Peeking Inside the DH Toolbox – Detection and Classification of Software Tools in DH Publications

Nicolas Ruth, Andreas Niekler and Manuel Burghardt

Computational Humanities Group, Institute for Computer Science, Leipzig University – Augustusplatz 10, 04109 Leipzig

Abstract

Digital tools have played an important role in Digital Humanities (DH) since its beginnings. Accordingly, a lot of research has been dedicated to the documentation of tools as well as to the analysis of their impact from an epistemological perspective. In this paper we propose a binary and a multi-class classification approach to detect and classify tools. The approach builds on state-of-the-art neural language models. We test our model on two different corpora and report the results for different parameter configurations in two consecutive experiments. In the end, we demonstrate how the models can be used for actual tool detection and tool classification tasks in a large corpus of DH journals.

Keywords

tool studies, software entity recognition, neural language models

1. Introduction: Tools in DH

In their ”notes toward an epistemology of building in the digital humanities”, Ramsay & Rockwell [23] stress the importance of prototyping and tool building in the Digital Humanities (DH) as a scholarly activity. Building on Vannevar Bush’s [6] early notion of machines as ”*extensions of the human mind*” and J. C. R. Licklider’s [17] follow-up concept of a ”*man-computer symbiosis*”, the epistemological implications of digital tools have been discussed time and again [23, 15, 7, 5]. Besides this epistemological dimension, [13] note the importance of DH historiography. They argue that a history of the DH cannot be written without documenting its respective tools in a sustainable way. However, this has proven to be quite a difficult undertaking, as tools are oftentimes rather short-lived and at the same time highly specific for particular areas of applications or even project contexts. Furthermore, tools and software packages are oftentimes not documented at all, because the humanities still have a strong focus on bibliographic scholarship and fail to capture the work of computing humanists [13]. As a result, tools are often not adequately cited in publications [14], which further complicates sustainable documentation.

To address these problems, a number of resources have been developed to document tools

CHR 2022: Computational Humanities Research Conference, December 12 – 14, 2022, Antwerp, Belgium

nr78riku@studserv.uni-leipzig.de (N. Ruth); andreas.niekler@uni-leipzig.de (A. Niekler); burghardt@informatik.uni-leipzig.de (M. Burghardt)
https://ch.uni-leipzig.de/ (N. Ruth); https://ch.uni-leipzig.de/ (A. Niekler); https://ch.uni-leipzig.de/ (M. Burghardt)
0000-0002-1126-6280 (N. Ruth); 0000-0003-1354-9089 (A. Niekler); 0000-0003-1354-9089 (M. Burghardt)

in DH. These include tutorials such as the renowned *Programming Historian* site[1], but also tool directories, such as the *Digital Methods Initiative*[2] or the *SSH Open Marketplace*[3]. The oldest and most extensive tool directory is TAPoR (*Text Analysis Portal for Research*). TAPoR started out as a database for collecting primarily text analysis tools and recently has integrated another tool directory named DiRT (*Digital Research Tools*), which also contains tools that go beyond the text modality [13, 10]. Up to now, TAPoR 3.0 contains approx. 1,600 digital tools. Some of these tools also have information about its type and function, which is largely based on the TaDiRAH (*Taxonomy of Digital Research Activities in the Humanities*) categories [2]. While TAPoR 3.0 is already pretty extensive, it is still far from being complete. This is also reflected by a recent call for contribution, where DH scholars are asked to contribute more tools to TAPoR 3.0.[4] Although a complete list of all digital tools ever developed in the context of DH will probably never be produced, we believe that many of the current blind spots can be covered by an automated approach that will eventually allow us to paint a more coherent picture of DH tool historiography.

In this paper we propose a binary and a multi-class classification approach based on neural language models to detect and classify tools. For the detection, the binary classifier checks if *sequence has tool* or *sequence has no tool*, while for the multi-class task the model tries to assign one of seven TaDiRAH categories to a sequence that contains a tool. Taking up the perspective of scientometrics, we believe that searching for tools in existing DH publications is a good approximation of the most important tools actually used in DH research. While TAPoR may be categorized as a more qualitative, crowdsourcing-like approach, where DH scholars actively submit their tool candidates, our approach takes a more empirical, corpus-based route. In this current paper we present experiments that are based on a corpus of three established DH journals. However, we plan to use the approach on further DH publications such as DH abstracts or journals articles from neighboring disciplines such as information science or computational linguistics [20, 4].

