UNIVERSITÄT ZU LÜBECK

From the Institute of Information Systems
of the University of Lübeck
Director: Prof. Dr. rer. nat. habil. Ralf Möller

Enabling Research Data Management for Non IT-Professionals

Dissertation
for Fullfillment of
Requirements
for the Doctoral Degree
of the University of Lübeck

from the Department of Computer Sciences and Technical Engineering

Submitted by
Simon Schiff
from Hamburg

Lübeck 2023

First referee: Prof. Dr. rer. nat. habil. Ralf Möller
Second referee: Prof. Dr. Heike Zinsmeister
Date of oral examination: November 25, 2024
Approved for printing. Lübeck, February 16, 2025

Abstract

In almost all academic fields, results are derived from found evidence such as objects to be digitized, case studies, observations, experiments, or research data. Ideally, results are linked to its evidence to ease data governance and reproducibility of results, and publicly stored in a research data repository to be themselves linked as evidence for new results. This linking has created a huge mesh of data over the years. Searching for information, deciding whether found information is relevant, and then using relevant information for producing results costs a lot of time in such a mesh of data. Due to the fact that a high investment of time is associated with high costs, funding agencies such as the German Research Foundation (Deutsche Forschungsgemeinschaft; DFG) or the Federal Ministry of Education and Research (Bundesministerium für Bildung und Forschung; BMBF) demand a data management plan (DMP). A DMP is designed to reduce the costs of projects submitted to a funding agency and to avoid future costs when data repositories are to be reused. Nevertheless, a DMP is often not fully implemented because it is too costly, which in the long run leads to a mesh of data. In this thesis, we identify problems and present solutions usable by non-IT-experts to spend less time on solving problems that arise at implementing a DMP at each project's repository and coping with a huge mesh of data across many repositories. According to our observations, humanities scholars produce research data that are meant to be printed later or uploaded at a repository. Potential problems that arise at a repository, independent of other repositories, to be solved are manifold. Data to be printed is encoded with a markup language for illustration purposes and not machine interpretable formatted. We not only show that such formatted data can be structured with a parser to be interpreted by machines, but also what possibilities open up from the structured data. Structured data is automatically combined, linked, transformed into other formats, and visualized on the web. Visualized data can be cited and annotated to help others assess relevance. Once, problems are solved at each repository, we show how we cope with data linked across repositories. This is achieved by designing a human-aware information retrieval (IR) agent, that can search in a mesh of data for relevant information. We discuss in what way the interaction of a user with such an IR agent can be optimized with human-aware collaborative planning strategies.

Kurzfassung

In nahezu allen Forschungsgebieten werden Ergebnisse aus gefundenen Belegen wie zum Beispiel aus Digitalisaten von Objekten, Fallstudien, Beobachtungen, Experimenten oder Forschungsdaten abgeleitet. Im Idealfall sind Ergebnisse mit Belegen verlinkt, um Data Governance und Reproduzierbarkeit von Ergebnissen zu erleichtern, und öffentlich in einem Forschungsdaten-Repositorium gespeichert, um selbst als Beleg mit neuen Ergebnissen verlinkt werden zu können. Durch diese Verlinkung entstand im Laufe der Jahre ein riesiges Datengeflecht. Die Suche nach Informationen, die Entscheidung über deren Relevanz und die Verwendung der als für relevant befundenen Ergebnisse kostet in solch einem Datengeflecht sehr viel Zeit. Aufgrund dessen, dass ein hoher Zeitaufwand stets mit hohen Kosten verbunden ist, fordern Finanzierungsagenturen wie die deutsche Forschungsgemeinschaft (DFG) oder das Bundesministerium für Bildung und Forschung (BMBF) einen Datenmanagementplan (DMP). Ein DMP wird entworfen, um Kosten bei einem bei einer Finanzierungsagentur eingereichtem Projekt zu reduzieren und um zukünftige Kosten zu vermeiden, die bei der Wiederverwendung von Forschungsdaten-Repositorien entstehen können. Jedoch wird ein DMP aufgrund der hohen Kosten häufig nicht umgesetzt, was langfristig zu einem Datengeflecht führt. In dieser Arbeit präsentieren wir Lösungen, um nicht-IT-Spezialisten dabei zu unterstützen, weniger Zeit zum Lösen von Problemen sowohl bei der Umsetzung eines DMP lokal bei jedem Repositorium als auch bei der Bewältigung des Datengeflechts über viele Repositorien hinweg zu verschwenden. Nach unseren Beobachtungen produzieren Geisteswissenschaftler Forschungsdaten, die dazu gedacht sind, später gedruckt oder in einem Repositorium hochgeladen zu werden. Potentielle Probleme, die innerhalb eines Repositoriums entstehen können, sind vielfältig. Zu druckende Daten sind, wenn diese mit einer Markupsprache kodiert wurden, für Illustrationszwecke und nicht maschineninterpretierbar formatiert. Wir zeigen nicht nur, dass auf diese Art formatierte Daten mit einem Parser strukturiert werden können, um von Maschinen interpretiert zu werden, sondern auch welche Möglichkeiten sich aus den strukturierten Daten eröffnen. Strukturierte Daten können automatisch kombiniert, verlinkt, in andere Formate transformiert und im Web visualisiert werden. Visualisierte Daten können zitiert und annotiert werden und damit bei der Abschätzung der Relevanz helfen. Sobald die Probleme bei jedem Repositorium behoben sind, zeigen wir wie große Datenmengen, die über Grenzen von Repositorien

