From Qualitative to Quantitative Research: Semi-Automatic Annotation Scaling in the Digital Humanities

Fynn Petersen-Frey* **Tim Fischer*** **Florian Schneider***
Isabel Eiser† **Gertraud Koch†** **Chris Biemann***
*Language Technology Group, Department of Informatics, Universität Hamburg
†Institute for Anthropological Studies in Culture and History, Universität Hamburg
{first.last}@uni-hamburg.de, florian.schneider-1@uni-hamburg.de

Abstract

In today's digital era, massive amounts of data are ubiquitous including discourses in natural language, such as news articles, social media posts or forum threads. The digital humanities aim to qualitatively and quantitatively analyze such data. For interpretive research, it is difficult to benefit from large data. An example is grounded theory, an interpretative method to deal with larger datasets by annotating or coding. However, such approaches are too time-consuming to bridge the gap from qualitative to quantitative analyses. In this work, we propose assistive methods to semi-automatically scale a small number of manual annotations to large corpora. Our approach uses contextualized embeddings of annotated data to find similar occurrences. By interactively providing suggestions learned automatically from user interactions, our method provides a convenient and fast way to annotate large corpora with minimal manual effort. The method finally produces a classifier able to annotate the entire dataset. We performed experiments on multiple tasks and datasets to evaluate our methods demonstrating strong performance. Further, we designed a software for researchers who want to scale their annotation-based research, bridging the gap from qualitative to quantitative results.

1 Introduction

There is a growing interest in the Digital Humanities (DH) to apply Natural Language Processing (NLP) methods to explore textual data and scale textual data analysis. The reason for this is twofold. First, due to the advancing digitization of humanities and cultural studies data, both through retro-digitization and the increase in born digital data, large quantities of data are available that are often infeasible for a single person or team to study. Second, the groundbreaking success of NLP in various disciplines makes it attractive to adapt methods to the DH domain. This is a great opportunity for qualitative DH researchers to benefit from large datasets where in-depth qualitative analysis and annotations cannot be extended to large-scale corpora.

The Digital Humanities often rely on qualitative methodology like grounded theory. Hermeneutic circular processes and theoretical sampling approaches include iterative search, selection, collection, analysis, and interpretation of research data. Grounded theory can be understood as an interactive process where researchers, participants and data construct research together in interaction repeatedly, producing a category system that effectively captures the research problem. It is of great interest to apply the category system to larger datasets for quantitative analysis, but infeasible to do so manually.

Currently, data scientists would need to train a Machine Learning (ML) model requiring large amounts of training data that have to be created by qualified annotators using to-be-developed annotation guidelines. This is time-consuming, costly, requires ML expertise, and is consequently rarely done in the context of DH projects. Thus, new methods combining human and computer actions are needed to enable research on larger datasets and foster further research in the digital humanities as typical ML approaches are no good fit for most projects. Recent studies (Ostheimer et al., 2021; Koch et al., 2022) in the field of human-computer-interaction have shown fruitful incorporation of human decision-making in ML processes and that human-in-the-loop methods can significantly improve ML models. ML-supported annotation needs the human supervision and refinement to offer useful and accurate alternatives in qualitative data analysis. Strengthening the synergy between humans and machines is a promising direction where both sides are profiting: ML-based annotation is improved by human refinements, whereas automation aids iterative processes of human meaning-making and interpretive research by scaling annotations to enable the analysis of vast materials.

Proceedings of the 19th Conference on Natural Language Processing (KONVENS 2023), pages 52–62
September 18–22, 2023.

This work targets qualitative researchers who annotate textual data to analyse it in-depth and want to increase their efficiency and/or want to leverage large datasets as quantitative grounding for their hypothesises. We propose an ML-based assistive system leveraging current NLP to ease the annotation task and semi-automatically scale annotations from few manual annotations to a fully-annotated corpus. After the user has annotated a few text spans with their categories, representatives for each category are utilised for semantic similarity search to suggest relevant text spans with their context. While the user accepts or discard these suggestions, the system adapts to feedback by updating the category representatives instantly after each verified suggestion. Since verifying suggestions is a much faster task than reading and annotating, users can efficiently annotate their documents. The system automatically fine-tunes models to predict higher-quality suggestions and to apply the learned categories to a large document collection with high accuracy, thereby scaling the annotations.

In this paper, we make several contributions towards a system supporting researchers during and after their qualitative analysis and aids them in scaling their annotations to large corpora: (1) A two-stage method usable without programming or NLP know-how to semi-automatically scale annotations to large-scale corpora by interactively providing adaptive suggestions and employing adapter (Houlsby et al., 2019) technology to automatically annotate large corpora. (2) A user interface for quick batch validation of suggestions while still displaying the most relevant contextual information. (3) An evaluation of our method with a simulated annotation process on multiple large datasets for sentence-level and word-level annotations demonstrating strong scaling capabilities.