2. Related Work: Tool Detection and Software Entity Recognition

For the identification of software tools in academic publications, two main approaches can be identified: (a) dictionary-based approaches and (b) machine learning approaches. Most of the existing lexicon-based approaches use the TAPoR site as a basic dictionary to detect tools mentioned in DH abstracts [1, 12], DH tutorials [11] and DH journal articles [5]. Some of the key problems of dictionary-based approaches are the static list of tools and the great number of false positives that are created by highly ambiguous tool names (for instance Python or R). To overcome these limitations of tool dictionaries, machine learning approaches are a promising alternative, for which we find some first attempts in the DH community already.

[1]https://programminghistorian.org/
[2]https://wiki.digitalmethods.net/Dmi/ToolDatabase
[3]https://marketplace.sshopencloud.eu/
[4]Tweet from tapordotca, April 22nd, 2022: https://twitter.com/tapordotca/status/1517564033519345664?s=21&t=Mj6Hk76pigAxGtK8iaUuKA

[14] use a combination of named entity recognition frameworks and manual filtering to detect software citations in German DHd abstracts. [25] also train an NER approach, which gives good results for a specific area of application. Some first experiments on using Transformers for the detection of tools in DH publications have been conducted by [5], who successfully use BERT embeddings to expand a static list of tools. Outside the DH domain, there are also many examples of pre-trained models for the Software Entity Recognition task [19, 8], but most of these come from the natural sciences and are therefore only moderately suitable for use in Digital Humanities publications.

In this paper, we present a new model based on the tansformer architecture RoBERTa [18] to reliably detect tools in DH publications and beyond (will be referred to as *binary model*). Furthermore, we enable the classification of tools into basic usage categories (*analysis*, *capturing*, *creation*, etc.; will be referred to as *multi-class model*), something that none of the aforementioned approaches has considered so far.

3. Data

In this study, we will use two datasets for building and evaluating our models. The first corpus (cor-1) consists of 3,737 English-language publications from three DH journals (*Computers and the Humanities*, *Digital Humanities Quarterly*, *Literary and Linguistic Computing/Digital Scholarship in the Humanities*) published between 1966 and 2020. We use cor-1 as our main data input for the training and evaluation of the model. In addition, we use another existing corpus (cor-2) that has been used by [26] to train a machine-learning approach for extracting software mentions from scholarly articles. The corpus comprises approx. 1.9 million sentences from ADHO DH conference abstracts (2015 and 2020) and PLOS ONE papers from Linguistics and Sociology[5] and is used as a model-external data source to test the generalization capabilities of our model.

cor-1: This dataset was compiled by the authors and contains no annotations. To generate training, testing, and validation data from cor-1 we need to identify passages with tool mentions. A manual extraction of text passages in our data is not feasible given the amount of text documents available. To obtain high-quality training data, we first searched for particularly popular tools by a string based search. Popular tools were identified by going through existing lists and tutorials that document DH tools (see Table 1) and selecting only those tools that were mentioned in at least two different lists. From these tools, we then removed those that exhibited a high degree of ambiguity to further improve quality. This reduced lexicon, with a total of 246 tools, served as the starting point for creating the training data. In the next step, all papers in the corpus were tokenized and searched with the tool lexicon. Whenever a tool was found successfully, the sequence was transferred to the training dataset as a text snippet with 15 tokens before and 15 tokens after the tool mention as a sequence-label pair.

> *Example*: [analysis of the sample tweets . Students evaluated each of the default analysis tools in] [**Voyant**][, and were given examples of how to use the visualizations , statistics , and]

[5]The dataset is available from https://gitlab.gwdg.de/sshoc/data-ingestion/-/tree/master/repositories/extraction. Note that cor-2 is already split into sentences which is different to the format we used in cor-1.

Table 1
List of all sources which were used to compile an inital list of tools. The resulting list was used to create a dataset with training data using the tool names as search strings.

