Towards the Generation and Application of Dynamic Web-Based Visualization of UIMA-based Annotations for Big-Data Corpora with the Help of Unified Dynamic Annotation Visualizer

Thiemo Dahmann Julian Schneider Philipp Stephan
Giuseppe Abrami Alexander Mehler

Goethe University Frankfurt, Texttechnology
Robert-Mayer-Strasse 10, 60325 Frankfurt am Main

thiemo@dahmann.info, mail@julian-schneider.de, philipp.stephan@stud.uni-frankfurt.de
{abrami, mehler}@em.uni-frankfurt.de

Abstract

The automatic and manual annotation of unstructured corpora is a routine task in many scientific fields and is supported by a variety of existing software solutions. Despite this variety, few solutions currently support annotation visualization, especially for dynamic generation and interaction. To bridge this gap and visualize annotated corpora based on user-, project-, or corpus-specific aspects, we developed Unified Dynamic Annotation Visualizer (UDAV). UDAV is a web-based solution that implements features not supported by comparable tools, enabling a customizable and extensible toolbox for interacting with annotations and allowing integration into existing big-data frameworks. We exemplify UDAV through a range of visualizations and also provide an evaluation of corpus import and processing performance.

Keywords: NLP, UIMA, Annotations, dynamic visualization

1. Introduction

With the growing availability of powerful (multimodal) large language models ((M)LLMs) and systems based on them, automatically achieving optimal and efficient pre-processing of unimodal and multimodal corpora remains a challenge across disciplines (Cai et al., 2022; Virk et al., 2024; Tang et al., 2025). These challenges are addressed by frameworks, with existing solutions for one-dimensional (e.g. Kim et al. 2024; Ni et al. 2024; Sun et al. 2023) and multidimensional annotation structures (e.g. Aote and Potnurwar 2018; Chen et al. 2020; Ji et al. 2023). Although this will likely remain unfinished for some time, there are approaches to another challenge: the *dynamic visualization of and interaction with annotations*. The challenge concerns the fact that, alongside a variety of software frameworks that enable automatic analysis, only relatively static and limited visualization capabilities are available. Consequently, substantial effort is still required to visualize the results of many scientific projects and their application areas. At the same time, integrating ever new visualizations, as well as their processing and aggregation, requires considerable effort in terms of mapping, implementation, and final instantiation. Furthermore, changes in data formats across projects or pre-processing routines substantially increase implementation effort, causing delays and unnecessary complexity. Well-known corpus managers (e.g. Kilgarriff et al. 2004) provide functionality for generating frequency charts, for instance, but their flexibility in terms of modification and parameterization is limited.

Difficulties in aligning annotation formats across projects at the document and corpus level, together with limited interaction options for the resulting annotations, indicate a functional gap in extensible, schema-based annotation support. To bridge this gap and allow for feasible visualizations of schema-based annotation results via an API- and web-based solution, we developed Unified Dynamic Annotation Visualizer (UDAV). UDAV is a server-side web-based visualization platform for UIMA-based, schema-driven annotations of multimodal corpora. UIMA (*Unstructured Information Management Architecture* – Ferrucci et al. 2009) is an annotation standard that, based on extensible class structures, supports the annotation of unstructured information, whether textual or multimodal (Götz and Suhre, 2004). In contrast to formats such as CoNLL (Buchholz and Marsi, 2006), UIMA supports complex data structures via an extensible annotation schema and their serialization in databases (Fette et al., 2013; Abrami and Mehler, 2018). A wide variety of data formats (e.g., CoNLL, TCF (Heid et al., 2010)) can be represented in UIMA, and, together with the aforementioned features and its integration into NLP pipelines (e.g. Cunningham et al. 1995; Kano et al. 2009; Uryupina et al. 2016), this makes it well suited

Proceedings of the Fifteenth Language Resources and Evaluation Conference (LREC 2026), pages 6695–6705
11–16 May 2026. ©ELRA Language Resources Association (ELRA), 2026

for UDAV in the context of big data (Abrami et al., 2022). The flexible use of UDAV is enhanced by supporting both stand-alone deployment via Docker and reuse as a plugin in web-based applications, since its API enables partial or complete, modifiable sectioning, aggregation, and visualization of corpus- and document-level results. UDAV can be integrated into the web-based corpus explorer UCE (Bönisch et al., 2025) to extend its functionality (see Section 3). It is available via GitHub[1] under an AGPL license along with a detailed documentation and a live demonstration.

The paper and the software solution it describes provide three main functions:

1. generating schema-based annotation visualizations in an API- and web-based software solution;
2. enabling dynamic, modifiable, and reproducible visualizations at the corpus and document level;
3. providing a modular and expandable architecture, including encapsulation in microservices.

The paper is structured as follows: Section 2 describes features required for dynamic visualization of annotations. Section 3 overviews related work; Section 4 describes UDAV in detail. Finally, Section 5 presents our evaluation, followed by future work (Section 6) and our conclusion (Section 7).

2. Features

Ensuring dynamic, reusable, and flexible visualizations requires a range of functions and features. The more of these features a system offers, the more versatile it is. Building on prior work (Hemati et al., 2016; Bönisch et al., 2025) and the solutions reviewed in Section 3, we specify the following features for implementing dynamic visualizations:

I **Multi-corpus-based**: Across research questions and projects, corpora of different languages and feature sets require the same level of compatibility. Together with Feature VII, this includes the capability to filter and provide collections at the corpus level.

