AnnoPlot: Interactive Visualizations of Text Annotations

Elisabeth Fittschen[†], **Daniel Bruehl**[†], **Julia Spahr**[†]
Phuoc Thang Le[‡], **Yuliia Lysa**[†], **Tim Fischer**[‡]
Language Technology Group, Department of Informatics, Universität Hamburg, Germany
[†] {firstname.lastname}@studium.uni-hamburg.de,
[‡] {firstname.lastname}@uni-hamburg.de

Abstract

This paper presents AnnoPlot, a web application designed to analyze, manage, and visualize annotated text data. Users can configure projects, upload datasets, and explore their data through interactive visualization of span annotations with scatter plots, clusters, and statistics. AnnoPlot supports various transformer models to compute high-dimensional embeddings of text annotations and utilizes dimensionality reduction algorithms to offer users a novel 2D view of their datasets. A dynamic approach to dimensionality reduction allows users to adjust visualizations in real-time, facilitating category reorganization and error identification. The proposed application is open-source, promoting transparency and user control. Especially suited for the Digital Humanities, AnnoPlot offers a novel solution to address challenges in dynamic annotation datasets, empowering users to enhance data integrity and adapt to evolving categorizations.

1 Introduction

Within the context of NLP, sequence annotation is the practice of tagging sections of text with appropriate labels. For example, this could involve classifying the phrase "Finding Nemo" as a "movie" in the sentence "I love Finding Nemo."

Annotation projects, such as the creation of annotated datasets, face a variety of challenges, one of which is data quality and validity. As data is added and use cases evolve, prior category definitions might be insufficient and need to be updated or reorganized. As datasets grow and evolve, achieving a good overview becomes increasingly complex, which is necessary to revise labeling strategy and category definition (Payan et al., 2021).

AnnoPlot addresses this issue by offering comprehensive two-dimensional views of the provided dataset without any prior training or classification model necessary. Such views of span annotations and category systems enable users to:

- quickly gain an overview of large datasets and their overarching category structure
- reorganize categories within the dataset
- interactively adjust the visualization to fit the intended category definition better
- better identify and fix potential erroneous annotations

Many of these functionalities are valuable for applications in NLP and especially for the management of Named Entity Recognition datasets. However, they are also of great interest to domains and disciplines beyond those. In the digital humanities, annotating datasets, building complex category systems, and the refinement of such are typical workflows of qualitative, hermeneutic research processes.

AnnoPlot generates two-dimensional representations akin to systems such as BERTopic (Grootendorst, 2022). First, the span annotations are encoded with a pre-trained transformer model, yielding large embeddings, typically of 512 up to 1024 dimensions. In this work, we use BERT-based transformer encoders since BERT (Devlin et al., 2019) embeddings have proven to be very information-rich, meaning a large variety of data is encoded in the high-dimension embeddings.

Next, dimensionality reduction techniques are implemented to generate a two-dimensional representation for visualization purposes.

Our proposed tool offers an interactive dimension reduction algorithm based on the well-known UMAP (McInnes et al., 2020) algorithm and its parametric neural network implementation (Sainburg et al., 2021). The user can remotely train the small reduction algorithm network to fit annotated text passages to the categories. In case of further customization necessity, dots or clusters representing annotations can be dragged in the interactive view to inform the model about the desired changes for the subsequent training iterations. This training happens in real-time; the user can see updates or

Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics
System Demonstrations, pages 106–114

stop the process in case of mistakes or overfitting.

AnnoPlot provides a customizable and dynamic application for the analysis of annotated datasets that stands apart from expensive commercial solutions like MAXQDA or Galileo. Our application lets users interactively modify annotations, hierarchical category systems, and visualizations in real-time without requiring pre-trained, category-specific models. By bridging the gap between advanced data visualization techniques and user-friendly applications, AnnoPlot makes a noteworthy addition to current annotation tools. AnnoPlot is an open source project; the code can be found in its github repository[1] together with a short video demonstrating its core functions. A live demonstration of AnnoPlot can be found here[2].

