – Graph-based visualizations of named entities often involve selecting a subset of entities and relations from a larger corpus, the scope and relevance of which may not be known in advance. This selection process can become labor-intensive if performed manually. Existing network visualization systems often assume that all entities and relations in a dataset are to be included in the visualization, overlooking the exploratory step required to identify subsets of interest. This can result in cluttered or unreadable graphs, as well as delays in achieving meaningful insights.

- **Cross-Document Entity Relations** – Beyond the classification of individual mentions, named entity linking (NEL) and coreference extraction are often employed to link multiple entity instances that refer to the same underlying entity. This task extends both within a document, where entities may appear under different surface forms, abbreviations, or titles, and across documents, where mentions must be linked despite variation in spelling, context, or language. Commonly used named entity visualizations, such as displaCy [12] (see fig. 1 (left), struggle to showcase such relations effectively as they are only depicted as properties of individual entity instances.

- **Retaining the Fuzziness of Entity Data** – Information extraction processes related to named entities, such as NER and NEL, rarely yield unambiguous results, especially for historical and heterogeneous corpora, where metadata is often incomplete [13]. Instead, they often produce multiple candidate entities and relations with associated confidence scores [14, 15]. Instead of only preserving the "best guess", we aim to showcase multiple candidates and let the researcher explore the spectrum of possible interpretations.

- **Optional automation** – While automation is critical for making graph visualizations accessible, it should not eliminate authorial control. Meaningful network visualizations often require manual refinement, such as pruning peripheral nodes, re-weighting edges, or emphasizing particular subgraphs. Automation should therefore serve as an initial scaffold, enabling authors to iteratively adjust and enhance the visualization to highlight the most relevant aspects of the data.

- **Progressive visualization** – Interactive graph exploration depends heavily on timely and comprehensible visual feedback. Research indicates that user interactions should produce visual responses within 50–100 ms to maintain a fluid sense of control [16]. At the same time, sudden or large-scale layout changes can disorient users and obscure the perceived impact of their actions. Progressive visualization techniques [17] mitigate these issues by subdividing expensive computations, such as force-directed layouts or entity disambiguation steps, into incremental updates with smooth transitions [18]. This approach provides clarity and responsiveness while supporting iterative exploration of large or evolving networks.

4. System Design

NERVIS is an interactive web-based system for graph visualization and editing, implemented using HTML, CSS, TypeScript, and React framework[1]. The system is designed around a modular architecture that separates data management, visualization, and user interaction. At its core, NERVIS maintains structured data models that represent nodes, edges, and associated metadata, enabling efficient storage, retrieval, and manipulation of graph information. These models feed into a graph visualization pipeline,

---

[1]https://react.dev

which computes layouts, applies styling and interaction rules, and renders the graph for exploration and analysis.

The user interface provides direct access to the system's functionality, allowing users to inspect and modify graph elements, control visualization parameters, and apply filters. The interface is organized into distinct components that facilitate navigation, node-level editing, and rule-based filtering, providing a cohesive environment for interactive graph analysis.

4.1. Data Model

In this section, we present the data model underlying NERVIS, designed according to the principles of *Cross-Document Entity Relations* and *Retaining the Fuzziness of Entity Data* (see section 3).

Typical text-based named entity visualizations represent entities as annotated spans over a sequence of tokens (see fig. 1, left). To extend this approach to a network graph spanning multiple documents, entities are first reformulated as *entity instance nodes*, while token sequences are abstracted into *document nodes* (fig. 1, middle). Although this representation captures basic entity-document relationships, it cannot effectively represent higher-level relations, such as links between mentions of the same entity across different documents. To address this, entity instance nodes are further separated into *mention nodes*, representing individual occurrences of an entity within a document, and *entity nodes*, representing the underlying entity itself (fig. 1, right).