hinweg verlinkt sind, bewältigt werden können. Dies geschieht durch den Entwurf eines Human-Aware Information Retrieval Agenten, welcher in einem Datengeflecht nach relevanten Informationen suchen kann. Wir diskutieren, wie die Interaktion eines Benutzers mit solch einem Agenten mithilfe von Human-Aware kollaborativen Planungs-Strategien optimiert werden kann.

Acknowledgments

Before I wrote my bachelor's thesis at the Institute of Information Systems at the University of Lübeck, I did not initially intend to pursue a doctorate later. During the master's program, I worked for the institute and learned a lot by writing papers, giving talks at conferences, implementing something that I initially thought was not possible, and correcting and supervising exercises. All this would not have been possible without my supervisor Ralf Möller. Thank you for everything I have learned in the last years which made me have the confidence to pursue a doctorate, and sorry for not participating at the lecture "Non-Standard Databases".

I really missed all discussions and conversations with my colleagues in person during the pandemic. Working from home all the time might be practical from a first sight, however it is not in the long run. Whenever I mentioned in this dissertation that I used a server for something, it ran flawlessly thanks to Nils. I really enjoyed all technical discussions I had with you and learned a lot from you. Thank you, Angela, for helping me filling out forms like "Reisekostenrückerstattung" so many times. Whenever I had a conceptual idea on how to solve a particular problem and then got stuck, it was very helpful for me to discuss with you, Magnus. Writing my first paper was very challenging. Thank you, Marcel, for your patience in going over our paper again and again, and for the many conversations we had at the institute. It's always good to have a plan, and it really helped me to talk to you regularly, Sylvia. Without any research data it is nearly impossible writing a dissertation about data management and to include insights from the real-world. Many thanks to Eva Wilden, who not only shared her data with me but also gave me an insight into the precise work of a humanities scholar.

I really enjoyed, with minor exceptions, writing my thesis by using LaTeX and would like to thank everyone who has contributed to this project.

Thanks to my parents, family and friends for supporting me and always having an sympathetic ear for me. Thank you, Mena, for helping me throughout my work and to motivate me to continue when I have not made any progress.

Simon Schiff, Lübeck, September 2023

Contents

1. Introduction

Research data is any information that has been collected, observed, generated or created to validate original research findings (University of Leeds, 2022) and has proven to be beneficial. Various research fields in data science, such as information retrieval (IR), natural language processing (NLP), data mining (DM), question answering (QA) or machine learning (ML), have emerged to support this validation. For instance, in the field of IR, validation is supported by designing and implementing a system that returns relevant information from data concerning a query. Hence, problems such as finding information, translating texts, answering questions by extracting facts from a large corpus are to be solved in data science. Solving these problems in data science often requires access to huge amounts of data. Although data science is supposed to solve problems in the handling of data, many arise even in this field that need to be solved before data is usable for research purposes. Such problems are caused by a lack of time, money, planning of a project in advance, experience, or IT skills and occur in almost all research areas, not just data science. For a deeper understanding of issues that arise across different scientific fields – and not only in data sciences – it is useful to consider the whole life cycle of data.