2 Related Work

AnnoPlot offers an interactive user interface for data visualization of annotated texts and classification category management. In the context of category structuring and reorganization, AnnoPlot can be compared with MAXQDA's *Creative Coding* feature. MAXQDA[3], a qualitative data analysis software, facilitates the organization and hierarchical structuring of codes through its *Creative Coding* functionality. This process involves generating, sorting, and organizing codes, defining relationships between them, and creating a hierarchical structure. However, *Creative Coding* primarily focuses on manual categorization and organization and does not offer any automation. In contrast, AnnoPlot leverages embedding representations to offer an initial starting point while offering similar category management functionality.

Examining general visualization, AnnoPlot's 2D visualization of annotated text passages can be contrasted with tools like the embedding view functionality of Galileo[4]. Galileo is a platform that provides tools and modules to evaluate machine learning classifiers, including NLP classifiers for sequence annotation. The Embedding view utilizes embeddings and a parametric UMAP reduction to visualize classifier results. However, its primary focus is on showcasing classifier performance, and potential areas of uncertainty, rather than facilitating dataset analysis or category reorganization. AnnoPlot, on the other hand, is designed for quick and interactive analysis of annotated text datasets, enabling users to gain insights into the overarching structure and quality of their annotations. The absence of a classifier in AnnoPlot makes its interactive dimension reduction feature particularly valuable, allowing for real-time adjustments and customization based on user input.

In terms of visualization techniques that utilize a processing pipeline of embedding and dimensionality reduction, AnnoPlot shares similarities with many applications such as BERTopic (Grootendorst, 2022) and the TensorFlow embedding projector (Smilkov et al., 2016).

In regards to user interactions, research analysis of interactive dimension reducing tools (Sacha et al., 2017) identified seven distinct categories of interactive dimensionality reduction algorithms, three of which apply to AnnoPlot:

- Annotation & Labeling: the visualization can leverage user given/changeable annotation data when fitting visualization to correspond to categories.
- Parameter Tuning: the user can alter the visualization by tuning UMAP hyperparameters and choosing which transformer model computes the embeddings.
- Data Manipulation: the user can move data points in the dynamic view, and get feedback on how other points move in response.

A notably similar approach to interactive data visualization is taken by Zexplorer (González Martínez et al., 2020) an extension of the bibliography system Zotero. Zexplorer maps selected features of research papers, including a BERT embedding of the paper's abstract, to two dimensions utilizing a neural network approximation of UMAP. Similar to AnnoPlot it offers interactive training of this network through draggable data points.

The development of AnnoPlot was motivated by the absence of interactive features for the visual analysis of annotations in the most popular (Neves and Ševa, 2019) open-source annotation tools such as WebAnno (Yimam et al., 2014), brat (Stenetorp et al., 2012), CATMA (Gius et al., 2023), and INCEpTION (Klie et al., 2018). AnnoPlot combines contextualized embedding representations of annotations with interactive visualizations for analyzing annotated datasets and category structuring.

[1] https://github.com/uhh-lt/anno-plot
[2] https://anno-plot.ltdemos.informatik.uni-hamburg.de/
[3] https://www.maxqda.com/
[4] https://www.rungalileo.io/

3 System Architecture

The application is designed to visualize annotated text segments within the provided datasets and facilitate the modification of existing annotations. Further, it computes and assigns clusters to text segments, helping users to identify annotation errors. It allows users to create projects to manage datasets within it and the flexibility to import data in standard data formats such as CoNLL (Tjong Kim Sang and De Meulder, 2003). When exporting the data, datasets within a project are merged. The tool includes a search function to navigate the dataset and the clusters. Finally, the tool offers detailed statistics for projects, codes, and clusters.

AnnoPlot offers two views: one dedicated to presenting annotated text segments and the other focused on hierarchical category systems. Users can create, delete, and rename data points in both views. Furthermore, users can merge categories and train the existing embeddings further. The views are rendered using the D3.js library. In general, the frontend implementation of the application is realized through React/Next.js.

The backend was implemented in Python using FastAPI and PostgreSQL for data storage. The data processing pipeline to compute 2D representations of annotated text passages consists of a BERT-based transformer model, UMAP, and HDBSCAN. The Hugging Face library is employed to load models and allows for flexibility in which model to use in the pipeline.