2 Related work

Qualitative analyses can be powerfully supported with digital solutions and ML methods addressing annotation, analysis, and interpretation. MAXQDA and ATLAS.ti are two commonly used closed-source, paid solutions for qualitative analysis trying to offer all-in-one-solutions, but include no ML assistance. Prodigy is an annotation software where the workflow is dictated by the active learning (AL) model. Label studio is an annotation platform that offers AL functionalities requiring set-up by connecting external models, thereby making it unsuitable for domain experts. Existing open-source software that offers (semi-)automatic annotation aid in various variations are outlined in the following.

WebAnno (Yimam et al., 2014; Eckart de Castilho et al., 2016) is a web-based tool for fine-grained NLP annotations and includes an automatic method where the system learns from user-provided annotations. However, it requires expert ML knowledge to perform feature engineering and training. The successor INCEpALTION (Klie et al., 2018) can suggest possible labels and includes an AL mode to guide annotators to improve the system by labeling examples providing valuable information to the classifier. Neither WebAnno nor INCEpTION are designed to work on a content-level or on large-scale corpora. CodeAnno (Schneider et al., 2023b) is another WebAnno successor focusing on document-level coding and supports training ML classifiers for this. LabelSleuth (Shnarch et al., 2022) is a software to build binary classifiers by labeling text data using active learning suggestions targeted at domain experts. While it is probably the closest to our application, it only supports binary classification of sentences, unsuitable for multi-class word-level annotations.

Active learning (AL) is a technique to obtain a classifier by soliciting feedback from the user on the most informative sample identified using e.g. uncertainty sampling (Lewis and Gale, 1994) which requires a classifier to output certainty scores. AL is often associated with the Human-in-the-loop (Holzinger, 2016) paradigm where human feedback is integrated in the loop of machine learning development. We see our work as AI-in-the-loop where ML systems assist the workflow of humans who stay in control all the time. Our system assists the user by providing sensible suggestions, but neither disrupts nor dictates their workflow.

Few-shot classification of named entities is a task relevant to our annotation scaling scenario where a classifier is trained to generalize to new classes after observing only a small number of examples. This task was tackled (Fritzler et al., 2019) by using prototypical networks (Snell et al., 2017) which learn a prototype for unseen classes by averaging the representations of the support samples for that class. However, this approach is not made for incrementally increasing samples and it does not scale well with increasing numbers of examples. Our approach utilizes a few-shot classification system based on adapters (Houlsby et al., 2019) to provide

a high-quality classifier from few training samples.

Previous work (Remus et al., 2022) evaluated different strategies to find related items in an information retrieval scenario demonstrating that contextualized word embeddings of pre-trained models are suitable to retrieve word-level items such as named entities. They showed a small speedup of manual annotations when annotating similar instead of random items. We build upon these findings and devised a method to scale manual annotations to datasets orders of magnitude larger.

3 Application

In this work, we describe a software for qualitative annotation of text documents that assists users during their annotation process with ML functionality requiring neither programming nor ML knowledge. The assistance comes in two forms: Providing suggestions for semi-automatic annotation and fully automatic annotation to analyse large text collections. Our methods apply to sentence-level (e.g. headlines, arbitrary sentences) and token-level (e.g. named entities) annotations. Document-level annotations are not in the scope of this work. Further, our methods are not intended for a MATTER annotation process (see Pustejovsky and Stubbs (2012)) but for use in hermeneutic contexts where users move back-and-forth between understanding parts of a text and the whole (the so-called 'hermeneutic circle'), continuously build and modify their tagset resp. labels along the way (see Horstmann (2019)).

3.1 Requirements

Throughout our close collaboration between Digital Humanities and Computational Linguistics groups, we identified four essential requirements for our system (method and user interface) to provide the most benefit to the users: (1) *Usable without machine learning knowledge*. This enables all researchers to benefit from the method. (2) *Applicable from small to large-scale document collections containing thousands of documents*. This allows to use the method for a wide range of research questions and datasets. (3) *Instantaneous responses & quick adaptation to the users feedback*. This greatly improves the user experience. (4) *Correction of mistakes*. Users must be able to correct system errors and preventing similar mistakes to achieve their desired outcome which has been found highly important for interactive systems like *new/s/leak* (Wiedemann et al., 2018).

3.2 Workflow

Imagine Alice, a DH researcher working on a climate project, who needs to identify, categorise and quantify different actors and stakeholders involved in the parliamentary discussions about climate change to answer one of her research questions. She downloaded a large collection of all plenary minutes and printed matter of the German Bundestag which is available as open data. Alice creates a new project in the software and adds the crawled documents.

Early guidance She works with the software as always: Finding relevant documents by using the built-in search and filtering methods, reading documents, making spontaneous annotations (possibly creating new classes as necessary in this hermeneutic process) while reading and selecting some documents to annotate in detail. During close reading, the system already suggests and highlights text passages in the current document based on the annotated material if she enabled the feature (see Figure 1, left). Her annotations are quite diverse: Some annotations are named entities coded as actor or stakeholder, possibly hierarchically with fine-grained classes (politician, scientist, activist etc.) while others refer to different entities or entire sentences like public statements. After annotating a handful of documents, she has seen enough different cases of actors and stakeholders.

Semi-automatic annotation scaling She enables the suggestion panel (see Figure 1, right) to explore annotation suggestions in the text collection. The system provides a list of suggestions showing the context of each annotation, i.e. sentence(s) and the document title with a link to the full document in case it's needed for verification. Alice can accept or discard suggestions which are then automatically persisted as annotations in her project.