II **Modification-based**: Dynamic annotation representation requires functionality to modify visualizations, templates, and parameters. This also involves visualization layout and composition, mapping to data sources, and aggregation. This may include rules and mapping functions to generate visualization components based on the underlying annotation model (see Feature III).

III **Schema-based**: To maximize flexibility and reusability, a dynamic annotation-visualization framework requires a complex annotation schema. The schema should be easily extensible, support multimodality (Feature IV), and enable filtering (Feature VII).

IV **Extensible**: As requirements evolve, new or modified visualizations and their associated generators should be implementable, extensible, and modifiable with moderate effort (see Feature II).

V **Multimodal**: Since corpora may contain video, audio, and other data types, dynamic visualization should extend to multimodal information. This includes visualizing, presenting, and interacting with multimodal content, particularly through filtering (Feature VIII).

VI **API-based**: Beyond Feature VII, API-based visualization and processing of annotations are mandatory and underpin all other features. This includes optimized backend retrieval, aggregation, processing, and presentation based on the annotations in Feature III and the generators in Feature II, enabling accessibility and usability as specified in Feature VII.

VII **Web-based**: In distributed, web-based settings, dynamic annotation support is indispensable, enabling responsiveness and native mobile use. This feature relates to features III–V, as the chosen web-based implementations influence those functions and their accessibility.

VIII **Filter-based**: Filtering instantly modifies selected annotations based on Features II and III, enhanced by Feature VI. It is a core component of dynamic annotation visualization.

IX **AI-driven**: Integrating AI methods (e.g., RAG) enables effective generation, aggregation, evaluation, and presentation of dynamic annotations, but requires hardware resources, API functionality (Feature VI), flexible data formats (Feature III), and generators (Feature II).

3. Related Work

Many annotation tools exists for diverse application purposes. To provide an overview, the following list focuses on tools that offer graphical annotation visualization, excluding well-known solutions such as Nghiem et al. (2021); Terdalkar and Bhattacharya (2023); Kim et al. (2024). Table 1 lists the supported features of each framework.

GATE (Cunningham et al., 1995) can use UIMA and thus supports schema-based annotation (Feature III). It displays annotated documents across

[1] https://texttechnologylab.github.io/Unified-Dynamic-Annotation-Visualizer/

No.	Framework	Reference	I	II	III	IV	V	VI	VII	VIII	IX
1	GATE	Cunningham et al. (1995)									
2	Glozz	Widlöcher and Mathet (2012)									
3	RATTE	Wild and Pissarek (2022)									
4	TAG	Forbes et al. (2018)									
5	TextImager	Hemati et al. (2016)									
6	Voyant Tools	Sinclair and Rockwell (2016)									
7	UCE	Bönisch et al. (2025)									
8	UDAV										
			2	1	5	1	0	4	6	3	0
			3	4	2	3	1	1	1	1	1
			3	3	0	4	7	3	1	2	7
			0	0	1	0	0	0	0	1	0

Table 1: Comparing tools using the features of Sec. 2: met, partially met, not met, unknown.

corpora but does not provide corpus-specific annotation representations (partially Feature I). GATE's web interface lacks visualization, so Features VII and VIII are not supported. Runtime modification (Feature II) is not supported, and extensibility (Feature IV) requires modifying the program, so neither feature is available. GATE supports neither multimodality (Feature V) nor AI (Feature IX).

Glozz (Widlöcher and Mathet, 2012) is an annotation tool for creating and visualizing a range of linguistic annotations: Since it follows the *Unit Relation Scheme* (Widlöcher, 2008), it partially meets Feature III and, via its filter function, partially meets Feature VIII; however, as a desktop application it does not support Features VI and VII, extensions require program changes (Features II and IV), and multimodality (Feature V) and multi-corpus support (Feature I) are unavailable.

RATTE (Wild and Pissarek, 2022) generates only a limited set of linguistic features but offers straightforward, effective annotation visualizations. Due to limited documentation, this web-based tool (Feature VII) can only be partially classified in our scheme, though this could be addressed through direct exploration. As a result, some features are unavailable, and for some criteria we cannot provide a definitive assessment.

Text Annotation Graphs (TAG; Forbes et al. (2018)) supports web-based text annotation and graph-based visualization (Feature VII), but offers only minimal modification (Feature II) and does not meet the other features.

In contrast to the previously mentioned tools, TextImager (Hemati et al., 2016) offers visualizations based on schema-driven preprocessing (Feature III) and can display different corpora (Feature I), but corpus differentiation is not parameterized. TextImager provides an API (Feature VI), is web-based (Feature VII), and supports annotation filtering (Feature VIII). Extensibility (Features II and IV) is limited, since changes require updating both the visualizations and the API, which also constrains Feature VIII. Multimodality (V) and AI utilization (IX) are also not supported.

Another framework that supports a wide range of visualizations and imports various input formats is *Voyant Tools* (Sinclair and Rockwell, 2016). It supports multiple corpora (Feature I) and, as a web-based tool, supports Features VII–IX. It can also be used with existing visualizations (Feature II). However, it supports neither multimodality (Feature V) nor AI (Feature IX), and extensibility (Feature IV) requires adapting the implementation.