Both *entity* and *mention nodes* include an entity class property derived from the CoNLL-2003 dataset [19], which classifies named entities as *Person*, *Organization*, *Location*, or *Miscellaneous*. This classification was chosen because it underpins many state-of-the-art NER methods [14]. To maintain the principle of data fuzziness, mentions are not strictly constrained to match the class of their connected entity, enabling the system to highlight potential inconsistencies while allowing users to correct them at their discretion.
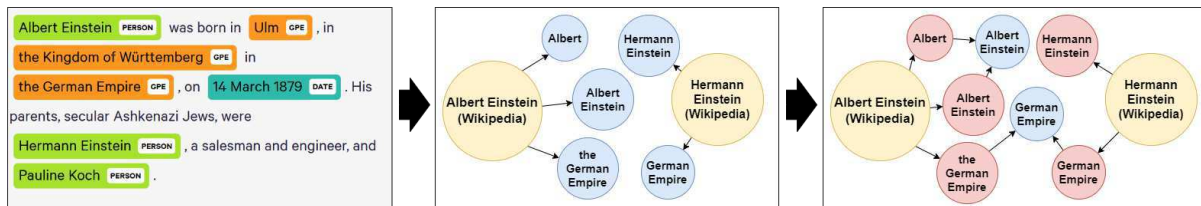

Figure 1: Data model diagrams for text-based named entity visualizations from displaCy [12] (left), simple document-entity model (middle), and proposed document-mention-entity model (right).

The data model also defines four types of connections i.e., graph edges. First, *mention-to-document* and *mention-to-entity* connections reflect relationships provided in the input data, and as per design principles, impose no limit on the number of entities a mention may be associated with. Second, *mention-collocation* connections capture which mentions occur within the same sentence. Finally, a virtual *entity-to-document* connection is introduced to facilitate observation of cross-document entity relations and to enable more efficient filtering and exploration of the graph.

4.2. Visualization Pipeline

The NERVIS visualization pipeline is illustrated in fig. 2 and is modeled after the classical visualization pipeline framework [20]. It is designed to transform the structured data described in section 4.1 into an interactive graph representation, and is organized into five main stages:

1. **View Filtering** – The structural abstraction of the graph is adjusted to emphasize different levels of detail in the visualization.
2. **Data Filtering** – Input data is refined according to user-specified criteria, such as rule-based filters, user selection, or document subsets, to focus the visualization on relevant nodes and edges.

3. **Data Mapping** – Filtered data is mapped onto geometric structures, with selected elements optionally receiving additional visual emphasis based on user interactions.
4. **Layout Creation** – Node and edge positions are computed to produce a spatial arrangement that highlights relational structures and preserves graph readability.
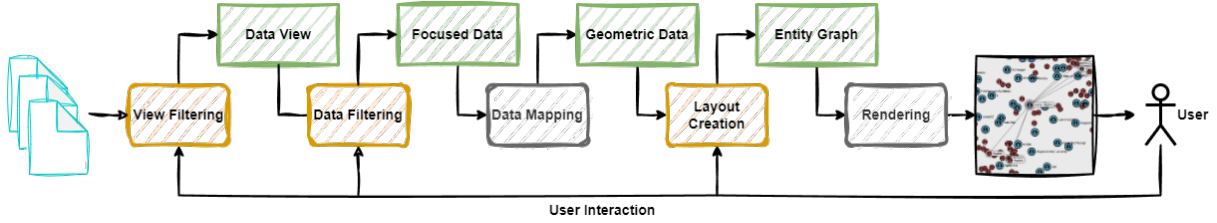5. **Rendering** - The graph is visually rendered on the display.

Figure 2: NERVIS visualization pipeline. Orange and gey nodes indicate intractable and uninterpretable stages respectively. Green nodes indicate intermediary data representations.

4.2.1. View Filtering

The first stage of the pipeline allows users to toggle between two structural views of the graph: the default Document–Mention–Entity (D–M–E) view and a simplified Document–Entity (D–E) view. In the D–E view, mention nodes are abstracted away and replaced with virtual entity-to-document edges, producing a more concise visualization that emphasizes cross-document entity relations.