Name	Type	URL
Programming Historian	Tutorials	https://programminghistorian.org/
forTEXT	Tutorials	https://fortext.net/
Teresah	Catalogue	https://github.com/lehkost/ToolXtractor/blob/master/src/main/resources/tools_teresah.txt
TAPoR	Catalogue	https://tapor.ca/home
Digital Methods Initiative	Catalogue	https://wiki.digitalmethods.net/Dmi/ToolDatabase
Alan Liu's DH Toychest	Catalogue	http://dhresourcesforprojectbuilding.pbworks.com
DMI	Catalogue	https://wiki.digitalmethods.net/Dmi/ToolDatabaseh
DigiHum	Catalogue	https://digihum.de/tools/

Such a sequence is annotated with the label TOOL. Then, the set of TOOL sequences was supplemented by randomly chosen NO_TOOL examples. Those were gathered from evenly distributed samples from the entire corpus and serve as negative examples of tool occurrences. For the multi-class training dataset, we used the available categories which TAPoR provides for the majority of the 246 tools and mapped them to their corresponding TaDiRAH [2] annotations in our training set. For those tools that were still lacking a category annotation, we provided a manual TaDiRAH annotation. Since TAPoR sometimes assigns multiple categories for single tools, we selected the – in our opinion – best-fitting category to prevent our classifier from having to deal with multi-label situations. [6]

COR-2 is used to test the generalization capabilities of our model and to identify external examples which can be used to enrich the training data. In a second experiment, we will investigate how the model performance is influenced by this alteration. Each line in COR-2 contains a sentence and an annotation that documents whether there are tool mentions in a particular section of the sentence or not. If any tool occurrences are annotated, we take over the sentence with the label TOOL. If there are no tool annotations, we label the sentence as NO_TOOL. This data set therefore offers the opportunity for us to define an external gold standard.

4. Methods

Software and tool entity recognition can be seen as a text classification problem. Our approach is a supervised machine learning method that consists of a binary and a multi-class classification task. The basic idea of the approach is that the mention of a tool in a paper happens in a specific linguistic context, which can be mapped and classified in the form of a sequence embedding using a Transformer-based language model. On the one hand, we try to determine whether a tool is in a sequence (binary task) and on the other hand, we try to assign a tool usage category to a sequence with a tool named in it (multi-class task).

[6]All the tools and their TaDiRAH categories can be found here: https://git.informatik.uni-leipzig.de/computational-humanities/tools-in-dh/tool-classifier/-/blob/main/resources/tool_lists/tadirah_taxonomy.json

We used a uniform framework for all classification tasks. The superior performance of Transformer-based language models in the context of text classifications was one of the main decisions for our technical implementation. We experimented on the Transformer-based model RoBERTa-Base as the main input for the classification component [18]. For efficiency reasons, distilRoBERTa-Base was used as a ”case-sensitive” and ”knowledge distilled” variant, as it produced comparably good results to the RoBERTa-Base variant, but with considerably less computation effort. In detail, the solution consists of a Transformer-based encoder which uses the classification token ([CLS]) of the model directly, a linear layer, and an output layer with Softmax activation. For each instance, the encoder converts each token into an embedding vector and adds a special classification token at the beginning of the sequence. The final hidden state corresponding to this token is used as the aggregate sequence representation for classification tasks [9]. This sequence embedding is then fed into the linear layer and the output layer. The output is a probability vector with values between 0 and 1, which is used to compute a logistic loss during the training phase, and the class with the highest probability is chosen during the inference phase. Language models are often trained on very general language and do not represent the language well for a particular text source. Therefore, a fine-tuning step is added and the weights of the language model are adjusted to better support the classification task at hand. In the experiments, different numbers of epochs were used in fine-tuning to test the generalization. 1.0 and 0.2 epochs were used on a batch size of 16. We also experimented with different amounts of epochs but found only one setting to be superior among all other experiments. Because of that, we only used the reported epoch numbers for the experiments in this paper.

The experiment design in our study consists of several parts that build on each other. In the first step, we use only the training data from COR-1. Following this step, we assess automatically classified false positive (FP) examples from COR-2 to qualitatively investigate errors in the model. In a second experiment, we enrich our training data with the identified FP-examples from COR-2 and re-evaluate the approach.