Last but not least, UCE provides fewer visualizations than the aforementioned frameworks, but offers much stronger modification capabilities (Features II and IV) and supports AI integration (Feature IX) via RAG bots (Gao et al., 2024). Like TextImager, UCE is UIMA-based (Feature III) and supports web-based exploration of multiple corpora (partially Feature I) in different presentation modes (Feature VII). Due to its robust schema (Feature III) and flexible API (Feature VI), it also supports filtering (Feature VIII), though this may require minor source-code adaptations. UCE does not currently support multimodal use (Feature V), but this appears achievable given its existing features.

The literature shows that most tools visualize annotations only as a practical by-product, and typically in a static way. There are exceptions (e.g., Voyant Tools), but follow-up use is still not widespread. To combine these features and integrate a schema supported by existing big-data frameworks, we describe UDAV in the next section.

4. Unified Dynamic Annotation Visualizer

UDAV allows for displaying processing results in a schema-based (Feature III), reproducible, dynamic, and interactive way without hard-coding project-specific visualizations. UDAV is a web-

based framework (Feature VII) that integrates into existing systems (e.g., UCE) and supports multiple corpora in different languages (Feature I). It uses the big-data framework DUUI (Leonhardt et al., 2023; Abrami et al., 2025) to import annotations and enable distributed, efficient processing of annotated documents. Through its embedded architecture (see Figure 1) and customizable GENERATORS (Layer B), it is highly configurable and supports multi-level filtering (Feature VIII) via its schema-based (Feature III) API (Feature VI). UDAV implements interactive components such as text highlighting, networks, and charts, which users can arrange in a dashboard-like grid and integrate from visualization repositories (e.g. d3.js[2], sigmajs[3]), if configured (IV). Developed with Java, Spring[4] and *Apache FreeMarker*[5], it reduces the time and effort required to create reusable visualizations for multi-level annotation data while preserving user control over data representation. As it allows for simultaneously addressing data from different corpora, it is also multi-corpus capable (Feature I).

4.1. Architecture

UDAV is a modular Java Spring application for dynamic visualization of UIMA-based annotations. Its architecture is multilayered, with modules for the API, data import, generation, and visualization. Each component is separated to improve maintainability, scalability, and extensibility. Maven (Miller et al., 2010) manages dependencies and builds, ensuring consistent, reproducible configurations across environments. UDAV can also be built as a Docker image, with settings controlled via environment variables.

This allows the system to be integrated into various environments. The following layers are important for understanding UDAV and are illustrated in Figure 1:

A **API Layer**: The API follows best practices by separating *controllers*, *services*, and *handlers* (Shikder, 2024; Skills, 2024) and supports Feature VI.

- **Controller**s (⇄) are addressed by the frontend; they provide visualization data for presentation on the client system or pre-rendered UI components using FreeMarker templates.
- **Service**s (⚙) encapsulate the logic modules and transform and aggregate data according to the specified requirements (Section 4.2) for visualization in the frontend or permanent storage.
- **Handler**s (⚙⚙) abstract database interactions and provide an interface for querying and storing annotations, enabling a loose coupling between logic and data storage.

The *DataService* delegates the provision of data for visualizations to a series of chart handlers, each of which implements a common *ChartHandler* interface. This allows developers to extend diagram types by adding a single additional handler that implements this interface.

B **Importer Layer**: This layer imports annotated UIMA documents into the database. As shown in Figure 2, UDAV creates one internal table per annotation class and serializes the source

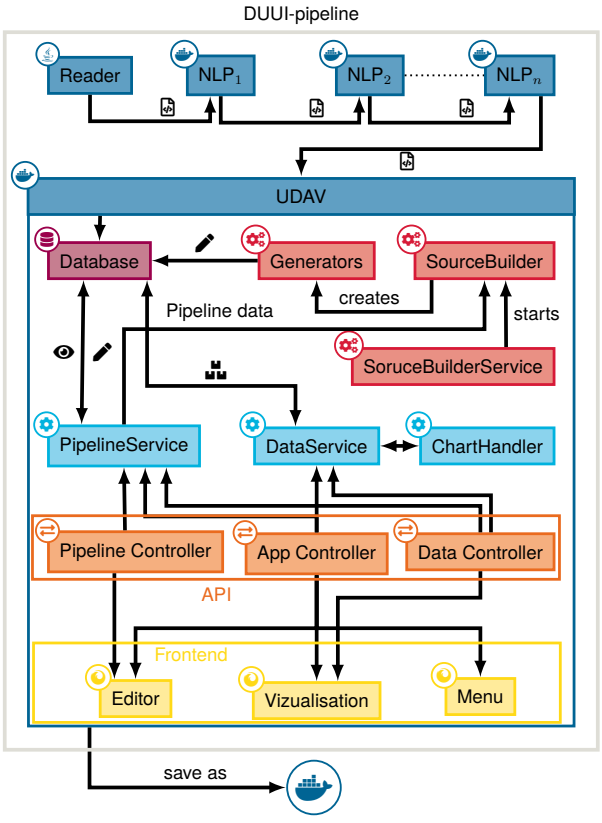

Figure 1: UDAV includes backend and frontend components that interact via an API layer, both instantiated in a Docker image. It runs on any system because all components are encapsulated in a Docker container. Importing data is performed using a standardized procedure as part of a DUUI pipeline. A reader ingests documents, which are then automatically pre-processed in an NLP pipeline. At the end of the pipeline, a Java-based importer loads all annotated documents into UDAV's integrated database.