In general, the life cycle of data spans its creation and archiving and ends when it is not needed further. Data is either created through the digitization of objects, created virtually as born-digital data (Society of American Archivists, 2023), or is created by reusing preserved data. In the case of research data, *desired* research data life cycles should share the following steps (Longwood Research Data Management, 2022):

Plan & Design One should plan and design of how data is managed at a project at beforehand to reduce costs. Planning is simplified by creating a data management plan (DMP) that is not only recommended but also mandated by many funding agencies, such as the German Research Foundation (Deutsche Forschungsgemeinschaft; DFG) (Jahnen and Hartig, 2022) or the Federal Ministry of Education and Research (Bundesministerium für Bildung und Forschung; BMBF) (Universität Konstanz, 2022). A DMP can help prevent making mistakes, such as violating data protection laws, as the violation of these laws can result in expensive penalties.

Collect & Create The actual data needs to be either collected from existing data or created by digitization of objects, or as born-digital data. Collecting or creating data from scratch can be challenging if, for instance, objects to be digitized are not easy to access, but it is easier if it is done according to a plan.

Analyze & Collaborate Analyzing data, whether collaboratively or not, is the actual step to validate original research findings. Problems arise if, for instance, data is analyzed collaboratively, but collaborators do not have a data usage agreement, work from various places, or use different tools and document formats.

Evaluate & Archive If a project on which data is produced or reused ends, data is often discarded or stored in a place where it is inaccessible to others, as it may be forbidden to archive or publish the data, too costly, or one simply does not want to give others access to the data. However, publication of research data is now often desired and sometimes even mandated, and is therefore becoming more common.

Share & Disseminate Despite the benefits of sharing data, one should not ignore ethical issues and data privacy. In addition, data is often shared without additional information that would make it easier for those with whom the data is shared to reuse the data.

Access & Reuse Access and reuse of data are beneficial, as data need not to be produced from scratch. However, data is often not reused because it has not been found, is formatted in a proprietary way, or a lot of money is charged for access.

As a rule of thumb, in order to implement concepts from the data life cycle plan (Longwood Research Data Management, 2022), one can follow the **F**indability, **A**ccessibility, **I**nteroperable, and **R**euse (FAIR) principles (GO FAIR, 2022):

Findable Before anyone can make decisions based on data that has been reused rather than created from scratch, one needs to be able to find the data. A proposed solution is to associate data with metadata having a unique identifier. Many standards for unique identifiers have been established and are widely accepted, such as the digital object identifier (DOI) or the International Standard Book Number (ISBN) (Paskin, 1999). Metadata and data need to be findable by, for instance, being indexed by search engines such as Google Scholar or Semantic Scholar. Unique identifiers are citable and, thereby, data associated with the identifier can be found more easily by others.

Accessible If data is found, one needs to be able to fetch the metadata via standardized protocols, and metadata should be persisted, even if the data itself no longer exists, as metadata contains as well as the data important information.

Interoperable Metadata and data itself need to be standardized to simplify the integration of associated data with existing data. Without standards, integrating data may even be impossible without the help of an IT-expert and very expensive when using an expert's assistance. Metadata itself should, depending on the use case, point to other metadata which is associated with related data. Such reference data may be an earlier version based on raw data such as sensor readings or supplemental data.

Reuse The last and most important point, according to the FAIR guidelines, is the reuse of data. For reusing data, the associated metadata requires specific contents such as a license and provenance of the data. The latter could be important for ethical reasons or to trace back errors or biases. A license is important to clarify access and usage rights.

Principles, such as FAIR are domain independent but there are also other guidelines especially for domain-specific areas. As an example for such a domain-specific guideline, **C**ollective benefit, **A**uthority to Control, **R**esponsibility, and **E**thics (CARE) can be considered which is designed with the focus on a correct handling of data of indigenous people. For instance, CARE (Carroll et al., 2020) is complementary to the FAIR principles and is domain-specific with respect to data about indigenous peoples (Martinez Cobo, 1987):

Collective benefit Everyone, especially indigenous peoples, need to benefit from the use of data.

Authority to Control Indigenous peoples should have control of the data about themselves.

Responsibility Everyone working with data about indigenous peoples needs to be responsible for showing how their work supports the collective benefit.

Ethics Data about indigenous peoples is not meant to be used to stigmatize them.

Adhere to principles, such as those proposed by the FAIR or CARE guidelines, help to implement concepts from Longwood Research Data Management (2022) to solve problems that would otherwise arise during managing data stored at project repositories.