Experiment 1: In our first experiment, the Transformer model is fine-tuned on the training data. Instead of using a static train, test and validation split, the datasets for each of the two classification tasks were split into five training and test sets according to the principle of a 5-fold cross-validation. This was done using the K-fold component in the SciKitLearn framework[7]. Finally, the dataset consists of 18,898 sequence-label pairs divided into 5 folds. For each round of cross-validation, four folds were used for training and the fifth fold was used as a test dataset for validation. The experiment tests performance individually for the binary and multi-class cases on COR-1. We report on the average performances for each split of COR-1 in the evaluation section. We also evaluate the binary case of EXPERIMENT 1 twice. On COR-2, we apply the binary model and use the existing annotations of this dataset to evaluate whether our classifier can correctly determine the existence of a tool.

Qualitative Model Assessment: In a second step, we use this model to identify additional tool mentions from COR-1 and COR-2 which are not yet annotated. In detail, we look at where

[7]https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html

Table 2
Presentation of the evaluation results of the binary model on our corpora. The model was evaluated based on COR-1 via 5-fold cross-validation and COR-2 could be evaluated via the annotations it contains. With +500 we denote the evaluation based on the model which is trained with additional 500 examples of FP from COR-2. Classification results are averages of the 5-fold cross-validation.

Corpus	Precision	Recall	F1
COR-1	0.97	0.99	0.98
COR-2	0.57	0.88	0.69
COR-1 +500	0.96	0.99	0.97
COR-2 +500	0.67	0.90	0.77

the model makes mistakes and which text components are relevant for the classification. This allows us to better understand what the model learns and what consequences our modeling decisions have.

Experiment 2: In the second experiment, we will expand the training data set. Based on the evaluations from EXPERIMENT 1 and the qualitative tests, we repeat the evaluation again under different conditions. For this purpose, we will include the false positive examples from COR-2 into the training data.

5. Evaluation

In this section we present the evaluation of the classification results with the values precision, recall and F1. Furthermore, we will investigate the multi-class case with micro and macro averaged F1 values. In the first experiment, we will document the binary and the multi-class case. In the qualitative evaluation, we will show some examples and explain the decisions of the classifier. Based on these observations we adapt the model and repeat the evaluation.

As mentioned before, we based the evaluation in our first experiment on a 5-fold cross-validation restricted to COR-1. Based on the averaged metrics, the result of the binary classification is very good on the corpus internal evaluation. The detailed results are shown in Table 2. For the second case of multi-class classification, Table 3 shows the results. With an accuracy of 0.974 the model shows excellent performance within COR-1. Although the recall is below 0.9 in two categories, the overall precision is very high.

Based on these promising results, we wanted to test how the model behaves when transferred to another text type. Since we have annotations in COR-2 to indicate whether a sentence contains a tool, we can evaluate the binary approach very easily outside the training data. The results are documented in Table 2. Although we have a very high recall in the classification of tools, the precision here is very low. This can only be due to false positive classifications made in COR-2. For this reason, we will look at the classification decisions in some more detail in the next step and take a closer look at the false positive decisions found in COR-2. We looked at the models output using the *Captum-based Transformers Interpret library* [16, 21]. This library calculates weights on how individual words in a sequence contribute to a classification. Thereby, positive values are associated with the searched class. In Table 4 we show three examples

Table 3
Presentation of the evaluation results of the multi-class model. The results are averages that are produced with a 5-fold cross-validation on COR-1.

Class	**Precision**	**Recall**	**F1**
Analysis	0.946	0.974	0.960
Capture	0.984	0.964	0.972
Creation	0.932	0.852	0.890
Dissemination	0.950	0.972	0.960
Enrichment	0.966	0.936	0.952
Storage	0.954	0.848	0.898
NO_TOOL	0.992	0.984	0.990
Accuracy (Micro)			0.974
Macro	0.956	0.938	0.950

Table 4
Illustration of three selected examples of FP classifications from COR-2. The background colors represent the weights for the positive and negative class. Terms contributing to the positive class are green and red is assigned to terms contributing to the negative class.