This choice directly shapes the set of nodes and edges made available to subsequent stages of the pipeline. By selecting the D–M–E view, users retain the full detail of entity mentions within documents, enabling fine-grained exploration of textual annotations. By contrast, the D–E view allows users to focus on higher-level relationships across documents without the visual clutter of mention nodes.
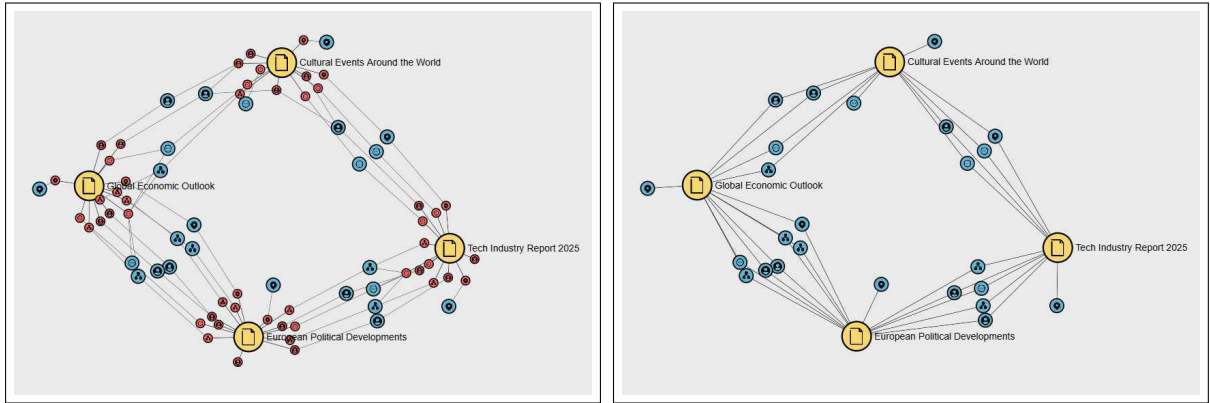

Figure 3: A comparison of Document–Mention–Entity view (left) and Document–Entity view (right).

4.2.2. Data Filtering

To satisfy the *Integrated Data Exploration and Visualization* design principle (see section 3), NERVIS supports interactive filtering of the input data. Inspired by other network visualization tools (*e.g.,* PatientFlow [9] and CellNetVis [8]), we implement two filtering steps:

1. **Focus filter** – This filter allows the user to specify a node or edge of interest. Two focus filters can be applied concurrently: the *selection filter* and the *node focus filter*. The *selection filter* highlights a chosen node by removing all edges that do not connect it to its neighbors, reducing ambiguity in dense graphs. The *node focus filter* removes all nodes and edges not directly connected to a

specified node, enabling users to concentrate on the local neighborhood of interest. If applied concurrently, they are designed not to conflict with each other; the selection filter will not remove edges of the focused node and *vice versa*.

2. **Rule filter** – This filter allows users to control which types of nodes and edges are displayed based on their properties. Filtering can be performed by node type (e.g., *document*, *mention*, *entity*), by entity class (e.g., *Person*, *Location*), or by any combination of these criteria.

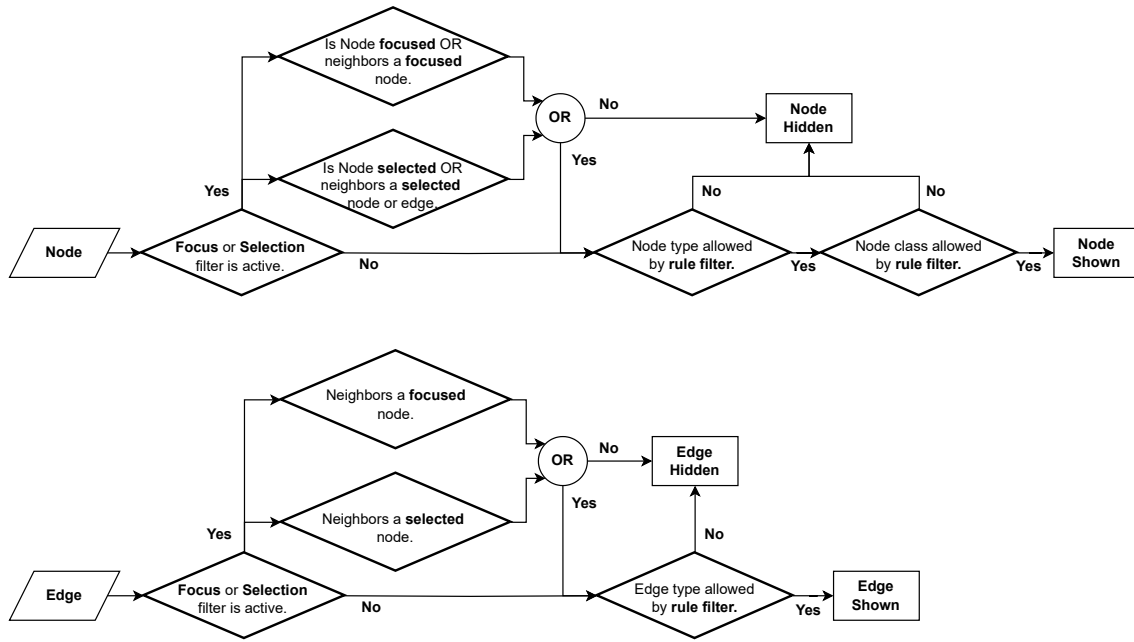The complete filtering algorithm is illustrated in fig. 4.