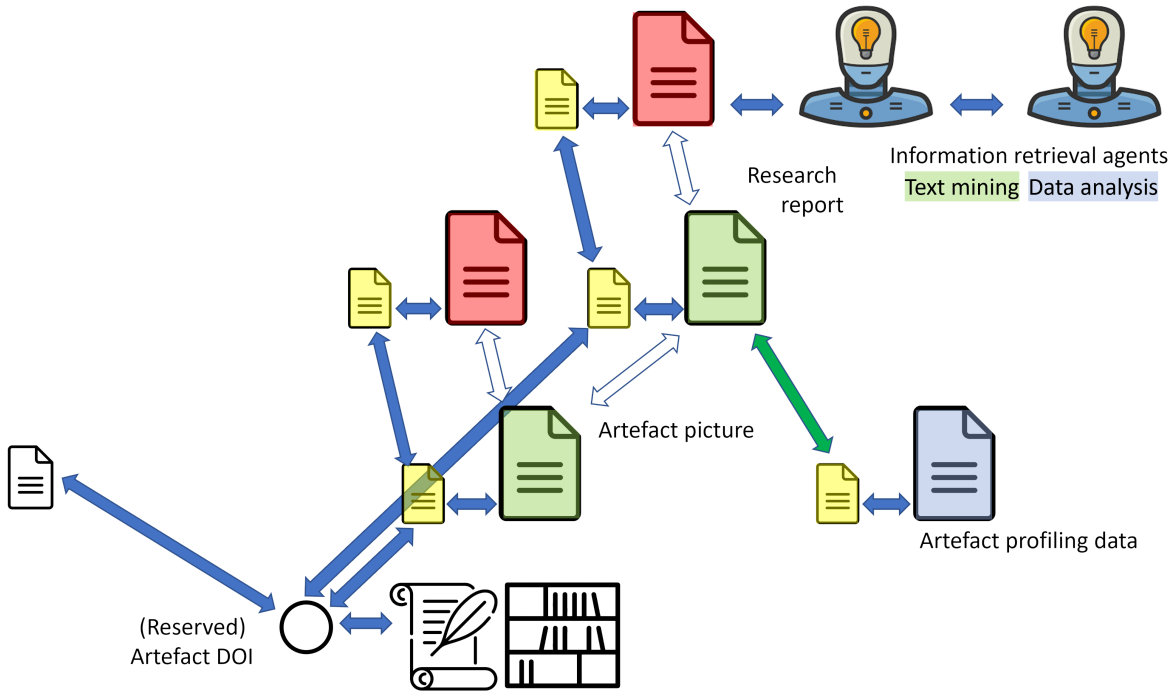


Figure 1.1.: Illustration of a mesh of data

However, researchers spend a lot of time and money on data cleaning, formatting, linking and metadata creation. From a global perspective, across many repositories containing data that are linked with data stored at other repositories, researchers have to search in a very huge mesh of data. Such a huge mesh of data is created over a long period of time and is becoming bigger and more complex.

Independent of the research field, researchers make observations, such as finding written artefacts from the past, a disease that cannot be treated at the moment, or problems that occur during a specific process to be optimized. Observations need to be kept, by creating a digitized version of them in any format, so that they can be reviewed, examined and used as evidence later. Problems that occur during the digitization are manifold, have various reasons, and are the fault for a mesh of data locally at each repository. For example, a mesh of data is depicted in Figure 1.1 that stems from the digitization of artefacts stored at a library. At the bottom middle, hundreds of described artifacts are the subject of research findable in a library that is difficult to access. It is almost impossible to uniquely refer to these artefacts via a unique identifier as the library has no bookshelf numbers. Pictures of written artefacts can be made only once

as access to the library is highly restricted and it might be impossible to transcribe all texts from pictures that where made. Additionally, a material analysis of the artefacts is promising, however either too costly or not permitted. Hours of audio transcriptions of the contents of the written artefacts are available as well. All observations, where it was possible to collect them on site, are stored in a repository.

The task of a researcher is then to produce results based on data stored at the repository, such as transcribing texts from written artefact pictures to create a critical edition. However, data was collected on site in a hurry without associating it with additional metadata. Metadata contains a description, image resolution, length of a recording, a unique identifier, a title and many more and its creation is very time-consuming. An IR-system, supporting a researcher to find data quickly, cannot index the data if it is not associated with metadata, researchers cannot point explicitly to any data point without a unique identifier part of the metadata, and one needs to download and analyze data manually from the repository to decide whether it is relevant or not.

Hence, a lot of work needs to be done before deriving results from observations, such as producing metadata about the artefacts. However that is mostly not done caused by lack of time, of knowledge in IT, or of knowledge why the effort would be worthwhile. Results then do not contain any explicit link to made observations and are therefore implicit links. Instead, descriptions pointing to the origin from where results stem from are written in natural language.