[2] https://d3js.org/
[3] https://www.sigmajs.org/
[4] https://spring.io/
[5] https://freemarker.apache.org

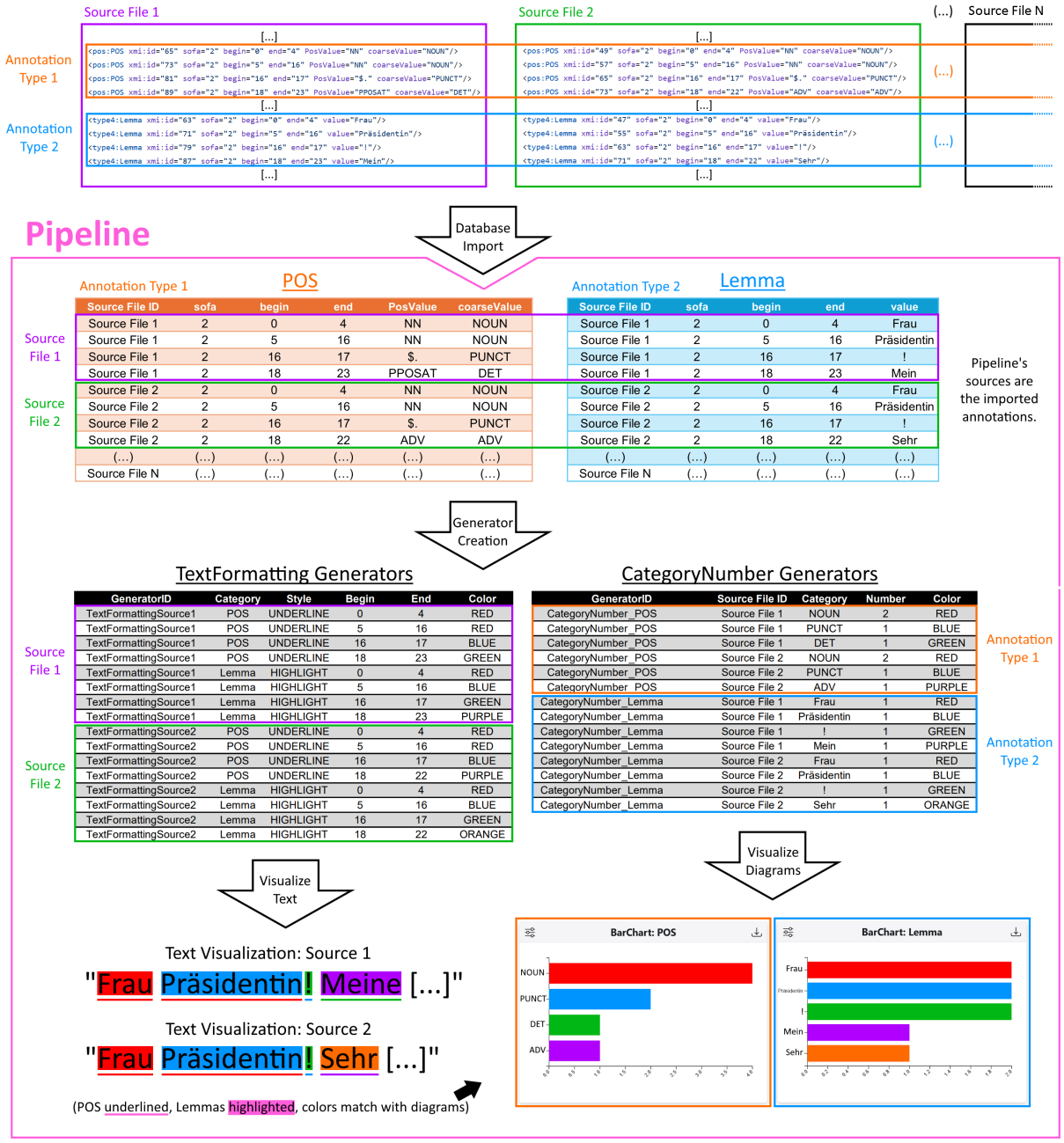

Figure 2: Overview of the PIPELINE architecture from UIMA annotations to UDAV widgets, including sources, GENERATORS, and the pipeline JSON schema. The example text is drawn from GERPARCOR (Abrami et al., 2024) and is annotated in UIMA.

reference. It runs asynchronously and independently of user requests, enabling efficient background processing. The importer can run standalone or, as in our prototype, within a big-data framework as a DUUI-COMPONENT.

C **Data Generation Layer**: To visualize annotations from imported UIMA documents subject to user requirements, UDAV implements a *Data Generation Layer* with two modules:

- **SourceBuilder** interprets PIPELINE configurations (see Section 4.2) and instantiates GENERATORS for each visualization.
- **GENERATORS** transform, aggregate, and structure annotations to meet the requirements of the API and underlying visualizations (according to the PIPELINE definition; cf. Section 4.2). That is, each diagram or visualization type with distinct data requirements is produced by a dedicated generator. To support new data structures, developers need to implement the standard GENERATOR interface with custom logic. The database schema en-

sures unique PIPELINES and their GENERATOR functionalities. Three GENERATORS are currently available: `CategoryNumber`, `TextFormatting`, textttMapCoordinates.