Figure 4: Node (top) and edge (bottom) filtering algorithm flowchart.

4.2.3. Data Mapping

In the data mapping stage, the filtered data is transformed into geometric structures that will be rendered in the final visualization. This process is subject to two key constraints. First, the number of nodes can reach several thousand, requiring the use of simple and recognizable shapes to maintain clarity in high-density graphs. Second, the layout algorithms, described in section 4.2.4, are similarly limited to simple geometric structures to ensure computational efficiency and maintain performance.

To satisfy these constraints, each node is represented as a circle, with size and color encoding its type. To support a variety of use cases, two color schemes are provided: one based on node type and another based on entity class. Additionally, to enhance visual distinction and improve user interpretation, a pictogram indicating the node's class and type is displayed. Examples of the resulting node structures are shown in fig. 5.

4.2.4. Layout Creation

In the layout creation step, the positions of nodes within the graph are determined. In line with the design principle of *optional automation* (see section 3), NERVIS allows users to either employ an automated layout algorithm or manually adjust node positions. For automated layout generation, we use the *ForceAtlas2* algorithm [21], which is efficient and supports iterative computation, aligning with our design principle of *progressive visualization*.
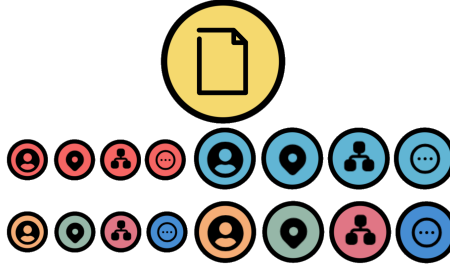
Figure 5: Visual structures used to depict different node types. The top row shows a *document* node. The middle row illustrates *mention* nodes (left) and *entity* nodes (right) across different entity classes. The bottom row depicts mention and entity nodes using an entity class-based color scheme. From left to right the individual node pictograms depict *person*, *location*, *organization* and *miscellaneous* entity classes.

We employ the implementation provided by the Graphology framework [22], which leverages the Web Workers API[2] to multithread the computation, ensuring that the user interface remains responsive during layout processing. An example of a layout produced by the ForceAtlas2 algorithm is shown in fig. 6.

4.2.5. Rendering

Rendering constitutes the final stage of the visualization pipeline, responsible for producing the visual representation of the graph on the user interface. To ensure a smooth and responsive experience when visualizing high-density graphs, NERVIS integrates the *Sigma.js*[3] framework. Sigma.js employs an instance-based WebGL rendering pipeline, allowing computations to be offloaded to the GPU and fully leveraging modern graphics hardware for high-performance visualization.

One limitation of this rasterized rendering approach is that it does not produce vector-based outputs, which restricts the possibility of external post-processing or vector editing of the visualization. However, this is compensated by a comprehensive set of in-system editing tools, including node manipulation, focus and rule filtering, and layout adjustments, which allow users to modify the graph without requiring external editing.