Other local repositories containing related observations or results may exist, as depicted in Figure 1.1 on the middle left. Data across repositories are possibly linked with each other implicitly, explicitly or not at all. In the latter case, however, an indirect linkage through identically but differently named individuals cannot be excluded. A faceted search engine can only index data stored at a repository with available metadata. Published results, containing among others implicit links, are less likely to be reused or further processable by machines.

Despite following guidelines such as FAIR, research based on data as well as managing data is still time intensive, as a mesh of data is almost impossible to avoid. We argue that coping, after decades of research, with a huge mesh of data is a complex problem which requires simple solutions, and propose the need for an IR agent that can seek for information. The agent seeks in a mesh of data by interacting with other agents that have access to data stored in a repository, by searching in repositories, not violating FAIR principles, and by cooperating with a human that has an information need the agent aims to satisfy.

1.1. Complex Problems Require Simple Solutions

Many solutions, for problems that arise at managing data locally at repositories, exist to handle data according to the FAIR principles; however, these solutions are often not implemented, as they are widely unknown, not viable due to a lack of IT-knowledge, it is not obvious why the effort would be worthwhile, or ignored due to bad initial planning of a project. A lack of IT-knowledge can range from not being aware of how to use state-of-the-art solutions to a lack of knowledge of how novel solutions can be implemented to be used by others. The former is currently a subject of research in data science, hoping to bring developed solutions closer to IT specialists and end users. The latter requires to analyze the need for new solutions in practice and to implement them so that they are usable and therefore utilized by end users. Even implementing state-of-the-art solutions, especially in artificial intelligence (AI), to be used by IT-experts can be challenging as IT-experts are mostly not domain-experts, it is impossible for them to fully understand the needs and goals of the users of their developed solutions. Therefore, it is necessary to develop methods in AI that are usable directly by domain experts, i.e. end-users, which is the subject of current research in humanities-centred artificial intelligence (CHAI) (Möller, 2021). This can be achieved by implementing an agent that is perceived by its users in such a way that the agent can anticipate their goals. An agent can draw attention to itself and show its perception to the user by acting legible according to the user's expectations or by explicitly explaining its behavior if the gap between the user's expectations and the agent's capabilities is too large. Acting in such a way requires planning, including concerning the goals the agent itself is initially implemented for. Additionally, the agent needs to model and plan with respect to the current mental model of the user. Such an agent is referred to as a human-aware agent, and it can assist a human to handle a huge mesh of data by, for instance, collaboratively searching for information with the user (Kambhampati, 2020). In practice, not all problems need forcibly be solved by AI methods, as often combining existing approaches in a new manner will suffice.

1.2. Real World Observations

In this thesis, a real world research data life cycle from humanities scholars is examined at the Understanding Written Artefacts (UWA) excellence cluster, data linking infrastructure - foundations and architecture (DL01) project, part of the Center for the Study of Manuscript Cultures (CSMC) at the Universität Hamburg. Humanities scholars generate data by digitizing a set of palm leaves and paper manuscripts stored at the U. V.

Swaminatha Iyer Library in Chennai (India) by taking pictures of them (Subramanian, 2013). On the palm leaves and paper manuscripts poems are written in an ancient Tamil language dialect and almost all poems are present more than once. Pictures of the palm leaves are stored at the research data repository (RDR), managed by the Center for Sustainable Research Data Management (RDM) at Universität Hamburg, and access is restricted as publishing the pictures is forbidden if not permitted explicitly by the librarians. Stored data at the RDR have associated metadata for data governance and a unique identifier such that others can refer to them. All data that are created from the pictures are, if uploaded at the RDR, linked to them. Therefore, the data stored in the RDR are almost managed according to the principles of FAIR and CARE.

From these pictures, critical editions were and are created by many humanities scholars (Wilden, 2018). All textual variations found on pictures of the palm leaves and manuscripts are examined. Partially, the original texts are not recoverable, and humanities scholars can use a critical edition as a reference. By our observations, most critical editions are published as a printed book or as a PDF document, as a large percentage of the readers of an edition prefer that (Al-Shboul and Abrizah, 2014).

However, the content of a critical edition offers more than just to be read by humans, if it is machine interpretable. Data is machine interpretable, if it is encoded by markup languages such as XML. Despite the advantages of rewriting an edition to be interpretable by machines, the effort doing it manually would be too high, and automatic rewriting is impossible without the help of an IT-expert. IR systems would not only index a critical edition based on associated metadata, additionally the content of the edition itself is used, if it is interpretable. Additionally, interpretable data could be transformed into other formats, accessed via an API by an agent, automatically analyzed, and used as training data in AI (Schiff, Kuhr, et al., 2020).