The hardware requirements for running AnnoPlot are largely dependent on the embedding model chosen. For development and testing, we used the "bert-base" model with a 3080 RTX GPU (10 GB). The backend uses transformer/BERT-based models for inference and, at most, trains a small (4 layer) dynamic umap model. While a GPU is not strictly required it is highly recommended for UX performance. Annoplot is deployed using Docker. Every System component i.e., the database, frontend, and backend is deployed in a separate docker container, which are orchestrated in a docker compose file.

3.1 Visualization of Annotated Text Segments

The Plot View (see Figure 1), visualizes the uploaded annotated data. Each dot corresponds to an annotated text passage, for example, the annotated named entity "Finding Nemo" in the sentence "My favorite movie is Finding Nemo"; the color indicates the user-given category, i.e., "Movie".

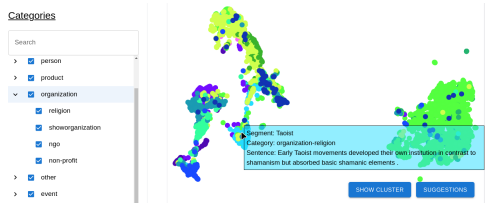

Figure 1: The Plot View, generated using the static UMAP dimension reduction. Left is an overview of the hierarchical category system for filtering the annotations. A tool-tip shows related information about the hovered annotation, including the context sentence.

Dots corresponding to similar entities tend to cluster together. Ideally, this clustering occurs according to the pre-defined categories.

User Interaction Overview

Dots are the product of embeddings of corresponding annotations and a reduction function. These two processes can be customized in the Configurations tab. The user can choose which embedding model to use by linking to a corresponding Huggingface repository; any BERT or RoBERTa-based (Zhuang et al., 2021) model is compatible. This also allows the user to use their own fine-tuned models. The user can also choose between a static and dynamic UMAP model to be used for dimensionality reduction, of which the "n components" hyperparameter can be adjusted.

The main features of this view are the overview of the dataset and functionalities to revise annotations and category choices. For example, the plot can indicate that two categories are not clearly separated or overlap significantly; this might suggest sub-optimal category choice or inconsistent annotation. Further, the plot can reveal that annotations of the same category group in different regions; this might suggest introducing a new (sub-) category.

To the plot's left is an overview of all categories. These can be selected and de-selected to filter for those of current interest. Hovering over dots allows the user to see the respective entity data. Upon right-clicking, a context menu allows the user to re-categorize or delete the corresponding named entity.

If the dynamic UMAP model was selected, the user has two more options, as shown in Figure 2.

The user can train the model according to categories by pressing the train button. This option uses the annotated data for semi-supervised training of the dynamic UMAP model. It will be trained for ten epochs; after each epoch, the model updates,

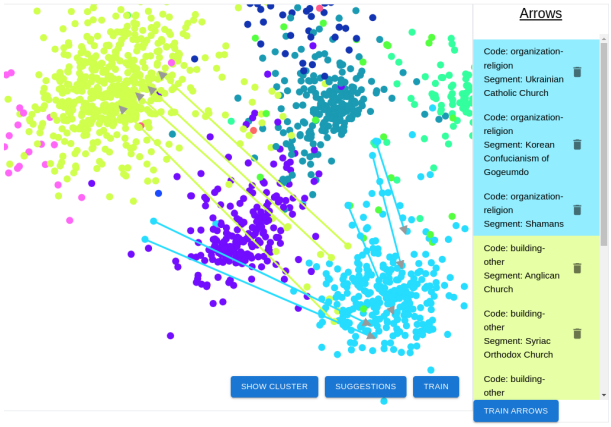

Figure 2: The Plot View, generated using the dynamic UMAP dimension reduction. Arrows have been manually drawn to denote the desired position for multiple points. Pressing the "Train Arrows" button will rearrange the visualization accordingly.