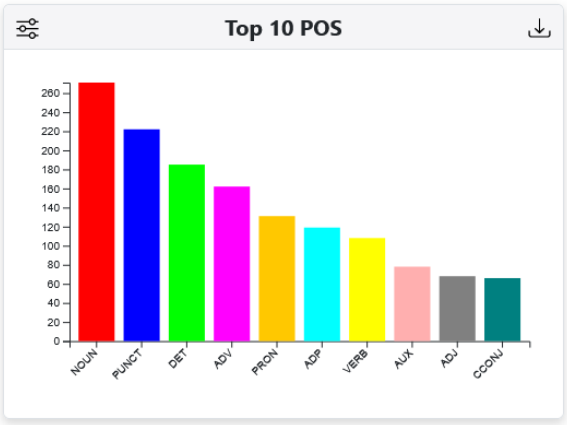

Figure 3: Dynamic visualization of POS annotations based on the PIPELINE in Code 1.

4.2. Pipeline

A UDAV PIPELINE defines each visualization's structure, configuration, composition, and interaction capabilities. This includes specifying which UIMA-class annotations to visualize and which GENERATOR to use. An integrated EDITOR (see Section 4.3) supports easy PIPELINE creation, which is serialized as JSON (see Code 1). Each PIPELINE consists of three parts:

1. **Sources** provides a list of annotation classes to visualize (see Code 1, line 6).

2. **Generators** define the data representations for the defined visualization type. There are different GENERATORS for various types of visualizations. Their implementation enables reuse across purposes, providing flexibility and data filtering. Pie and bar charts, for example, require similar parameters (labels, values, and colors). They can be created with a dedicated generator type `CategoryNumber`. This lets users switch between charts or view multiple perspectives on the same data, while defining generation only once. This enables new diagram types without changing the underlying data model. GENERATOR specifications (see Code 1, lines 7–11) can be customized per PIPELINE, and multiple GENERATORS can be combined (lines 14–19). This allows for synchronizing multiple GENERATORS to ensure consistent visualization results. Figure 2 illustrates how the GENERATORS `TextFormatting` and `CategoryNumber` use consistent colors for the same annotation classes.

3. **Widgets** concern the visual elements presented in the frontend, defined by type (e.g., pie charts, bar charts, highlighted text; see Figure 4). For bar and pie charts that aggregate annotations from multiple sources (see Figure 2), users can interactively select visualization documents in the frontend. The displayed values update dynamically. From a technical point of view, various visualization libraries can be integrated, as each visualization is defined via a FreeMarker template. The current implementation of UDAV uses the JavaScript framework d3.js[6].

Code 1: Exemplary pipeline configuration

```
{                                                                    1
  "id": "simple-demo",                                               2
  "sources": [                                                       3
    {                                                                4
      "id": "Source_POS1",                                           5
      "type": "POS",                                                 6
      "createsGenerators": [                                         7
      {"type":                                                       8
       "CategoryNumber",                                             9
       "id": "POS_Numbers"}                                          10
      ]                                                              11
    }                                                                12
  ],                                                                 13
  "derivedGenerators": [                                             14
    {                                                                15
      "id": "POS_Numbers_10",                                        16
      "fromGenerators": [{"id":                                      17
          ↪ "POS_Numbers",
          ↪ "settings": {"keepTop": 1
          ↪ 0}}]
    }                                                                18
  ],                                                                 19
  "widgets": [                                                       20
    {                                                                21
      "id": "bar-chart",                                             22
      "x": 0, "y": 2, "w": 4, "h": 3,                                23
      "type": "BarChart",                                            24
      "title": "Top 10 POS",                                         25
      "generator": {"id":                                            26
          ↪ "POS_Numbers_10" },
      "options": {"horizontal":                                      27
          ↪ false },
      "icon": "bi bi-bar-chart"                                      28
    }                                                                29
  ]                                                                  30
}                                                                    31
```

The EDITOR (Section 4.3) manages available GENERATOR elements, annotation-class selection, and frontend positioning. Based on the PIPELINE in Code 1, UDAV generates the visualization shown in Figure 3 for a document or an entire corpus.