Editions contain many links, explicit as well as implicit ones, that link related content with each other. Links are explicit, if agents can follow them automatically, and implicit if they are, among others, descriptions of other locations, written in natural language. Readers of a critical edition follow implicit links via the description. Depending on the description, one needs an IR agent to find mentioned locations in the data. Again, without the help of an IT-expert, scholars would spend to much time on creating explicit links in addition to implicit ones, and are possibly unable to do so due to lack of IT-knowledge.

Sometimes, humanities scholars working across time and geographical borders, produce related editions about the same set of digitized objects. Linking of such related content with each other is important, as readers are often interested on all contents, however it is not necessarily the only option. Instead of linking, contents could be combined before being published, such that readers do not have to always follow links. By

our observations, combining contents manually take humanities scholars weeks or even months of work, and might be prone to error. Instead, if data is machine interpretable, it could be combined automatically. Combined data is linked with contents from which it stems, and if there are changes in the content of the original data, the combination can be repeated automatically.

Editions contain a lot of information that could be used as evidence for many different purposes. However, not the entire edition is always relevant, and one would like to precisely refer to one ore more parts of it. Uniquely referring to one or more parts of an edition is impossible if only the whole edition is associated with a DOI. Instead, a repository, such as the RDR, should provide a preview of the data it contains. Given the previewer, one could allow for adding parts of viewed contents of an edition to a collection, and associate the collection with a DOI. We argue that the visualization of research data on the web only really makes sense with the possibility to uniquely refer to any visualized content.

All mentioned improvements cannot be realized without the help of an IT-expert, and cannot prevent a mesh of data, and it is not obvious whether such a mesh of data could be omitted at all. However, if projects maintain and manage repositories with respect to the FAIR principles, researchers searching for evidence in a mesh of data could be supported with an IR agent, seeking collaboratively with a human for information. Improvements need to be implemented, such that they are usable by humanities scholars without the help of an IT-expert.

Our observations stem not only from humanities scholars, additionally we have observed data life cycles from the medical domain. Medical data can improve healthcare by supporting decision making, and thereby to reduce costs, by using it for ML for training a model that can be used to predict certain outcomes. However, despite the benefits of providing medical data for research purposes, access is restricted by data protection laws. Some repositories exist that contain medical data, however most repositories are not publicly available, patients are anonymized, data are synthetic, or repositories contain not a single, a few or only historical data (i.e., recorded streams of timestamped data) at all. Depending on the research goal, each repository, containing medical data, has its own advantages, and disadvantages in contrast to others. For instance, it might be essential that patients are not anonymized for acquiring further measurements, that data could be published to be accessible by everyone for reproducing results, or that data contains both static as well as historical data. We propose that data needs to be combined from various sources if available data from a single source does not satisfy all requirements for problems to be solved, instead of searching for problems that can be solved with available data.

1.3. Related Work

Creating research data and finally publishing them often requires transforming them into various formats to be widely accepted. These formats are possibly not those the researcher is familiar with. Methods such as eXtensible Stylesheet Language (XSLT) (World Wide Web Consortium, 2022b) or Pandoc (MacFarlane, 2022; Dominici, 2014) are used to transform data from one format into others. If available and applicable, end users can use XSLT sheets to automatically transform their source data into data with a desired target format. Availability depends on both the source and target schema, and any modification on the transformation process requires knowledge of XSLT and is limited to XML formatted data as a source.

Pandoc is implemented in Haskell[1] to automatically transform data formatted in a specific format into others. As many formats are supported, the intermediate representation of data is limited and transforming data from one format into another is often not loss-free unless data in the source format is fully covered by Pandoc. Extending Pandoc either requires knowledge of programming in Haskell or using custom solutions based on Pandoc as, e.g., implemented by Krijnen et al. in Krijnen, Swierstra, and Viera (2014). We argue that transforming data from a specific format into another one needs to be done with care and has to be loss free. Solutions such as Pandoc are only usable by limiting the source format to those that are fully supported by the tool; otherwise, the transformation into the target format can be marked by data loss. As time of writing, Microsoft Word DOCX is not fully supported as colors, boldface etc., are lost during the transformation process.