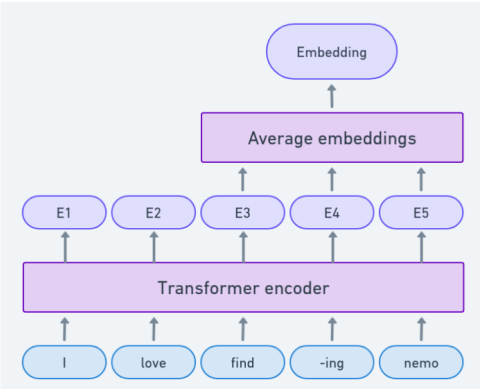

Figure 3: Visualization of the Embedding process. The sentences surrounding a span annotation are embedded by a transformer encoder. Tokens of the annotated entity are averaged to compute the final entity embedding.

and the dots move to their new positions. The user can choose to preemptively terminate the training steps by pressing stop. Training will stop after the current epoch has finished.

The user can also move data points to a desired position. When dragging points, an arrow will appear, following the cursor, indicating the direction the dot should move. The user can do this for multiple dots appearing on the right. When the user is done, clicking the train button will trigger the model to train for these new dot positions.

When using the dynamic model, the user can go through all dots not within their intended cluster and either fix their annotation or move them, thereby fine-tuning the UMAP reduction.

Implementation Details

Dot coordinates are generated in two steps. First, a context-rich, high-dimensional embedding (depending on the model typically 512-1024) is created from the entity data (entity and its surrounding text). Then, the embeddings of all data points are reduced to two dimensions using a dimension reduction algorithm.

There are dozens of combinations of embedding models and dimension reduction algorithms. In this paper, we focus on a BERT-based embedding model and UMAP-based reduction algorithms.

Embeddings are generated by passing the annotated text passage, e.g. a named entity, and surrounding sentences (limited by the model's input size) into a transformer-based encoder. This results in high dimensional embeddings per input tokens. The token embeddings, part of the named entity, are averaged to create the entity embedding (see Figure 3). Input token sequences are embedded in batches to enhance the processing efficiency.

Next, the embeddings of all annotated text segments within one dataset are passed to UMAP, which maps the data to two dimensions according to their structure in high-dimensional space.

In the process described, the 2D representation of an annotated text passage is created independent of its category, as the category is not passed to the embedding model. This results in a deterministic mapping from an annotation to two dimensions for a specific annotated dataset. This is a significant disadvantage of the proposed process, as such representation cannot possibly accurately capture the wide variety of classification schemas possible. The named entity *Finding Nemo* in "I love Finding Nemo" could, for instance, be categorized by type (Movie/TV series), company (Disney/Pixar), or release decade (the 2000s/2010s). However, with the current approach, it would be mapped to the same 2D coordinates, which is not desired.

For this reason, the tool also implements a dynamic UMAP solution for dimensionality reduction. Here, the user can interactively train a parametric neural network UMAP (Sainburg et al., 2021) to fit the user's category system and interpretation. The network employs a comparatively modest architecture, consisting of a single input layer that dynamically adjusts to the input size, followed by three hidden layers, each comprising 200 neurons, and a final output layer with a size determined by the number of components (defaulting to 2). This relatively small model reduces the

risk of overfitting and, therefore, of diluting the knowledge encoded within the embeddings.

There are two options to train the reduction model. The first option utilizes triplet loss to adjust the representations to better separate the categories from each other.

The triplet loss approach operates by selecting anchors, positives, and negatives to form triplets for each unique label in the data. The loss function used here is the TripletMarginLoss, defined mathematically as

$$L = \sum \max\left(d(a,p) - d(a,n) + margin, 0\right),$$

where $d(x, y)$ represents the distance between two points and a, p, n denotes the anchor, positive, and negative samples, respectively. In our case, the margin was set to 1.0.

The second approach allows for a more direct, interactive manipulation of the data points. Users can 'move' dots, and the model is trained to optimize for the new positions of these dots. This retraining uses Mean Squared Error (MSE) loss, with the positions of other dots being held in place using a moving average.

Both training methods utilize the Adam optimizer. The learning rate was set to 2×10^{-4} for the triplet loss method and to 5×10^{-5} for the direct dot movement method.