Fully-automatic annotation for quantitative analysis The system indicates to Alice that enough examples have been labeled and a reliable model is trained to apply her established category system to all documents. She double checks a random selection of these automatic annotations and decides that the quality is high enough for further quantitative analyses. In case she identifies erroneous annotations, she can easily update or remove such cases. The system adapts automatically and potentially fixes similar errors. Alice can iterate and correct the system as often as necessary. To

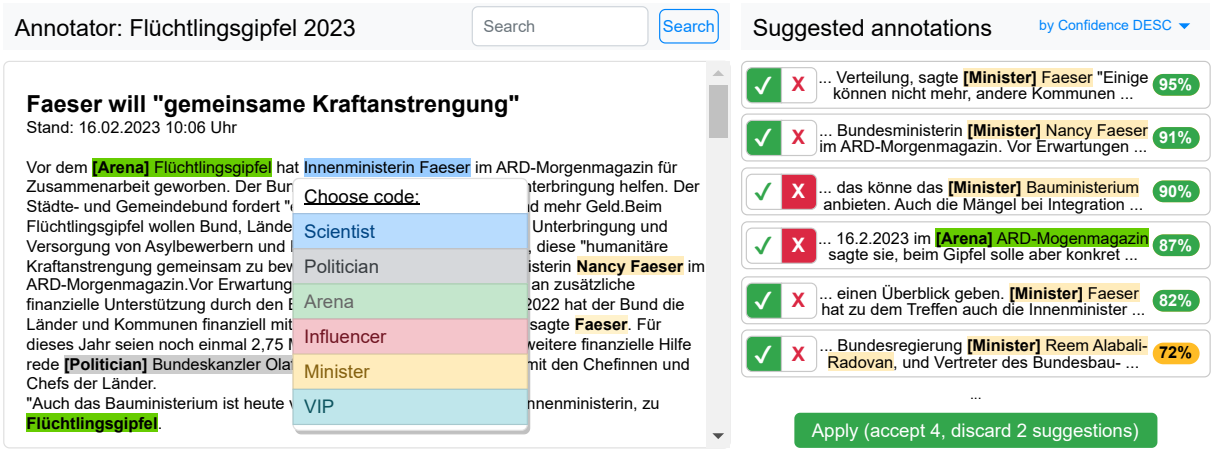

Figure 1: Annotation interface. Left: Document with automatic suggestions enabled. Highlighted texts prefixed with [code] are manually created annotations, other highlights are system suggestions. Right: Batch approving/discarding suggestions to semi-automatically scale annotations.

improve the quality, multiple users can annotate the same documents so Alice can curate annotations. Finally, she retrieves the statistics on the whole corpus (see Figure 2) such as frequency of politicians, scientists etc. or a list of ministers sorted by their frequency in the corpus. By exporting her manual and automatically generated annotations, she can use her favorite analysis and visualization tools to draw further conclusions about her material.

4 Methodology

In this section, we explain our approach to scale few manual annotations with minimal effort to large datasets. First, we perform a one-time process to generate contextualized embeddings for the document collection. Second, we provide interactive suggestions based on contextual embeddings of manual annotations and customized similarity computations. Third, when enough annotations are collected, a classifier is created and applied to the entire dataset. Optionally, the user refines the classifier by correcting aggregated results.

Pre-processing All documents added to the system run through an initial, one-time pre-processing phase: Apache Tika extracts plain text from any document. Sentence-splitting and entity recognition are performed by spaCy (Honnibal et al., 2020). Contextualized embeddings are computed using SBERT (Reimers and Gurevych, 2019) models for sentences, T-NER (Ushio and Camacho-Collados, 2021) for named entities and RoBERTa (Liu et al., 2019) for other structures (see Section 4.1). Multilingual models (e.g. LaBSE (Feng et al., 2022) for sentences, XLM-RoBERTa (Conneau et al., 2020) for tokens) can be used to apply our approach to different languages. The embeddings of each structure are stored in an approximate nearest neighbor (ANN) index like HNSW (Malkov and Yashunin, 2018) or FAISS (Johnson et al., 2021) to enable fast retrieval of similar embeddings for large datasets. The pre-processing is executed automatically in the background before interactive use, satisfying requirements 1–3 (4 is not applicable). Pre-computing contextualized embeddings of sentences and tokens for the document collection enables instantaneous suggestions.

4.1 Interactive, semi-automatic annotation scaling

The system performs the following steps to produce k suggestions for a class c.

Structure selection The system automatically detects which structure fits annotations of class c best by computing and comparing the overlap of annotations with all known structures computed during pre-processing (e.g. sentences, named-entities, noun chunks, single tokens or n-grams). This is a fast database operation (< 10 milliseconds) as only offsets for a few manual annotations need to be compared. The structure s with the highest overlap on average is chosen for the next step.