Example A	Diff erences in demographic and other characteristics noted there were examined using a Chi Square test , and differences were considered significant if p < 0 . 05 .
Example B	The angle display was realized simply by two bars , which could be opened or closed by left and right mouse clicks .
Example C	Four of the cafeteria diet food items were administered per day and variety of diet was maintained by alternating food items daily .

taken from the output of Transformers Interpret. The separation of the tokens was determined directly from the Transformer, which uses subword tokens. This results in fragmented single words in some representations (see *Diff* – *erences* in example A).

The FP predicted in COR-2 are almost all descriptions of a methodological approach. In these examples, one can relatively well recognize a certain pattern of syntactic components. In all examples, and also in many FP classifications from COR-2, there are constructions of nouns and proper nouns followed by verbs that indicate a methodological activity. In our examples, we find *were examined using*, *were considered significant if*, *was realized simply by*, *be opened or closed by*, *were administered per* and *was maintained by*. This surface form is very similar to the constructions used in the naming of tools. Actually, we could just insert a tool at the end of many of the phrases and the sentence would make perfect syntactical sense. We believe that all of these phrases are standardized language constructs to document a procedural approach. Since the naming of tools typically belongs to the methodical part of the text, their descriptions and what was done with them strongly resemble these passages.

The lesson to be learned here, is that our classifier learns more about syntactical patterns or sentence constructions than about the semantic representations of the entity references they contain. We conclude that COR-2 by containing a broader scientific domain, there are more such constructs in the data that would not meet the definition of a tool when annotated. While examples A-C demonstrate how syntactic constructions influence the classification model in a way it is likely to produce false positives, there are also many examples for words that have higher weights and that actually imply some realistic tool context, for instance *measure*, *(arithmetic) mean*, *assess*, *score* or *analyses*.

After showing that the false positive classifications come from a somewhat more restrictive definition of what is and is not a tool, we can still evaluate whether the results can be improved by expanding the training data set with the false positive classifications from COR-2. We retrained the model with 500 additional data points and evaluated it again on the COR-2. The results show a boost in quality, which also shows that our model is easy to adapt and can be adapted to other definitions of tool mentions.

The experiments with extended training samples show that we can easily extend our training data with another dataset and not lose any quality on the COR-1. On the contrary, we gain almost 0.08 points on the COR-2 and the precision has increased significantly. Although both datasets were created and annotated with a slightly different idea of what a tool is, we were able to create a reliable classifier. We assume that the loss of quality with COR-2 is also due to the fact that we do not have sequences, but sentences, which can sometimes be very short.

6. Model Applications – Examples

In this section we demonstrate how our classifier can be used to detect and classify tools in COR-1 – a corpus with three popular DH journals. With the detection of tools in scientific papers we can learn more about methodological standards and software packages that are being used within a certain discipline, such as the DH. Furthermore, the categorization of tool occurrences is useful to understand typical scholarly activities of a discipline and to investigate which activities have been digitally enhanced so far.

6.1. Application of the binary model

As was argued in the beginning, the TAPoR 3.0 tool repository includes a great number of tools already. However, using our binary classifier on COR-1, we found numerous examples of tools that are currently not included in TAPoR 3.0. Our approach produces sequences of length 31 tokens and labels those as TOOL or NO_TOOL. To be able to extract the actual tool entities from these sequences, we used the Stanford Stanza POS tagger [22] and generated a list of all the proper nouns contained in the sequences. Besides tool entities, these also include place and person names, organizations and institutions, etc. We had a look at any proper noun with a document frequency > 3 and removed any non-tool entities from the list. This way we were able to detect around 100 tools that are not yet reported in TAPoR 3.0.[8]

[8]Note that we also identified around 50 tool candidates that need some more close reading to verify if they are actual tools or some other named entity.