3.2 Visualization of the Category System

In the Category View (see Figure 4), categories are visualized in a bubble chart, mirroring the presentation style of the Plot View. It provides an overview of all categories within the imported datasets.

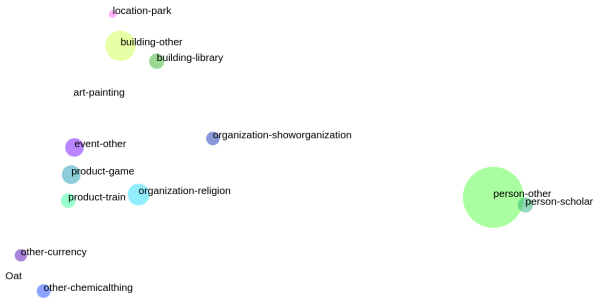

Figure 4: Category View, corresponding to the data seen in Figure 1. Notably, person, scholar (right two points) and location, building, library (top three points) cluster together.

User Interaction Overview

The Category View helps the user reorganize and refine the categories by merging, deleting, and changing their hierarchical structure.

Like in the Visualization for Annotated Text Segments (Figure 1), a legend on the left side shows all possible categories and sub-categories within a project, allowing the user to search or filter for specific categories.

The colors and positions of categories are in sync with the Annotated Text Segments View. The number and position of annotated text segments within a category determine the size of the bubble, providing users with a quick overview of the distribution of annotations across different categories.

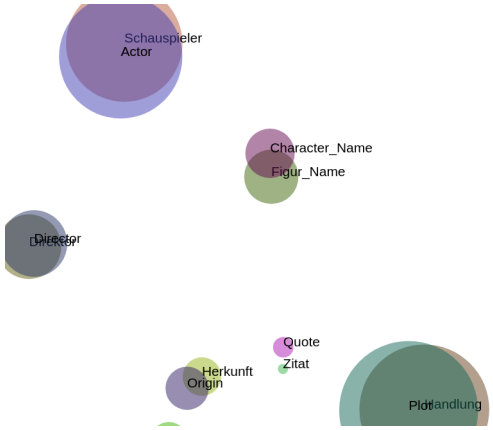

Figure 5: Visualization of MITMovieCorpus(eng, ger). Notably, the German categories like "Schauspieler", "Direktor" and "Figur Name" are very close to their English counterparts (Actor, Director, Character Name).

The user can use these spatial relations to get a quick insight into their data. In Figure 5, for instance, the categories of two datasets can be seen; one is part of the MIT Movie Corpus (Liu et al., 2013) annotated using German labels, the other using English labels. The categories containing semantically similar data are easily identifiable, often wholly overlapping. Even without understanding German, it is straightforward to determine which categories can be merged.

By right-clicking a category, the category or subcategory can be renamed, deleted, or merged with another category. At the bottom of the view, additional options allow the creation, removal, or merging of categories.

Visualizing category positions, sizes, and colors offers users an intuitive and comprehensive tool for managing and understanding the structure of their annotated datasets.

Implementation Details

The position and size of categories are calculated using the underlying annotated text segments and their position in 2D (see Section 3.1). A category's position is the average of all its entities' positions, whereas its radius corresponds to its number of entities, whereas a category's radius corresponds to its number of entities.

3.3 Automatic Error Detection

In the annotated text segment view discussed in Section 3.1, the user is also supported in finding categorization errors. The identification of annotation errors is achieved through clustering and outlier detection.

User Interaction Overview

AnnoPlot aids in finding potential annotation errors through the HDBSCAN (Campello et al., 2013) clustering algorithm. Similarly to the embedding and dimensionality reduction functions, this algorithm can also be configured in the Configurations tab. Here, the user can adjust the "min cluster size", "metric", and the "cluster selection method."