Candidate retrieval The embeddings of the structures matching each annotation (positive support set, size n) are used to search for the most similar embeddings using cosine similarity in the

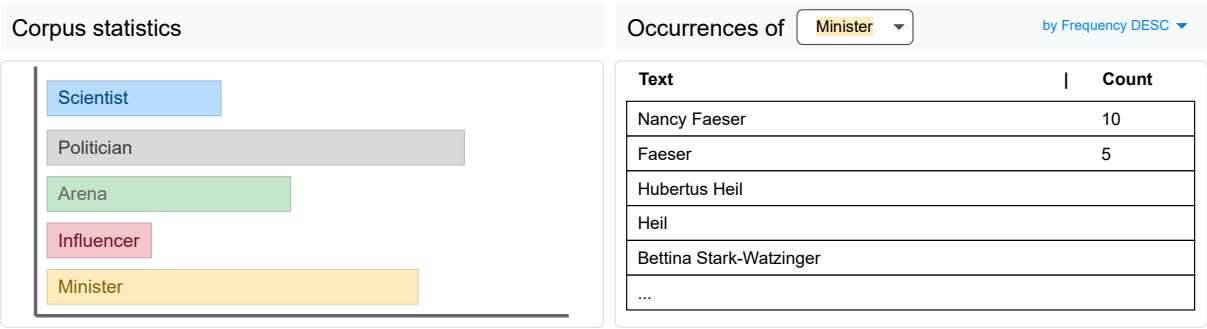

Figure 2: Quantitative analysis interface. Left: Estimation of the frequency of annotated categories in the entire corpus. Right: List of all annotations of a selected category.

ANN index. To do so efficiently, a batch query is performed on the index to retrieve a set of candidates. In total, $N > n$ similar embeddings and their corresponding text spans are retrieved from the ANN index.

Candidate filtering Any already annotated candidates are excluded. The remaining candidates are further filtered by removing a candidate if it is nearest to another sample having a class different than c (negative support set). This *advanced* approach often results in large quality improvements (see Section 5.2 for a comparison). The *naive* approach does not filter using a negative support set. A batch query returning only the most similar item is used to to so efficiently. The remaining candidates are re-ranked by their maximum similarity to any annotated sample of class c. Finally, the top candidates are returned to the user as suggestions.

Reviewing suggestions The user can accept/discard the automatic suggestions batch-wise and accept/discard/edit individual ones. Accepted and edited suggestions are stored as annotations in the database. Rejected suggestions are assigned a hidden 'not-c' class and also stored in the database. The suggestions iteratively improve with every use as more samples of the class c become available as positive support and more samples of other classes become available as negative support.

Requirement check Our method fulfills all requirements (see Section 3.1): Requirement (1) is achieved since no configuration is required. Requirement (2) and (3) are fulfilled as the search in the ANN index scales to billions of embeddings and returns results in a few milliseconds regardless of the collection size. In accordance with requirement (4), users can edit/discard suggestions which automatically affects future suggestions.

4.2 Fully-automatic annotation scaling for quantitative analysis

While the interactive mode allows to quickly annotate hundreds of examples, a real classifier is needed to obtain quantitative results on large datasets. To build a classifier, enough negative samples are required beside the positive samples. If there are fewer negative samples, negative candidates are randomly sampled from all unannotated items (the number of positive samples is usually small compared to all samples). To guard against accidentally choosing a positive sample, candidates are removed if their nearest embedding neighbor is belonging to a positive class. This strategy is more beneficial than selecting negative samples by taking the nearest neighbors of known items as it would not produce a diverse negative support set.

A k-nearest neighbor (KNN) classifier is constructed from the positive and negative support set. To annotate the entire dataset with this classifier, each unannotated structure s is compared to the support set containing all positive and negative samples. This is efficiently done by computing a single matrix multiplication (for cosine similarity) between the support set and all unannotated items. We considered four different strategies to make the final prediction: (1) *Nearest*: The class of the nearest neighbor is chosen as the prediction. (2) *Centroid*: The positive and negative support set are each averaged to centroid embedding. The class of whichever centroid is closer is chosen. This approach is similar to prototypical networks and computationally highly efficient, but often lacks quality. (3) *Majority voting*: The most frequent class within the k nearest neighbors is chosen as the prediction. (4) *Weighted majority voting*: The similarities of each class of the nearest k neighbors are added and the highest is chosen.

This classification is a fast and quickly adaptable approach since no training is required. Applying the classifier on the corpus allows to count how many annotations of a specific class are in the document collection. A list of all potentially annotated text spans can be produced, merged by the same surface text (e.g. all politicians with the same name), and sorted by their frequency. The user can correct the output by assigning a different label to each aggregated group of annotations, thereby immediately improving the KNN classifier to return updated results in less than a second. While a KNN classifier is fast and and adapts quickly, it leaves room for higher-quality predictions. Thus, we experiment with training a stronger classifier in Section 5.3 with few annotated samples.