The publication of research data itself in addition to a publication based on this research data is often mandatory nowadays and no longer simply good research practice. Additionally, in many research fields, publishing research data itself is a widely accepted result of research. It allows for critically assessing results based on research data and for reusing instead of recreating the data from scratch. Platforms such as Kaggle or Zenodo (Sicilia, García-Barriocanal, and Sánchez-Alonso, 2017; Peters et al., 2017) allow for sharing and finding research data. However, once research data is found, it is challenging to decide whether the data is relevant or not. The former allows viewing the first n rows of uploaded CSV files, while the latter only provides a preview of the file structure of uploaded zip files. To view the content of the data, they need to be downloaded in their entirety. Instead, more fine grained viewer are required to view the contents of the data stored at a repository. They need to be general enough to cover as many schemata of data as possible, as otherwise viewer need to be implemented for repositories individually, but

[1] `https://www.haskell.org/`

specific enough to make decisions based on visualized data. In some cases, depending on the data, it might be impossible to create viewer for all data even if they have the same schema. For instance, data encoded in XML associated with a specific xml schema definition (XSD) schema might not be visualizable with a generic viewer in any case, as some elements of the XSD schema are possibly designed to markup certain parts of the data that need to be visualized with respect to specific requirements determined by the author. For the author, a correct and precise visualization of her or his research data might be crucial.

Data stored in a repository and associated with metadata needs to be findable in a huge mesh of data. Due to the complex and diverse nature of a mesh of data, a human sends a query to an IR agent, and the IR agent assigns a score to each data point in a repository it has access to. Finally, the IR agent returns data with the top n highest scores in descending order to the human who gives optional feedback, given the data she or he has received. A human, however, might be unable to express her or his information need ad-hoc, within one step, and thus might send a reformulated query back to the IR agent. This loop is repeated until the human's information need is satisfied or either the IR agent or the human breaks the loop. An IR agent could greatly improve its performance if it accounted for previously received queries from a human with the same specific goal (Tang and G. H. Yang, 2022). A sequence of iterations, in which a human sends a query to an IR agent, the IR agent sends results back to the human, and the human returns optional feedback to the IR agent, is referred to as a session. Luo et al. in Luo, S. Zhang, and H. Yang (2014) and Luo, Dong, and H. Yang (2015), present an IR agent that improves its performance by direct policy learning, but they have not defined a policy for the human. Yang et al. in A. Yang and G. H. Yang (2017) showcase a scenario in which a human only gives feedback but does not change a query during a session that belongs to the same specific goal. We argue that both the human and the IR agent need to collaborate during searching for information. Kambhampati et al. argue in Kambhampati (2020) that an agent and a human with a specific goal can only cooperate if both are modeled with mental models but they have not specified these models any further formally.

At the time of writing, we have not found an optimal evaluation methodology for the evaluation of our human-aware IR agent that collaboratively searches for information with a human. The best fit was developed over years at Text REtrieval Conference (TREC) conferences for the evaluation of IR agents that search for information during a session. At the TREC interactive track between 1997 and 2002 (The TExt Retrieval Conference, 2002), the evaluation did not lead to a reusable test collection methodology (Kanoulas, Azzopardi, and G. H. Yang, 2018). This was followed by TREC Session Tracks between 2010 and 2014 (Information Technology Laboratory and Information

Access Division, 2014) in which metrics in the evaluation solely depend on the last iteration of each session. Evaluation was further improved at the Dynamic Domain Tracks between 2015 and 2017 (G. H. Yang et al., 2017) in which metrics for each iteration in each session were defined. The human simulator for the evaluation of an IR agent returns dynamic feedback but does not change an initially sent query to the IR agent during a session. Finally, at the Conference and Labs of the Evaluation Forum (CLEF) held between 2017 and 2018, the need for a dynamic human simulator was focused on. In 2018, no submissions were received, as setting up the lab required more time than anticipated (Kanoulas, Azzopardi, and G. H. Yang, 2018).

1.4. Contributions

In this dissertation, we identify problems that arise at managing data and at finding information in a mesh of data, and propose solutions that are implemented and evaluated. We summarize our contributions as follows:

(1) Transformation of research data Researchers have preferred tools and document formats in use for producing results, given evidence they have found. If results are meant to be printed later, preferred tools are often what you see is what you get (WYSIWYG) editors such as Microsoft Word, as one can always see the current state of the result as if it is printed. Nowadays, not only results derived from evidence found in data are published, additionally, raw data that stems from the digitization of objects, experiments, or observations is published as a result. Usually, huge amounts of raw data are created to be interpreted by machines and not to be read by humans, as humans would not read raw data anyways. However, it is beneficial if machines could interpret data that is meant to be printed later, and if humans could view raw data that is meant to be interpreted by machines. In the former, tools such as Pandoc can transform the format of data to be printed into a format, such as XML, which purpose is to encode data to be read by machines. However, such XML data produced by Pandoc contains mostly tags that have not any semantic meaning. We show that it is beneficial to manually implement parser for different types of research data for transforming them into formats, among others, to be interpreted by machines.