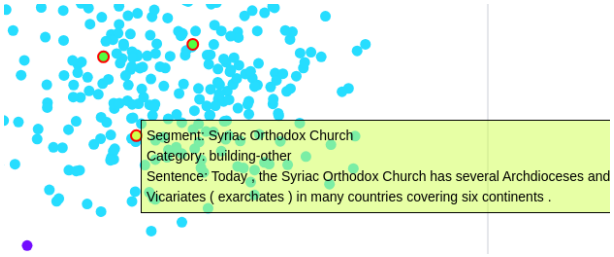

Figure 6: Error identification. The light blue dots are a cluster of religious organizations. A user must now decide if this annotation is correct or incorrect.

The user can choose to show automatically detected errors. All found inconsistent annotations are marked by a red circle and brought to the forefront of the scatter plot, allowing the user to see otherwise obstructed dots (see Figure 6).

The user can also view the raw cluster algorithm result when pressing the "show clusters" button. The result of this can be seen in Figure 7. This can help to understand the error suggestion and allows for transparency. Found annotation errors can be fixed by deleting or re-categorizing annotations that correspond to the dots. Users using the Dynamic UMAP model can additionally choose to move the dot to the category of choice and train the reduction model.

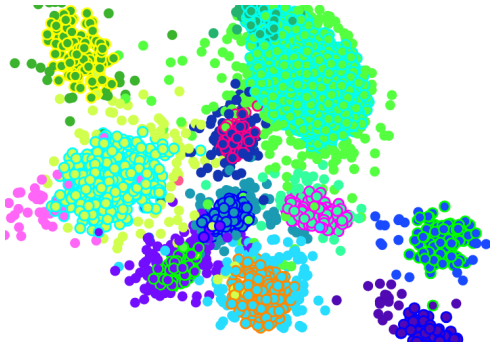

Figure 7: Visualization of calculated clusters as shown in the Plot View. Each cluster is marked by a distinct dot outline color.

Implementation Details

The HDBSCAN clustering algorithm is executed upon each data modification. For every identified cluster, the predominant category label is computed. We employ a simple heuristic to detect annotation errors: A cluster is designated as belonging to a specific category if over 70% of its data points fall within that category. Any data points within these clusters that do not align with the dominant category are flagged as errors. These erroneous points are sent to the frontend and visualized accordingly.

4 Evaluation

To evaluate the effectiveness of error detection and overall clustering validity, errors were intentionally introduced to a dataset. The tool was then monitored on its ability to identify these via its 'error detection' feature.

Dataset Preparation

The CoNLL-2003 (Sang and Meulder, 2003) dataset was chosen as a base, it has 23 thousand annotated segments and four categories "ORG", "MISC", "LOC" and "PER". Errors were uniformly introduced across all categories by selecting and modifying a proportional subset of segments. Labels of selected segments were randomly reassigned to alternative labels. Datasets with error rates of 1%, 5%, and 10% were generated.

Methodology

The datasets were uploaded to AnnoPlot and evaluated using both static and dynamic UMAP reduction. The dynamic UMAP reduction was evaluated for 50 (category) training epochs. The transformer model used for embedding was 'bert-base-uncased', a model without previous training on this task or dataset.

Table 1: The results of the error detection evaluation. True positives (TP), false positives (FP), precision (P%), and recall (R%) are reported for the three different error rates. Both approaches, static UMAP and dynamic UMAP (at epochs 2, 5, 10, 20, 50) were evaluated. The best precision and recall scores are highlighted.

	1% (229 errors)				5% (1163 errors)				10% (2328 errors)			
# epochs	TP	FP	P%	R%	TP	FP	P%	R%	TP	FP	P%	R%
static	60	966	5.85	26.2	284	789	26.47	24.4	573	682	45.66	24.6
2	188	736	20.35	82.1	938	711	56.88	80.7	1779	694	71.94	76.4
5	199	367	35.16	**86.9**	1020	501	67.06	87.7	1969	445	81.57	84.6
10	195	225	46.43	85.2	1039	312	76.91	**89.3**	1983	244	89.04	**85.2**
20	136	42	76.40	59.4	755	52	93.56	64.9	1443	69	**95.44**	62.0
50	42	2	**95.45**	18.3	201	8	**96.17**	17.3	491	24	95.34	21.1