4.3. User Interface

The NERVIS system provides a user interface for interactive graph exploration and editing (see fig. 6). The main display (1) presents the graph produced by the visualization pipeline described in section 4.2, enabling detailed inspection of nodes and edges. Additionally, the event-based interaction system allows direct selection, movement, and deletion of nodes and edges on the main display. The node editing widget (2) allows direct modification of node attributes and properties within the interface, and also supports immediate update of the visualized nodes.

The toolbar (3) offers core workflow functions, including graph import and export, view toggling, and activation of focus filters to highlight or hide specific portions of the graph. The toolbar also allows for efficient search and selection of nodes within the graph, utilizing MiniSearch[4], an efficient full-text search engine for JavaScript. A minimap (4) provides an overview of the graph layout, assisting in navigation and orientation on larger graphs. The rule filter panel (5) permits selective filtering according to user-defined criteria, allowing attention to be directed toward relevant patterns and relationships. The node actions panel (6) allows access to three context-sensitive operations: contextual zoom, which sets the view to encompass the node and its visible neighbors, the activation of the node focus filter (see section 4.2.2), and node deletion.

---

[2]https://html.spec.whatwg.org/multipage/workers.html
[3]https://github.com/jacomyal/sigma.js
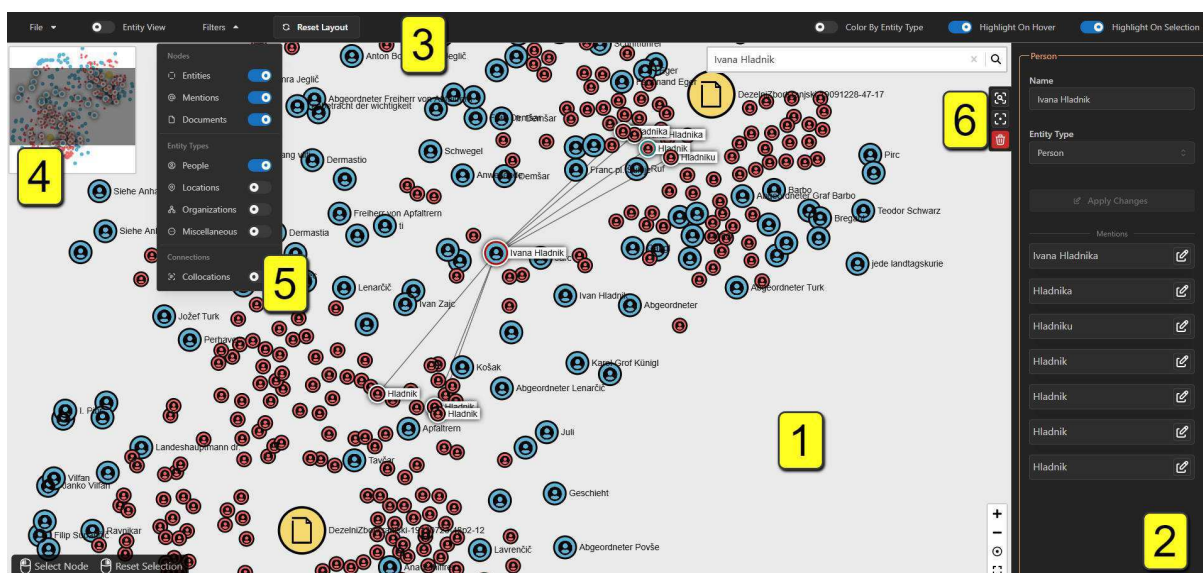[4]https://github.com/lucaong/minisearch

Figure 6: Overview of NERVIS graphical user interface. (1) Graph rendered using the visualization pipeline described in section 4.2. (2) Node editing widget (3) Toolbar. (4) Graph's minimap. (5) Selection of rule filters. (6) Node-specific actions and filters.

5. Expert Evaluation

Because NERVIS builds on established technologies, such as the Sigma.js rendering framework for GPU-accelerated graph drawing, a dedicated performance evaluation was not conducted. Instead, we carried out an expert evaluation to assess the system as a whole, focusing on usability, interaction design, and the extent to which the implemented features support the intended workflows.