5 Evaluation

To evaluate our proposed methods, we apply them on fully annotated datasets. This allows us to compare our methods outputs with the correct (as in human created) annotations. We perform three evaluations: (1) We simulate how a user would use the system interactively and evaluate the quality of the automatic suggestions. In this setting, the goal is to find as many annotations of the desired class with the same manual effort (i.e. number of verified suggestions). For named-entity datasets, we directly use the given entity span offsets (instead of first applying an entity detection model). We use the total number of successfully annotated samples as metric. (2) We evaluate the performance of the final KNN-based classifier created after the simulation with the macro F1 score (harmonic mean of precision and recall). (3) We evaluate how many annotated samples are needed to train an even stronger classifier. In these experiments, we train a neural end-to-end classifier with increasing amounts of training data and report the F1 scores.

5.1 Datasets

In this section, we introduce the datasets and data generation processes used in our experiments. An overview of the datasets is provided in Table 1.

OntoNotes5.0 (Weischedel et al., 2013) (ON5) is a well-known named-entity recognition (NER) dataset of 76,714 samples split into 59,924 *train*, 8,262 *test* and 8,528 *validation* samples. Each sample is a sentence in which each word is labeled as one of 18 classes or the other class `O`. We created a custom version of ON5 with the same number of samples and splits as the original dataset but only 12 of the 18 classes. Precisely, we removed `DATE`, `TIME`, `GPE`, `ORG`, `ORDINAL`, and `WORK_OF_ART` from ON5 by replacing the respective tags with the `O` tag. Following (Ding et al., 2021), we merged IOB tags (Ramshaw and Marcus, 1999) into a single tag to ease the few-shot episode data generation.

The MIT Movie Trivia (MITMT) and MIT Restaurant (MITR) datasets[1] are named-entity recognition datasets containing 6,816 *train*, 1,953 *test*, and 1,000 *validation* samples and 6,900 *train*, 1,521 *test*, and 760 *validation* samples. They contain domain-specific named entity classes like *Actor* and *Genre* or *Cuisine* and *Dish*. We created custom versions of the datasets with merged IOB tags, same splits and number of samples containing only classes of at least 1,000 samples. In the custom MITMT, we removed the classes *Award*, *Quote*, *Soundtrack*, *Relationship*, *Origin*, *Opinion* and *Character* and in the custom MITR, we removed *Rating*, *Hours*, and *Price*.

Yahoo! Answers (YA) (Zhang et al., 2015) is a topic classification dataset that includes 4.5 million questions and answers from 10 different categories. We divide the original test set (60,000 items) in half to obtain a validation/test split from which we use the title and its category for our experiments.

5.2 Semi-automatic annotation scaling

We simulate the usage of the system on our custom ON5 and YA dataset with the following strategy. For each of the unseen classes, we randomly select 20 samples (manual annotation) as the initial positive support set from the validation split and iteratively retrieve ten times 20 suggestions that are accepted/rejected with the human-annotated labels from the dataset. Finally, we build a multi-class KNN classifier from the collected samples and classify the test split. We compare the two approaches explained in Section 4.1 to provide suggestions: *Naive* and *advanced* (using a negative support set). For the KNN classifier, we compare four variants nearest neighbor, majority voting ($k = 5$), weighted majority voting ($k = 5$) and centroids. For ON5, we use our named-entity model trained only on 12 out of 18 classes (see Section 5.3 for details) to produce entity embeddings (first token of each entity). To obtain sentence embeddings for YA, we use the pre-trained SBERT model `all-MiniLM-L12-v2`.

[1] see `https://groups.csail.mit.edu/sls/downloads`

Dataset	Type	Size	Classes
OntoNotes 5.0	NER	76,714	CARDINAL, DATE, EVENT, FAC, GPE, LANGUAGE, LAW, LOC, MONEY, NORP, ORDINAL, ORG, PERCENT, PERSON, PRODUCT, QUANTITY, TIME, WORK_OF_ART
MIT Movie Trivia	NER	9,769	Actor, Award, Character, Director, Genre, Opinion, Origin, Plot, Quote, Relationship, Soundtrack, Year
MIT Restaurants	NER	9,181	Amenity, Cuisine, Dish, Hours, Location, Price, Rating, Restaurant_Name
Yahoo! Answers	SC	60,000	Society & Culture, Food & Drink, Cars & Transportation, Education & Reference, Science Mathematics, Business & Finance, News & Events, Computers & Internet, Pets, Politics & Government

Table 1: Overview of the datasets used in the experiments. In the Type column, SC is short for sentence classification.

			(macro) F1 score			
Class	**Approach**	**Samples**	**nearest**	**majority**	**weighted**	**centroid**
		OntoNotes5.0				
DATE	naive	161	0.92	**0.93**	**0.93**	0.77
DATE	advanced	190	0.89	0.91	0.90	0.67
WORK_OF_ART	naive	45	0.56	0.61	0.61	0.43
WORK_OF_ART	advanced	89	0.65	**0.71**	**0.71**	0.42
ORDINAL	naive	69	**0.91**	0.90	0.89	0.43
ORDINAL	advanced	148	0.88	0.88	0.88	0.37
GPE	naive	200	0.93	0.94	0.94	0.89
GPE	advanced	200	0.93	0.94	**0.95**	0.89
TIME	naive	27	0.60	0.62	0.62	0.36
TIME	advanced	99	0.60	**0.63**	0.62	0.33
ORG	naive	178	0.86	0.87	0.87	0.87
ORG	advanced	197	**0.89**	**0.89**	**0.89**	**0.89**
all six unseen classes	naive	680	0.83	0.83	0.83	0.65
all six unseen classes	advanced	923	0.83	**0.85**	**0.85**	0.62
		Yahoo! Answers				
all ten classes	naive	662	0.50	0.51	0.52	0.53
all ten classes	advanced	1,497	0.56	**0.59**	**0.59**	0.58

Table 2: Annotation simulation & KNN classification results. Samples are the number of correct suggestions.