[6] https://d3js.org/

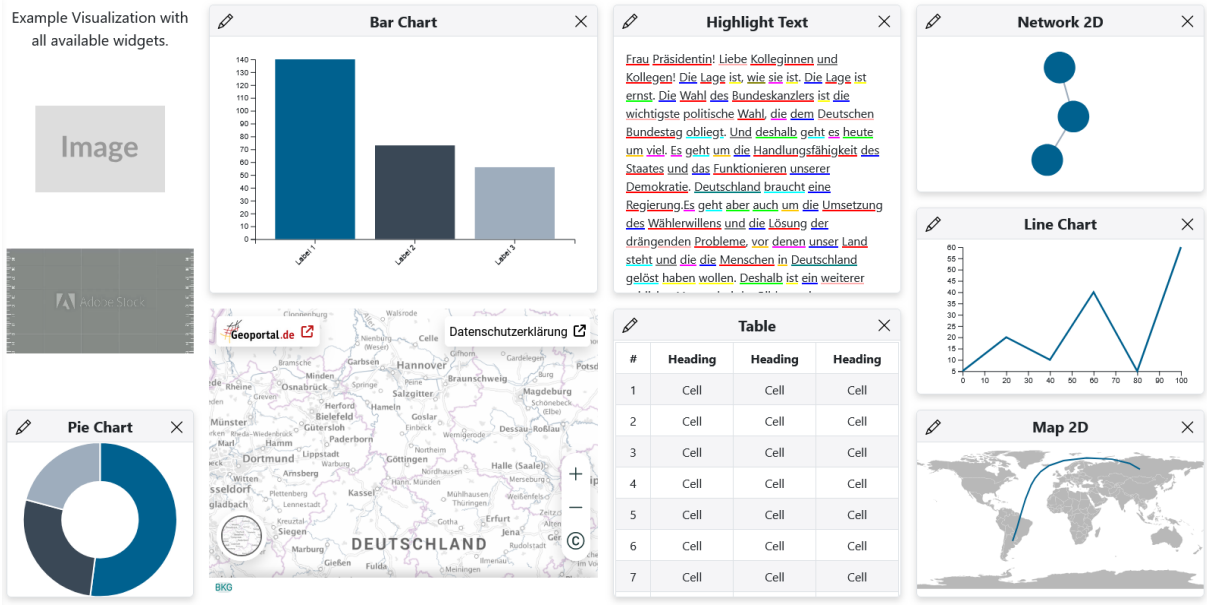

Figure 4: Overview of UDAV's extensible visualizations, depicting both static and dynamic widgets.

4.3. Provision and Interaction

Alongside its backend, UDAV includes a frontend that accesses annotations via the API and renders the pipelines described in Section 4.2. The pipelines can be selected, created, modified and deleted via a menu. The widgets created by the PIPELINES fall into two broad categories:

- *Static widgets* represent persistent elements such as text, images, videos and iframes, including project titles, logos and descriptions.
- *Dynamic widgets* visualize annotations interactively.

Dynamic widgets provide the features required for annotation visualization and interaction. This primarily includes filtering at the document or corpus level (Feature VII), which immediately updates the displayed annotation results (see Figure 6). Currently, dynamic widgets include text highlighting, 2D maps and networks, pie/line/bar charts, sortable tables, and show/hide controls for specific annotations (see Figure 4). All widget annotations can be exported at runtime in SVG, PNG, CSV, or JSON. Each export also includes corpus metadata, including filter settings, to ensure transparency. UDAV includes a drag-and-drop EDITOR to configure the visualizations (Figure 5). With EDITOR, users can create PIPELINES, arrange widgets without programming or manual JSON editing and have full control over visualization changes (Feature II). To add visual extensions (Feature IV), developers only need to add a new JavaScript class to the widgets folder and register it in the widget getter. At the same time, a corresponding form handler must be initialized to enable widget modification in the EDITOR.

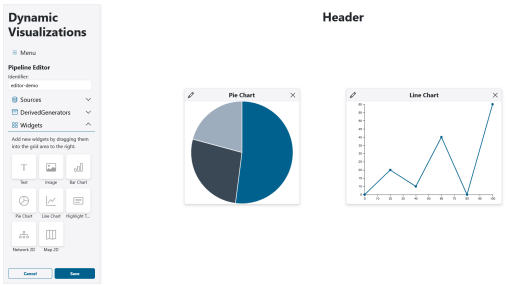

Figure 5: Example of UDAV's EDITOR for creating and modifying PIPELINES and arranging widgets.

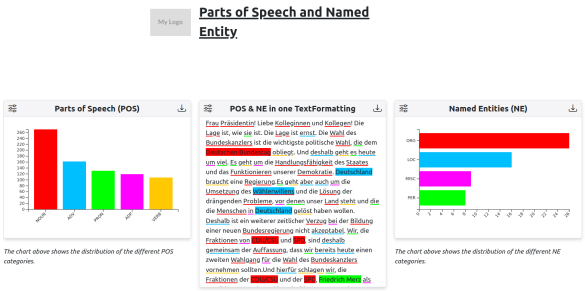

Figure 6: This pipeline view includes diagrams annotated for POS tags and NEs, plus static text for a title (top) and captions below each diagram. The middle shows a text selected via UDAV, with highlights matching the other diagrams' colors through a compatible GENERATOR (see Section 4.2).

5. Evaluation

To evaluate the effectiveness and usability of UDAV, we performed a quantitative evaluation that measures different aspects of corpus-data import and

also assesses performance. For this purpose, a sampled data set was extracted from GerParCor (Abrami et al., 2024), consisting of 3.5 GB of compressed (41 GB uncompressed) annotated UIMA-XMI files. The evaluation parameters selected were processing time, CPU time (Oracle, 2023b), and heap memory (Oracle, 2023a).

DUUI Import

Documents processed	1.299
Total import time	318.11 s
CPU time	2.693.531 ms
Available processors	8

Throughput

Docs / second	4.08
Characters / second	471,127
MB / second	0.459

Latency (per document)

Max latency	423 ms
Avg latency	31.2 ms

Document size (raw plain text extracted)

Total chars	149.872.142
Total bytes	153.087.908
Chars — avg / max	115.375 / 1.238.781
Bytes — avg / max	117.851 / 1.261.842

Annotations per document

Total annotations	122.895.985
avg / max	94.608 / 990.428

Memory (JVM heap)

Heap peak during import	13.418 MB
JVM max heap configured	15.784 MB

Figure 7: Results of the DUUI import into UDAV, including statistical data.

Pipeline build

Available processors	8

Time

Total processing time	2400 ms
JVM CPU time	1828 ms
Load pipeline	29 ms
Save visualisations	32 ms
Save generators (DB)	2331 ms

Memory (JVM heap)

Heap peak during build	360 MB
JVM max heap configured	15,784 MB

Figure 8: Results of the Pipeline build.

The results of the importer (Figure 7) and the Pipeline builder (Figure 8) indicate that the processing is fast with a thread count of 8. Due to the flexible increase in threads in DUUI, this can be parallelized to a certain extent, provided that the database is configured accordingly. Furthermore, the implementation is efficient, since heap consumption is minimal, which also allows it to be used on systems with limited resources.