We discussed the resulting system with a Digital Humanities (DH) scholar studying historical print media, investigating relationships and representations of historical figures and geographic locations. Since systems for NER and NEL perform worse than humans, historians have learned to be very skeptical of purely computer-based approaches. In order to make historical research reliable, software for interacting with computer-produced results is a critical part of the DH research process.

After initial consultation set the stage and expectations, we performed two rounds of expert system evaluation with synthetic data. After preliminary evaluation, the feature requests were prioritized and the most important ones were implemented. After the final round, the expert judged the system to be mature enough for use in real-life setting. They also composed the list of possible future improvements, additions and extensions.

In the initial consultation we discussed software goals and previous experience when handling named-entity data with existing software. One of the key things pointed out during this consultation was the burden of human labor when dealing with computer-generated labels. For historians, a recall-focused approach is preferred to an accuracy-focused one. They prefer broad sets of result candidates, which they can browse through and edit manually as opposed to only seeing a single most likely result. This encouraged the design choice to develop a tool where the deletion of nodes and edges is the most common user operation, as opposed to creating new ones. A high recall approach inevitably generates relatively high amounts of noise, which we planned to tackle with a variety of content filters that allow researchers to focus on specific documents, entities, or types when cleaning the data. Since the users of DH tools often do not have an engineering background and often struggle with the installation process, we decided on the web platform, which also makes the system broadly available.

After the prototype was tested, we followed up with our second consultation. At this stage, the tool received generally positive feedback, with feature requests focusing on the most common user actions. As already mentioned, with high recall results, most user actions involve deleting nodes and edges, which was made additionally accessible with the implementation of hotkeys for quick node/edge

deletion. We discussed the noisiness of the visualization when there are many documents present, which exposed the need for users to focus on a single document when cleaning the data. While collocations are a common tool for studying text, in this particular task the question of which entities appear in the same sentence is of relatively limited importance. We decided to prioritize edges between documents, mentions, and entities while initially hiding the edges denoting collocations.

The final evaluation presented us with a new list of requirements and feature requests, which are the subject of future work. The need for undo/redo actions and action history was expressed alongside the wish to see specific mentions in the text context. Finally, the need to merge recognized entity nodes arose, due to errors in NEL process.

6. Conclusion

In this paper, we introduced an interactive web-based system for the visualization and editing of named entity graphs. The system builds on existing technologies such as Graphology [22] and Sigma.js, but contributes a tailored data model, a structured visualization pipeline, and interaction techniques specifically designed for large-scale named entity graphs.

Because the system leverages established rendering and layout technologies, we concentrated our evaluation on usability and interaction rather than raw performance. To this end, we conducted an expert evaluation with a digital humanities scholar working on historical print media. The evaluation provided valuable feedback on design decisions, particularly regarding the trade-off between recall and noise in named-entity data, the need for efficient editing operations, and the importance of content filters to reduce cognitive load when exploring large graphs. Iterative consultation shaped the prototype into a system the expert deemed ready for real-world use, while also identifying directions for further development, such as undo/redo functionality, contextual access to mentions, and support for merging entity nodes.

References

[1] C. L. Paul, J. S. Chang, A. Endert, N. Cramer, D. Gillen, S. Hampton, R. Burtner, R. Perko, K. Cook, TexTonic: Interactive Visualization for Exploration and Discovery of Very Large Text Collections, Information Visualization (2018). doi:10.1177/1473871618785390.

[2] B. Lee, P. Isenberg, N. H. Riche, S. Carpendale, Beyond Mouse and Keyboard: Expanding Design Considerations for Information Visualization Interactions, IEEE Transactions on Visualization and Computer Graphics (2012). doi:10.1109/tvcg.2012.204.

[3] W. Tao, X. Liu, Y. Wang, L. Battle, c. Demiralp, R. Chang, M. Stonebraker, Kyrix: Interactive Pan/Zoom Visualizations at Scale, Computer Graphics Forum (2019). doi:10.1111/cgf.13708.

[4] M. R. Al Nasar, M. Mohd, N. M. Ali, A Conceptual Framework for an Interactive Personal Information Management System (2011). doi:10.1109/iuser.2011.6150545.