The results of the experiments are shown in Table 2. While both approaches are able to provide mostly correct suggestions for DATE, GPE and ORG in OntoNotes, the naive approach has a high error rate for the remaining three classes. The advanced approach provides more than twice the samples for the difficult classes (which are rarer and more overlapping, see Figure 3) and reaches an average suggestion precision of 76.9%. On Yahoo! Answers, the advanced approach provides more than twice the number of correct suggestions of the naive approach resulting in an average hit ratio of 75%.

On OntoNotes, the KNN classifier produces very strong F1 scores (≈ 0.9) except for the TIME and WORK_OF_ART classes having the fewest samples. While there is only a small performance difference between nearest and (weighted) majority voting KNN variants, the centroid performs worse by a noticeable margin. The performance on Yahoo! Answers reach scores of 0.59 with the advanced approach widely outperforming a random baseline of 0.1. We attribute this comparatively lower level to fact that the category in the dataset is not only dependent on the title but also the question text which we did not leverage in our experiments.

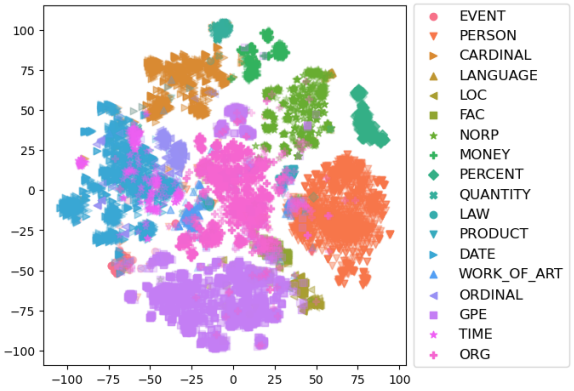

Figure 3: t-SNE Entity embeddings generated on the ON5 test split using our custom trained T-NER model.

We further analyzed the reasons for the behavior of the suggestions and KNN classifier with a visualization of the entity embeddings in Figure 3. Most instances of a class are visually clustered together, even for the the six unseen classes. However, `DATE` and `TIME` are mixed together and explain the lower scores for `TIME` as it is the rarer occurring class. Overall, the visualization shows that the use of embedding similarity metrics for suggestions and classification can work well and should also be able to distinguish between sub-classes as many of the clusters consist of smaller clusters. We also measured the run-time efficiency of our approach: Generating all suggestions and classifying every entity in the test splits for all experiments took only 0.7 seconds in total.

5.3 Few-Shot Transformer Adapter Classifier

With this experiment, we evaluate how many samples are necessary to train a classifier with superior performance to the KNN classifier. The user can instruct the system to (re-)train such a classifier whenever it makes sense, e.g. after annotating a bunch of samples of a new class. A straightforward approach to creating a NER or sentence classifier is to fine-tune a pre-trained large language model (LLM). Due to its numbers of parameters, it would require lots of training samples and be computationally expensive. Since labeled data is scarce in our scenario and a KNN classifier does not need to be trained at all, we train a classifier with as little training data and computational effort as possible. One approach successfully applied in various tasks is training a transformer adapter (Houlsby et al., 2019) using the convenient AdapterHub framework (Pfeiffer et al., 2020). Transformer adapters are a computation- and sample-efficient alternative to full fine-tuning. During training, the LLM's original parameters are frozen, and only the adapter layers are optimized. While different configurations or variants of adapters exist, the parallel configuration (He et al., 2022) outperformed others by a large margin in our preliminary experiments. A new classification head with output dimension equal to the number of classes is trained jointly with the adapter layers.

We used the two setups and the three NER datasets described in Section 5.1 to evaluate how many training samples are required to train an adapter classifier. In the first setup, the ON5 dataset is utilized as follows: We fine-tuned a `roberta-base` model on the train split of the custom ON5 that contains 12 of 18 classes. Next, we injected adapters in the fine-tuned model and trained on episodes containing all 18 classes of ON5. An episode is a set of training samples where every class has between K and $2K$ instances. An episode for $K = 1$ consists of N training samples so that each class is represented at least once and at most twice in all N sentences. In the second setup, we fine-tuned a `roberta-base` model on the train split of the original ON5. Then, we injected adapters and trained on episodes from the customized versions of the MIT Movie Trivia and MIT Restaurant datasets.

In both setups, we trained the adapters with samples of the training split for 5 epochs on episodes for $K \in \{3, 5, 10, 30, 50, 100, 300, 500, 1000\}$ and evaluated on all samples in the test splits. We used the default training hyperparameters from the AdapterHub framework.