6. Future Work

As listed in Table 1, UDAV includes features that are not yet fully implemented and will be realized in the next development steps. Currently, extensions for new visualizations can be added only by modifying template files; in the future, this will be implemented via the Editor with database support. In addition, only full texts can currently be visualized, which excludes multimodal content; this will be enabled in the next release (cf. Bundan et al. (2025)). Furthermore, support for AI models should be added so that visualizations can be generated dynamically from input prompts or via a RAG bot. This latter extension, in particular, points to one of the most important development directions for our future work: automatic, prompt-based creation of visualizations (A) using UDAV's existing functionality, (B) using prompt-based programming ("vibe coding") to develop UDAV extensions, and (C) by integrating agent technologies for largely proactive, automatic visualization generation. Ultimately, we aim for a broad spectrum of automation for interactive visualizations.

7. Conclusion

We presented Unified Dynamic Annotation Visualizer, a dynamic visualization framework based on UIMA. The core features of UDAV are the adaptability, extensibility, and configurability of pipeline elements. This enables the creation of various visualization widgets and Generators for API-based annotation aggregation. These features, along with continuous filtering capabilities, are underrepresented in the literature. This underscores the need for a new, unified approach, as presented here. UDAV can provide a foundation for straightforward and effective visualization of project results across scientific disciplines, especially when repetitive visualizations can be reused across corpora. This is supported by the low programming-skill requirement, since configurations can be modified using the integrated Editor. By publishing the full source code on GitHub under an AGPL license, the creation of further visualizations and Generators can become a community effort, increasing reusability.

Ethical Considerations

Within the scope of this work and the development of UDAV, no personal data was collected, processed or used, nor was a corpus created. The software solution presented enables corpora or documents to be visualized and explored. We acknowledge that this software solution could also be applied to restricted or prohibited content; however, the intention of this work is to support responsible and transparent scientific research.

Bibliographical References

Giuseppe Abrami, Mevlüt Bagci, and Alexander Mehler. 2024. German parliamentary corpus (GerParCor) reloaded. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 7707–7716, Torino, Italy. ELRA and ICCL.

Giuseppe Abrami, Mevlüt Bagci, Leon Hammerla, and Alexander Mehler. 2022. German Parliamentary Corpus (GerParCor). In *Proceedings of the Language Resources and Evaluation Conference*, pages 1900–1906, Marseille, France. European Language Resources Association.

Giuseppe Abrami, Markos Genios, Filip Fitzermann, Daniel Baumartz, and Alexander Mehler. 2025. Docker Unified UIMA Interface: New perspectives for NLP on big data. *SoftwareX*, 29:102033.

Giuseppe Abrami and Alexander Mehler. 2018. A UIMA database interface for managing NLP-related text annotations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Shailendra S. Aote and Archana Potnurwar. 2018. An automatic video annotation framework based on two level keyframe extraction mechanism. *Multimedia Tools and Applications*, 78(11):14465–14484.

Kevin Bönisch, Giuseppe Abrami, and Alexander Mehler. 2025. Towards unified, dynamic and annotation-based visualisations and exploration of annotated big data corpora with the help of unified corpus explorer. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (System Demonstrations)*, pages 522–534, Albuquerque, New Mexico. Association for Computational Linguistics. Best Demo Award.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 149–164, New York City. Association for Computational Linguistics.

Daniel Bundan, Giuseppe Abrami, and Alexander Mehler. 2025. Multimodal Docker Unified UIMA Interface: New Horizons for Distributed Microservice-Oriented Processing of Corpora using UIMA. In *Proceedings of the 21st Conference on Natural Language Processing (KONVENS 2025): Long and Short Papers*, KONVENS '25, pages 257–268, Hannover, Germany. HsH Applied Academics.

Yuexin Cai, Junbo Zeng, Liping Lan, Suijun Chen, Yongkang Ou, Linqi Zeng, Qintai Yang, Peng Li, Yubin Chen, Qi Li, Hongzheng Zhang, Fan Shu, Guoping Chen, Wenben Chen, Yahan Yang, Ruiyang Li, Anqi Yan, Haotian Lin, and Yiqing Zheng. 2022. Expert recommendations on collection and annotation of otoscopy images for intelligent medicine. *Intelligent Medicine*, 2(4):230–234.

Bowen Chen, Huan Ling, Xiaohui Zeng, Jun Gao, Ziyue Xu, and Sanja Fidler. 2020. Scribblebox: Interactive annotation framework for video object segmentation. In *Computer Vision – ECCV 2020*, pages 293–310, Cham. Springer International Publishing.

H. Cunningham, R.G. Gaizauskas, and Y. Wilks. 1995. A general architecture for text engineering (gate) – a new approach to language engineering r&d. Technical Report CS – 95 – 21, Department of Computer Science, University of Sheffield. `http://xxx.lanl.gov/abs/cs.CL/9601009`.

David Ferrucci, Adam Lally, Karin Verspoor, and Eric Nyberg. 2009. Unstructured Information Management Architecture (UIMA) Version 1.0. OASIS Standard.

Georg Fette, Martin Toepfer, and Frank Puppe. 2013. Storing UIMA CASes in a relational database. *Unstructured Information Management Architecture (UIMA)*, page 10.