[5] K. Sedig, P. Parsons, Interaction Design for Complex Cognitive Activities With Visual Representations: A Pattern-Based Approach, AIS Transactions on Human-Computer Interaction (2013). doi:10.17705/1thci.00055.

[6] A. Satyanarayan, R. P. Russell, J. Hoffswell, J. Heer, Reactive Vega: A Streaming Dataflow Architecture for Declarative Interactive Visualization, IEEE Transactions on Visualization and Computer Graphics (2016). doi:10.1109/tvcg.2015.2467091.

[7] M. F. Rahman, M. R. Islam, S. Akter, S. Akter, L. Islam, G. Xu, DiaVis: Exploration and Analysis of Diabetes Through Visual Interactive System, Human-Centric Intelligent Systems (2021). doi:`10.2991/hcis.k.211025.001`.

[8] H. Heberle, M. F. Carazzolle, G. P. Telles, G. V. Meirelles, R. Minghim, Cellnetvis: a web tool for visualization of biological networks using force-directed layout constrained by cellular components, BMC Bioinformatics 18 (2017) 395. doi:`10.1186/s12859-017-1787-5`.

[9] K. Kwak, J. Park, H. Song, A visual analytics framework for inter-hospital transfer network of stroke patients, Applied Sciences 13 (2023). doi:`10.3390/app13095241`.

[10] M. Rabzelj, C. Bohak, L. v. Južnič, A. Kos, U. Sedlar, Cyberattack Graph Modeling for Visual Analytics, IEEE Access 11 (2023) 86910–86944. doi:`10.1109/ACCESS.2023.3304640`.

[11] M. Mongiovì, A. Gangemi, GRAAL: Graph-Based Retrieval for Collecting Related Passages Across Multiple Documents, Information (2024). doi:`10.3390/info15060318`.

[12] M. Honnibal, I. Montani, S. Van Landeghem, A. Boyd, spaCy: Industrial-strength Natural Language Processing in Python (2020). doi:`10.5281/zenodo.1212303`.

[13] M. Ehrmann, A. Hamdi, E. L. Pontes, M. Romanello, A. Doucet, Named entity recognition and classification in historical documents: A survey, ACM Comput. Surv. 56 (2023). doi:`10.1145/3604931`.

[14] Z. Hu, W. Hou, X. Liu, Deep learning for named entity recognition: a survey, Neural Computing and Applications 36 (2024) 8995–9022. doi:`10.1007/s00521-024-09646-6`.

[15] Z. Zhang, Y. Zhao, H. Gao, M. Hu, LinkNER: Linking Local Named Entity Recognition Models to Large Language Models using Uncertainty, in: Proceedings of the ACM Web Conference 2024, WWW '24, Association for Computing Machinery, New York, NY, USA, 2024, p. 4047–4058. doi:`10.1145/3589334.3645414`.

[16] Z. Liu, J. Heer, The Effects of Interactive Latency on Exploratory Visual Analysis, IEEE TVCG 20 (2014) 2122–2131. doi:`10.1109/TVCG.2014.2346452`.

[17] M. Angelini, G. Santucci, H. Schumann, H.-J. Schulz, A Review and Characterization of Progressive Visual Analytics, Informatics 5 (2018). doi:`10.3390/informatics5030031`.

[18] J. Heer, G. Robertson, Animated Transitions in Statistical Data Graphics, IEEE TVCG 13 (2007) 1240–1247. doi:`10.1109/TVCG.2007.70539`.

[19] E. F. Tjong Kim Sang, F. De Meulder, Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition, in: Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003, 2003, pp. 142–147. URL: https://www.aclweb.org/anthology/W03-0419.

[20] S. K. Card, J. D. Mackinlay, B. Shneiderman (Eds.), Readings in information visualization: using vision to think, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.

[21] M. Jacomy, T. Venturini, S. Heymann, M. Bastian, ForceAtlas2, a Continuous Graph Layout Algorithm for Handy Network Visualization Designed for the Gephi Software, PLOS ONE 9 (2014) 1–12. doi:`10.1371/journal.pone.0098679`.

[22] G. Plique, Graphology, a robust and multipurpose Graph object for JavaScript., 2025. doi:`10.5281/zenodo.14835805`.