As can be observed from the results reported in Figure 4, adapter classifiers demonstrate strong performance (> 0.8 F1) already for small episodes with only 30 to 100 training instances per class. Our adapter models perform similar regardless of the dataset. This further endorses the choice of adapters for robust few-shot NER classifiers. Visible in the left plot in Figure 4 is that adapters are suitable when an existing NER model is extended. The model remembers classes learned in the fine-tuning phase and quickly adapts to new classes.

6 Conclusion

In this work, we proposed and evaluated methods to semi-automatically scale few annotations to large corpora by providing interactive suggestions from adaptable classifiers and developed user interfaces to make our methods usable for domain experts

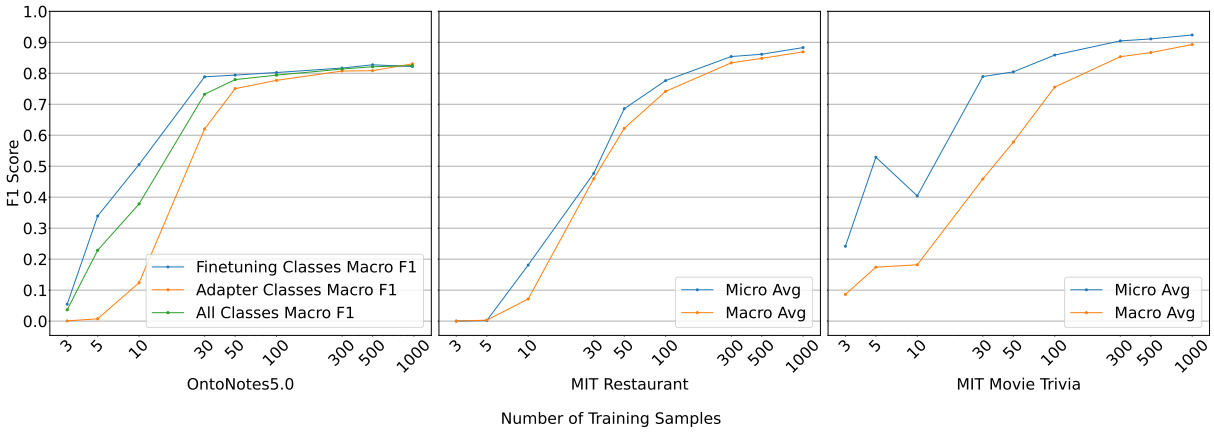

Figure 4: Few-Shot NER performance of transformer adapter classifiers with increasing training data

without programming or NLP skills. Our evaluation on existing datasets shows that these methods can quickly scale annotations with minimal manual effort to large corpora to both obtain quantitative results and aggregated lists enabling a verification of the automated processing. Thus, we see our methods aiding qualitative researchers to bridge the gap to quantitative results and providing quantitative grounding for their hypothesises.

In the future, we want to integrate the developed methods and user interfaces in production-grade code quality into our D-WISE Tool Suite (Schneider et al., 2023a), an open-source web application for digital qualitative discourse analysis in the Digital Humanities. In doing so, we make our developed methods easily accessible to other researchers and plan to further improve the methods by incorporating more feedback. As our methods are also applicable to image or video annotations via object detection, we might explore to adapt them for a good user experience.

Limitations & Ethics Statement

Our work makes NLP models and methods accessible to researchers that could previously not benefit from these advances. Our work targets Digital Humanities researchers and is intended to assist with qualitative discourse analysis. As with any ML-based method, though, it could somehow be misused for other, possibly inappropriate, work. We strongly believe that including and enabling more researchers to benefit from modern ML technology outweighs the potential for misuse.

When using ML models, it is important to understand their limitations and critically reflect on their predictions. ML models often include certain biases that can manifest in various types and forms and are certainly not without error.

Especially the proposed fully-automatic annotation scaling has to be used critically. We developed this method for researchers to easily obtain quantitative insights on the whole dataset, however, the results are most likely not comparable to a careful, manual or semi-automatic analysis of the full material. Instead, they should be understood as an estimation and help to quantitatively verify hypotheses that emerged from the qualitative analysis. We try to mitigate the issues of the fully-automatic annotation scaling by providing confidence scores where applicable, showing aggregated classification results and allowing the user to correct system mistakes. To evaluate the quality of the automatic annotations, a user can manually annotate a random subset and compare this with the automatic results.

We also possibly introduce a bias with our method design and envisioned workflow. While we tried our best to give the user the freedom when and how to use the automated components, we might still restrict a user in their workflow by our design decisions. It is important to note that the workflow was described in a way to best highlight the contributions of this paper. This not the only way to use the described methods. Instead, the proposed methods are intended to be used in addition to established qualitative methods or to augment them, but not to replace them entirely.

Acknowledgement

This work is supported by the D-WISE project, Universität Hamburg, funded by BMBF (grant no. 01UG2124).

References

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Ning Ding, Guangwei Xu, Yulin Chen, Xiaobin Wang, Xu Han, Pengjun Xie, Haitao Zheng, and Zhiyuan Liu. 2021. Few-NERD: A Few-shot Named Entity Recognition Dataset. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3198–3213, Online.