Angus Forbes, Kristine Lee, Gus Hahn-Powell, Marco A. Valenzuela-Escárcega, and Mihai Surdeanu. 2018. Text annotation graphs: Annotating complex natural language phenomena. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024. Retrieval-augmented generation for large language models: A survey.

T. Götz and O. Suhre. 2004. Design and implementation of the UIMA common analysis system. *IBM Systems Journal*, 43(3):476–489.

Ulrich Heid, Helmut Schmid, Kerstin Eckart, and Erhard W. Hinrichs. 2010. A corpus representation format for linguistic web services: The d-spin text corpus format and its relationship with iso standards. In *International Conference on Language Resources and Evaluation*.

Wahed Hemati, Tolga Uslu, and Alexander Mehler. 2016. Textimager: a distributed uima-based system for nlp. In *Proceedings of the COLING 2016 System Demonstrations*. Federated Conference on Computer Science and Information Systems.

Wei Ji, Renjie Liang, Lizi Liao, Hao Fei, and Fuli Feng. 2023. Partial annotation-based video moment retrieval via iterative learning. In *Proceedings of the 31st ACM International Conference on Multimedia*, MM '23, page 4330–4339, New York, NY, USA. Association for Computing Machinery.

Yoshinobu Kano, Jr Baumgartner, William A., Luke McCrohon, Sophia Ananiadou, K. Bretonnel Cohen, Lawrence Hunter, and Jun'ichi Tsujii. 2009. U-compare: share and compare text mining tools with uima. *Bioinformatics*, 25(15):1997–1998.

Adam Kilgarriff, Pavel Rychlý, Pavel Smrž, and David Tugwell. 2004. The sketch engine. In *Proceedings of the 11th EURALEX International Congress*, pages 105–116. Université de Bretagne-Sud, Faculté des lettres et des sciences humaines.

Hannah Kim, Kushan Mitra, Rafael Li Chen, Sajjadur Rahman, and Dan Zhang. 2024. MEGAnno+: A human-LLM collaborative annotation system. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 168–176, St. Julians, Malta. Association for Computational Linguistics.

Alexander Leonhardt, Giuseppe Abrami, Daniel Baumartz, and Alexander Mehler. 2023. Unlocking the heterogeneous landscape of big data NLP with DUUI. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 385–399, Singapore. Association for Computational Linguistics.

Frederic P Miller, Agnes F Vandome, and John McBrewster. 2010. *Apache Maven*. Alpha Press.

Minh-Quoc Nghiem, Paul Baylis, and Sophia Ananiadou. 2021. Paladin: an annotation tool based on active and proactive learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 238–243, Online. Association for Computational Linguistics.

Jingwei Ni, Minjing Shi, Dominik Stammbach, Mrinmaya Sachan, Elliott Ash, and Markus Leippold. 2024. AFaCTA: Assisting the annotation of factual claim detection with reliable LLM annotators. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1890–1912, Bangkok, Thailand. Association for Computational Linguistics.

Oracle. 2023a. Interface memorymxbean. Accessed: 2026-03-05.

Oracle. 2023b. Interface operatingsystemmxbean. Accessed: 2026-03-05.

A. Z. Shikder. 2024. The controller-service-repository pattern: A comprehensive guide. Accessed: 2025-10-19.

Stéfan Sinclair and Geoffrey Rockwell. 2016. Voyant tools.

PW Skills. 2024. Architecture of spring boot: Examples, pattern, layered controller layer. Accessed: 2025-10-19.

Yushi Sun, Hao Xin, and Lei Chen. 2023. Reca: Related tables enhanced column semantic type annotation framework. *Proc. VLDB Endow.*, 16(6):1319–1331.

Xuemei Tang, Zekun Deng, Jun Wang, and Qi Su. 2025. Chisnere: a premodern chinese corpus with named entity and relation annotation. *Digital Scholarship in the Humanities*, 40(2):617–638.

Hrishikesh Terdalkar and Arnab Bhattacharya. 2023. Antarlekhaka: A comprehensive tool for multi-task natural language annotation. In *Proceedings of the 3rd Workshop for Natural Language Processing Open Source Software (NLP-OSS 2023)*, pages 199–211, Singapore. Association for Computational Linguistics.

Olga Uryupina, Barbara Plank, Gianni Barlacchi, Francisco J. Valverde Albacete, Manos Tsagkias, Antonio Uva, and Alessandro Moschitti. 2016. LiMoSINe pipeline: Multilingual UIMA-based NLP platform. In *Proceedings of ACL-2016 System Demonstrations*, pages 157–162, Berlin, Germany. Association for Computational Linguistics.

Shafqat Mumtaz Virk, Claes Ohlsson, Nina Tahmasebi, Henrik Björck, and Leif Runefelt. 2024. Enhancing Swedish parliamentary data: Annotation, accessibility, and application in digital humanities. In *Proceedings of the 4th International Conference on Natural Language Processing for Digital Humanities*, pages 280–288, Miami, USA. Association for Computational Linguistics.

Antoine Widlöcher and Yann Mathet. 2012. The glozz platform: A corpus annotation and mining tool. In *Proceedings of the 2012 ACM symposium on Document engineering*, pages 171–180. ACM.

A. Widlöcher. 2008. *Analyse macro-sémantique des structures rhétoriques du discours : cadre théorique et modèle opératoire [Caen]*. Phd thesis. Http://www.theses.fr/2008CAEN2042.

Johannes Wild and Markus Pissarek. 2022. Ratte. regensburger analysetool für texte. version 2.0. Accessed 2023/05/02.