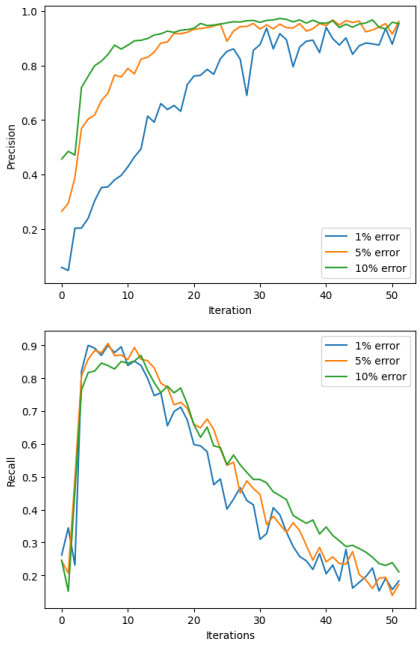

Figure 8: Precision (top) and recall (bottom) of the dynamic UMAP error detection over 50 epochs

Results

Table 1 shows a representative subset of the results. Notably, the dynamic UMAP approach consistently reached higher precision and recall scores than static UMAP. As training progresses, the model seems to overfit, reducing the total amount of errors detected (see Figure 8). However, precision increases at the same time, i.e. only "clear" errors are detected. Optimal F1 scores of 0.705, 0.841, and 0.881 were attained for the datasets (1%, 5%, 10% error) at iterations 13, 11, and 10, respectively. It should be noted that the error detection method only detects points situated within incorrect category clusters. Points outside of or on the periphery of clusters will not be detected.

5 Conclusion & Future Work

In this paper, we presented AnnoPlot, a web application that allows users to easily visualize, manage, and analyze annotated datasets. The proposed tool offers an efficient and streamlined solution for working with span annotations and their category system in a novel way on a 2D canvas. It enables users to upload datasets, add configurations, and manage them as projects, with different data extracted and presented across separate pages. The category and plot views are the main features, displaying categories and annotations clustered together. Users can perform tasks like merging, deleting, and rearranging annotated text passages directly within the view. The plot view also enables the training of dynamic UMAP to optimize the visualization and identify any errors or outliers in the clusters. Statistics are available for categories, clusters, and projects.

This work focused on the visualization, management, and analysis of annotated text datasets. However, extending the proposed approach and interface to support annotated data of other modalities is straightforward. In a future iteration of this tool, we will support image annotations by switching the embedding model with appropriate visual transformers and adapting the software to handle image datasets. This broadens our tool's analytical scope to encompass a large variety of data types.

Further, AnnoPlot will offer more varied dimensionality reduction algorithms, including t-SNE and Principal Component Analysis.

Finally, it is planned to integrate AnnoPlot into a more comprehensive open-source annotation software as an interactive visual analysis feature.

Limitations

The tool, especially the embedding process, relies on transformer models imported from the Hugging Face website. The tool is unable to download new models in case of Hugging Face unavailability.

The embedding process currently relies on large transformer models with a time complexity of $O(n^2)$, limiting the context data available for the embedding generation to about 512-1024 tokens. In our experiments, we limited the context of an annotated text passage to the surrounding sentences and did not run into the context limitation of 1024 tokens.

The visualization of annotated text segments is rendered using D3.js. Currently, up to 10.000 data points can be rendered reliably. To compensate for this, AnnoPlot offers filtering by categories to limit the amount of rendered annotations and allow the analysis of large annotated datasets.

Ethics Statement

AnnoPlot is designed with ethical considerations in mind, prioritizing user privacy, transparency, and control. Several key points highlight the ethical stance of the software:

1. **Open Source and Local Execution:** AnnoPlot is an open-source project, and users have the option to run it locally. This ensures transparency in the codebase and provides users with control over their data.

2. **No Collection of Private Data:** AnnoPlot respects the confidentiality and integrity of user data. The tool does not engage in any unauthorized sharing or transmission of data, and users maintain complete control over their datasets.

3. **Human in the Loop:** AnnoPlot adopts a human-in-the-loop approach, where users actively participate in the annotation and correction process. The tool serves as an assistant to users, allowing them to visualize and manage annotated data effectively.