In the following we present the main types of tools (manually assigned by us) that were identified by our classifier.[9]

- *markup technologies* (TEI, XML, SGML, HTML, XSLT, XPointer, etc.)
- *metadata standards* (Dublin Core, CIDOC, MARC, etc.)
- *programming languages* (Java, BASIC, Pascal, COBUILD, PASCAL, PHP, etc.)
- *authoring tools* (HyperCard, Photoshop, Storyspace, PageMaker, etc.)
- *operating systems* (DOS, UNIX, Linux, Windows, etc.)
- *web browsers* (Netscape, Lynx, Internet Explorer, etc.)
- *web services* (Google, Google Books, Google Scholar, Google Earth, Google Maps, Facebook, YouTube, Wikipedia, Yahoo, Flickr, Gopher, Internet Archive, etc.)
- *NLP* (Word Net, TreeTagger, etc.)
- *crowdsourcing* (Mechanical Turk, Transcribe Bentham, etc.)
- *corpora and databases* (Perseus, Project Gutenberg, EEBO, Europeana, HathiTrust, ProQuest, JSTOR, etc.)
- *statistics* (SPSS, Zeta, etc.)
- *word processors* (Microsoft Word, WordPerfect, WordStar, WordNet, Wordstar, MS Word, etc.)
- *infrastructures* (CLARIN, DARIAH, etc.)
- *hardware* (Kinect, iPad, etc.)

One reason for these *new* tool discoveries could well be our very broad definition of *digital tools*, which might be somewhat narrower in TAPoR 3.0. For future analyses of tools in DH, and also for a further fine-tuning of the classifier, we will need to discuss what should be counted as a tool and what not. While we believe *markup technologies* and *programming languages* could be classified as ”tools for making” (as opposed to tools for exploring and thinking) [3], *web services* and *corpora* might be harder to justify as actual tools.

6.2. Application of the multi-class model

Our multi-class model was able to classify the tool sequences into six of the seven main TaDIRaH categories for tool types. ”Interpreting” was not included in the model, as we did not have enough examples in our training data. Interestingly, tools for ”Analysis” are by far the most frequent tools, followed by tools for ”Dissemination” and ”Enrichment” (see Table 5).

To see how these categories have evolved through time we have plotted the frequencies diachronically (see Figure 1). While we can observe a general diachronic upwards trend for most of the categories, it is noticeable that tools for analysis are not only the most frequent tool category in total, but that they are also more or less consistently mentioned in DH publications throughout the whole time span – with a general upwards trend, similar to the other categories. Tools for analysis include examples from many different areas, including *network*

[9]For a list of all tool entities see https://docs.google.com/spreadsheets/d/1ZMV3kFAF2mooclIlvW_0gJYSh68FlE_jgGhyeT-VVqc/edit?usp=sharing

Table 5
Overview of TaDiRAH categories and how often they were assigned to tool sequences in our corpus.

Category	**Frequency**
Analysis	8,121
Dissemination	1,548
Enrichment	1,289
Capture	801
Storage	636
Creation	482

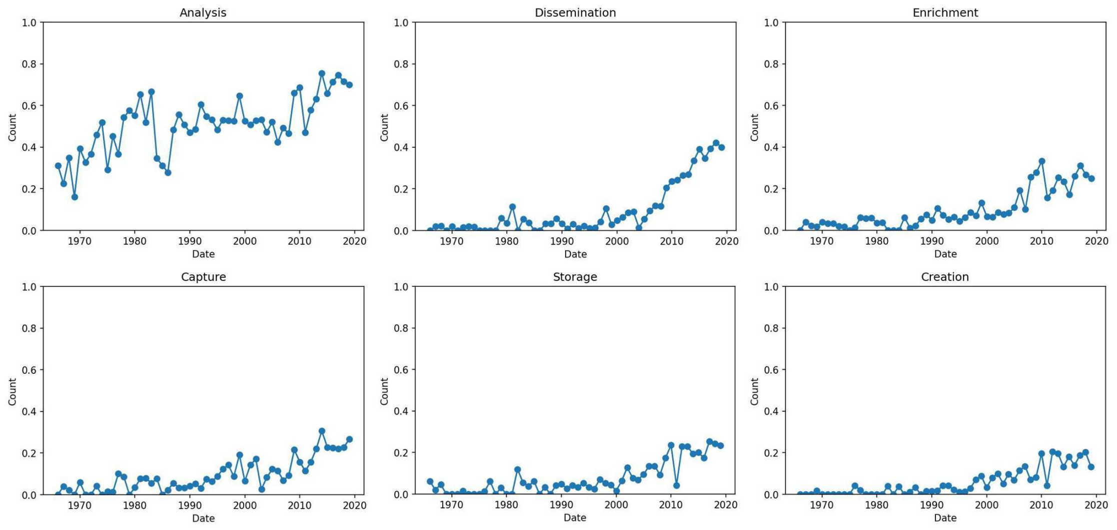

Figure 1: Diachronic overview of the identified TaDiRAH categories. The *Count* axis represents the normalized document counts of a categories occurrence in a year. Normalization was done using the total document count within a year.