Richard Eckart de Castilho, Éva Mújdricza-Maydt, Seid Muhie Yimam, Silvana Hartmann, Iryna Gurevych, Anette Frank, and Chris Biemann. 2016. A Web-based Tool for the Integrated Annotation of Semantic and Syntactic Structures. In *Proceedings of the Workshop on Language Technology Resources and Tools for Digital Humanities (LT4DH)*, pages 76–84, Osaka, Japan.

Fangxiaoyu Feng, Yinfei Yang, Daniel Cer, Naveen Arivazhagan, and Wei Wang. 2022. Language-agnostic BERT sentence embedding. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 878–891, Dublin, Ireland. Association for Computational Linguistics.

Alexander Fritzler, Varvara Logacheva, and Maksim Kretov. 2019. Few-shot classification in named entity recognition task. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, pages 993–1000.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2022. Towards a Unified View of Parameter-Efficient Transfer Learning. In *International Conference on Learning Representations*, Online.

Andreas Holzinger. 2016. Interactive Machine Learning for Health Informatics: When Do We Need the Human-In-the-Loop? *Brain Informatics*, 3(2):119–131.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python. https://spacy.io/.

Jan Horstmann. 2019. Theory. https://catma.de/philosophy/theory/. In *CATMA*. Last accessed: 4 May 2023.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-Efficient Transfer Learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 2790–2799, Long Beach, CA, USA.

J. Johnson, M. Douze, and H. Jegou. 2021. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(03):535–547.

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.

Gertraud Koch, Chris Biemann, Isabel Eiser, Tim Fischer, Florian Schneider, Teresa Stumpf, and Alejandra Tijerina García. 2022. D-WISE Tool Suite for the Sociology of Knowledge Approach to Discourse. In *Culture and Computing*, pages 68–83, Cham.

David D. Lewis and William A. Gale. 1994. A Sequential Algorithm for Training Text Classifiers. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 3–12, Berlin, Heidelberg.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Yu A Malkov and Dmitry A Yashunin. 2018. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(4):824–836.

Julia Ostheimer, Soumitra Chowdhury, and Sarfraz Iqbal. 2021. An alliance of humans and machines for machine learning: Hybrid intelligent systems and their design principles. *Technology in Society*, 66:101647.

Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020. AdapterHub: A Framework for Adapting Transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 46–54, Online.

James Pustejovsky and Amber Stubbs. 2012. *Natural Language Annotation for Machine Learning: A guide to corpus-building for applications*. " O'Reilly Media, Inc.".

Lance A Ramshaw and Mitchell P Marcus. 1999. Text Chunking Using Transformation-based Learning. *Natural language processing using very large corpora*, pages 157–176.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*.

Steffen Remus, Gregor Wiedemann, Saba Anwar, Fynn Petersen-Frey, Seid Muhie Yimam, and Chris Biemann. 2022. More like this: Semantic retrieval with linguistic information. In *Proceedings of the 18th Conference on Natural Language Processing (KONVENS 2022)*, pages 156–166, Potsdam, Germany. KONVENS 2022 Organizers.

Florian Schneider, Tim Fischer, Fynn Petersen-Frey, Isabel Eiser, Gertraud Koch, and Chris Biemann. 2023a. The D-WISE tool suite: Multi-modal machine-learning-powered tools supporting and enhancing digital discourse analysis. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 328–335, Toronto, Canada. Association for Computational Linguistics.

Florian Schneider, Seid Muhie Yiman, Fynn Petersen-Frey, Gerret von Nordheim, Katharina Kleinen-von Königslöw, and Chris Biemann. 2023b. CodeAnno: Extending WebAnno with Hierarchical Document Level Annotation and Automation. In *The 17th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2023), System Demonstrations*, Dubrovnik, Croatia.

Eyal Shnarch, Alon Halfon, Ariel Gera, Marina Danilevsky, Yannis Katsis, Leshem Choshen, Martin Santillan Cooper, Dina Epelboim, Zheng Zhang, Dakuo Wang, Lucy Yip, Liat Ein-Dor, Lena Dankin, Ilya Shnayderman, Ranit Aharonov, Yunyao Li, Naftali Liberman, Philip Levin Slesarev, Gwilym Newton, Shila Ofek-Koifman, Noam Slonim, and Yoav Katz. 2022. Label Sleuth: From Unlabeled Text to a Classifier in a Few Hours. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical Networks for Few-shot Learning. In *Advances in Neural Information Processing Systems*, volume 30.

Asahi Ushio and Jose Camacho-Collados. 2021. T-NER: An all-round python library for transformer-based named entity recognition. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 53–62, Online. Association for Computational Linguistics.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, Mohammed El-Bachouti, Robert Belvin, and Ann Houston. 2013. OntoNotes Release 5.0. *LDC2013T19, Philadelphia, Penn.: Linguistic Data Consortium*.

Gregor Wiedemann, Seid Muhie Yimam, and Chris Biemann. 2018. New/s/leak 2.0 – Multilingual Information Extraction and Visualization for Investigative Journalism. In *Social Informatics*, pages 313–322, Cham.

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.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level Convolutional Networks for Text Classification. In *Advances in Neural Information Processing Systems*, volume 28, pages 649—-657, Montréal, Canada.