4. **Potential for Bias Control:** The interactive nature of AnnoPlot, especially in the dynamic UMAP training, allows users to identify and rectify potential biases or errors in categorization. This empowers users to ensure fairness and accuracy in their annotated datasets.

In summary, AnnoPlot is designed to be a responsible and ethical tool, prioritizing user privacy and autonomy.

References

Ricardo Campello, Davoud Moulavi, and Joerg Sander. 2013. Density-Based Clustering Based on Hierarchical Density Estimates. In *Pacific-Asia conference on knowledge discovery and data mining*, volume 7819, pages 160–172, Gold Coast, Australia. Springer.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Evelyn Gius, Jan Christoph Meister, Malte Meister, Marco Petris, Mareike Schumacher, and Dominik Gerstorfer. 2023. *CATMA 7 (Version 7.0)*. Zenodo.

Alberto González Martínez, Billy Troy Wooton, Nurit Kirshenbaum, Dylan Kobayashi, and Jason Leigh. 2020. Exploring collections of research publications with human steerable ai. In *Practice and Experience in Advanced Research Computing*, PEARC '20, page 339–348, New York, NY, USA. Association for Computing Machinery.

Maarten Grootendorst. 2022. BERTopic: Neural topic modeling with a class-based TF-IDF procedure. *arXiv preprint arXiv:2203.05794*.

Jan-Christoph Klie, Michael Bugert, Beto Boullosa, Richard Eckart de Castilho, and Iryna Gurevych. 2018. The INCEpTION Platform: Machine-Assisted and Knowledge-Oriented Interactive Annotation. In *Proceedings of the 27th International Conference on Computational Linguistics: System Demonstrations*, pages 5–9, Santa Fe, New Mexico. Association for Computational Linguistics.

Jingjing Liu, Panupong Pasupat, Yining Wang, Scott Cyphers, and Jim Glass. 2013. Query understanding enhanced by hierarchical parsing structures. In *IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 72–77, Olomouc, Czech Republic. IEEE.

Leland McInnes, John Healy, and James Melville. 2020. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *arXiv preprint arXiv:1802.03426*.

Mariana Neves and Jurica Ševa. 2019. An extensive review of tools for manual annotation of documents. *Briefings in Bioinformatics*, 22(1):146–163.

Justin Payan, Yuval Merhav, He Xie, Satyapriya Krishna, Anil Ramakrishna, Mukund Sridhar, and Rahul Gupta. 2021. Towards Realistic Single-Task Continuous Learning Research for NER. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3773–3783, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Dominik Sacha, Leishi Zhang, Michael Sedlmair, John A. Lee, Jaakko Peltonen, Daniel Weiskopf, Stephen C. North, and Daniel A. Keim. 2017. Visual Interaction with Dimensionality Reduction: A Structured Literature Analysis. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):241–250.

Tim Sainburg, Leland McInnes, and Timothy Q Gentner. 2021. Parametric UMAP embeddings for representation and semi-supervised learning. *arXiv preprint arXiv:2009.12981*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. *CoRR*, cs.CL/0306050.

Daniel Smilkov, Nikhil Thorat, Charles Nicholson, Emily Reif, Fernanda B. Viégas, and Martin Wattenberg. 2016. Embedding Projector: Interactive Visualization and Interpretation of Embeddings. *arXiv preprint arXiv:1611.05469*.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. brat: a Web-based Tool for NLP-Assisted Text Annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107, Avignon, France. Association for Computational Linguistics.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. 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*, pages 142–147, Edmonton, Canada. Association for Computational Linguistics.

Seid Muhie Yimam, Chris Biemann, Richard Eckart de Castilho, and Iryna Gurevych. 2014. Automatic Annotation Suggestions and Custom Annotation Layers in WebAnno. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 91–96, Baltimore, Maryland. Association for Computational Linguistics.

Liu Zhuang, Lin Wayne, Shi Ya, and Zhao Jun. 2021. A robustly optimized BERT pre-training approach with post-training. In *Proceedings of the 20th Chinese National Conference on Computational Linguistics*, pages 1218–1227, Huhhot, China. Chinese Information Processing Society of China.