analysis (Gephi, Cytoscape, NodeGoat, networkX), *natural language processing* (NLTK, Stanford POS Tagger, Gensim), *text analysis* (Voyant, AntConc), *geo information systems* (ArcGIS, Leaflet), *music analysis* (music21) and many more. This observed dominance of tools for analysis is also interesting for future work, as we believe this is the category of tools that has the most potential for epistemological shaping of scholarly research processes, which means we will have a closer look at this category in follow-up studies.

7. Conclusion & Future Work

In this paper we have presented a Transformer-based classifier for the detection and classification of digital tools. The task of tool detection worked very well (F1 = 0.978) for a large corpus of DH journals, which also was the basis of the training data. Tool detection results for another corpus that includes ADHO abstracts and PLOS ONE articles from Sociology and

Linguistics also worked well, with an F1 score of 0.69 and an increase to 0.77 after fine-tuning the model with 500 false positive examples from the novel corpus. It can be concluded that the model yields good results in the domain of DH journal publications already, and can also be successfully adopted to other domains.

We plan to use this model for follow-up studies in which we want to investigate the actual tool results and their epistemological implications for DH scholars in a more detailed way. First, we plan to enhance our current DH corpus by adding some more recent journals such as "Journal of Cultural Analytics"[10] or "International Journal of Digital Humanities"[11]. We also plan to add abstracts from past DH conferences, which are available from the "Index of Digital Humanities Conferences"[12]. We will evaluate the performance of our model on such an enlarged corpus and fine-tune it with further training data if necessary. As was mentioned in the previous section, we will also have to agree upon a working definition of actual *tools* [see 3] as opposed to digital resources, services, infrastructures, etc. We certainly plan to correlate the tools and tool categories with further metadata such as geolocation of authors, gender of authors, disciplinary background of authors, keywords and LDA topics. By means of such large-scale analyses we hope to be able to contribute to the historiography of DH through the lens of what has been called "tool science" [24].

References

[1] L. Barbot, F. Fischer, Y. Moranville, and I. Pozdniakov. *Which DH Tools Are Actually Used in Research?* 2019. URL: https://weltliteratur.net/dh-tools-used-in-research/.

[2] L. Borek, Q. Dombrowski, J. Perkins, and C. Schöch. "TaDiRAH: a Case Study in Pragmatic Classification". In: *Digital Humanities Quarterly* 10.1 (2016).

[3] J. Bradley. "Digital tools in the humanities: Some fundamental provocations?" In: *Digital Scholarship in the Humanities* 34.1 (2019), pp. 13–20.

[4] M. Burghardt and J. Luhmann. "Same same, but different? On the Relation of Information Science and the Digital Humanities A Scientometric Comparison of Academic Journals Using LDA and Hierarchical Clustering". In: (2021).

[5] M. Burghardt, J. Luhmann, and A. Niekler. "Tools as Epistemologies in DH? A Corpus-Based Exploration". In: Book of Abstracts of the ADHO Digital Humanities Conference. Tokyo, 2022.

[6] V. Bush. "As We May Think". In: *The Atlantic* (1945).

[7] M. Dalbello. "A genealogy of digital humanities". In: *Journal of Documentation* 67.3 (2011), pp. 480–506.

[8] David Schindler, Felix Bensmann, Stefan Dietze, and Frank Krüger. "The role of software in science: a knowledge graph-based analysis of software mentions in PubMed Central". In: *PeerJ Computer Science* 14.8 (2022).

[10]https://culturalanalytics.org/

[11]https://www.springer.com/journal/42803

[12]https://dh-abstracts.library.virginia.edu/

[9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. “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*. Minneapolis, Minnesota: Association for Computational Linguistics, 2019, pp. 4171–4186.

[10] Q. Dombrowski. “What Ever Happened to Project Bamboo?” In: *Literary and Linguistic Computing* 29.3 (2014), pp. 326–339.

[11] F. Fischer and Y. Moranville. *DH Tools Mentioned in ”The Programming Historian”*. 2020. URL: https://weltliteratur.net/dh-tools-programming-historian/.

[12] F. Fischer and Y. Moranville. *Tools mentioned in DH2020 abstracts*. 2020. URL: https://weltliteratur.net/tools-mentioned-in-dh2020-abstracts/.

[13] K. Grant, Q. Dombrowski, K. Ranaweera, O. Rodriguez-Arenas, S. Sinclair, and G. Rockwell. “Absorbing DiRT: Tool Directories in the Digital Age”. In: *Digital Studies/le Champ Numérique* 10.1 (2020).

[14] U. Henny-Krahmer and D. Jettka. “Softwarezitation als Technik der Wissenschaftskultur: Vom Umgang mit Forschungssoftware in den Digital Humanities”. In: *DHd2022: Kulturen des digitalen Gedächtnisses. Konferenzabstracts*. Potsdam, 2022, pp. 203–206.

[15] Kaden, Ben. *Zur Epistemologie digitaler Methoden in den Geisteswissenschaften*. 2016. URL: https://zenodo.org/record/50623.

[16] N. Kokhlikyan, V. Miglani, M. Martin, E. Wang, B. Alsallakh, J. Reynolds, A. Melnikov, N. Kliushkina, C. Araya, S. Yan, O. Reblitz-Richardson, and F. Ai. “Captum: A unified and generic model interpretability library for PyTorch”. In: (2020), p. 11. URL: https://arxiv.org/abs/2009.07896.

[17] J. C. R. Licklider. “Man-Computer Symbiosis”. In: *IRE Transactions on Human Factors in Electronics* Hfe-1.1 (1960), pp. 4–11.

[18] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. 2019. arXiv: 1907.11692[cs]. URL: http://arxiv.org/abs/1907.11692.

[19] P. Lopez and Laurent Romary. “GROBID - Information Extraction from Scientific Publications”. In: *ERCIM News 100* 100 (2015).

[20] J. Luhmann and M. Burghardt. “Digital humanities–A discipline in its own right? An analysis of the role and position of digital humanities in the academic landscape”. In: *Journal of the Association for Information Science and Technology* 73.2 (2022), pp. 148–171.

[21] C. Pierse. *Transformers Interpret*. Version 0.5.2 date-released: 2. Feb. 14, 2021. URL: https://github.com/cdpierse/transformers-interpret.

[22] P. Qi, Y. Zhang, Y. Zhang, J. Bolton, and C. D. Manning. “Stanza: A Python Natural Language Processing Toolkit for Many Human Languages”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Online: Association for Computational Linguistics, 2020, pp. 101–108.

[23] S. Ramsay and G. Rockwell. “Developing Things: Notes toward an Epistemology of Building in the Digital Humanities”. In: *Debates in the Digital Humanities.* Minneapolis; London: University of Minnesota Press, 2012.

[24] C. Wolff. “The case for teaching ”tool science”: Taking software engineering and software engineering education beyond the confinements of traditional software development contexts”. In: *2015 IEEE Global Engineering Education Conference (EDUCON).* Tallinn, Estonia: Ieee, 2015, pp. 932–938.

[25] A. Zarei, Y. Seung-Bin, F. Fischer, M. Ďurčo, and P. Wieder. “Measuring the Use of Tools and Software in the Digital Humanities: A Machine-Learning Approach for Extracting Software Mentions from Scholarly Articles”. In: *Book of Abstracts, ADHO DH Conference.* Tokyo, 2022.

[26] Zarei, Alireza, Seung-Bin, Yim, Ďurčo, Matej, Illmayer, Klaus, Barbot, Laure, Fischer, Frank, and Gray, Edward. “Der SSH Open Marketplace: Kontextualisiertes Praxiswissen für die Digital Humanities”. In: *DHd2022: Kulturen des digitalen Gedächtnisses. Konferenzabstracts.* Potsdam, 2022.