(2) Merging and combining research data Often, researchers work across time and geographic boundaries together, generating results containing common content to be merged before being published. Manually merging results can take several weeks or even months of work and might be prone to error. We implemented and

evaluated the automatic merging of results containing common content. Not only automatically merging research data takes only a few seconds, rules determining of how merging is done can be changed afterwards at ease, for merging the data differently than before.

(3) Automatic linking of research data Research data, regardless of whether it has been formatted to be readable by machines or humans, contains either implicit or explicit links. Links are implicit if they are, for instance, only mentioned in texts formatted to be readable by humans and not clickable using a viewer, such as a web browser, or if texts are in a specific format to be readable by machines; however, the parser designed for such formats is unable to identify the links. Thus, in the case of implicit links, more sophisticated methods are required. We automatically make implicit links explicitly available such that humans as well as agents can follow them. Readers of research data spend less time on searching for data if they can simply follow links by clicking on them.

(4) Viewer for machine readable research data Depending on the particular use case, machine-readable data is preferred over research data formatted to be readable by humans. For instance, in ML, data scientists spend 80% of their time on data preparation and 20% on model implementation and deployment (Hameed and Naumann, 2020). Data preparation is necessary if data is formatted to be readable by humans rather than structured to be readable by machines. However, even researchers in ML prefer not to download research data from a server and implement a viewer to view the contents of the data to decide whether or not the data fits their needs. Hence, we implemented viewer for different kind of research data, for supporting researchers in decision making. Our viewer are different from tools that aim to transform data from any domain formatted to be interpreted by machines into formats, such as HTML to be rendered by a web browser. We show that viewer need to be often domain specific with respect to specified requirements by those who have created the data, and that viewer are implementable such that they can be easily adapted to visualize similar data as well.

(5a) Sharing archived research data Often a complete repository of research data is cited via a unique identifier associated with the repository, even if only a small part is mentioned with respect to the citation. Instead of citing research data at a repository, we argue that research data needs to be shareable by creating unique identifiers directly for and at a viewer for research data. Researchers can directly view cited content by following a unique link associated with a specific part of research data uploaded to a repository instead of downloading and searching for

the mentioned subset in the citation. We extend our viewer by enabling readers of textual content to share them via a DOI.

(5b) Annotating and sharing regions of images Sharing research data directly from a viewer is and should not be limited to texts. We extend our approach of sharing research data visualized by our viewer to images. Humanities scholars can highlight specific regions in images, comment on them, and share them as well as texts.

(5c) Collections of regions of interest (ROI) Regardless of whether links were created for texts or images, readers might prefer to share a set of links instead of only one. We further extend our viewer by allowing for the creation of collections of links. Each collection has an associated DOI that changes if the corresponding linked collection is changed and refers to each collection as ROI.

(6a) Human-aware information seeking using collections Human-aware information seeking is the collaboration of a human and an IR agent in searching for information during a session. A session is a sequence of iterations in which a human sends a query to an IR agent and the IR agent sends results to the human based on the query. Collaboration requires that the IR agent and the human are modeled with mental models (Kambhampati, 2020). We use collections of ROI that link to specific textual paragraphs of research data, a query, and a session of iterations to model the information need of a human part of her or his mental model.

(6b) Identify inline subjective content descriptions (iSCDs) Collections of ROI are not always available. However, research data might contain subjective content descriptions (SCDs) in form of texts that are located within the research data, adding additional information (Kuhr et al., 2019). These are subjective, as different humans create different content descriptions (CDs) for the same data. SCDs are comparable with post-it notes attached to a printed document. Sometimes, SCDs are interleaved with research data so that they are not differentiable from the rest of the data. In that case, a human-aware IR agent can only make use of SCDs by identifying them. We identify such inline subjective content descriptions (iSCDs), using a trained hidden Markov model (HMM) and, in our evaluation, show that it is indeed possible to distinguish iSCDs from research data using our approach (Bender et al., 2021).

(6c) Evaluation of a human-aware IR agent A human-aware IR agent that collaboratively searches for information with a human needs to be evaluated. We have not found an evaluation methodology for the evaluation of a human-aware IR agent

in the literature and thus propose a new approach in which a simulated human interacts dynamically with the IR agent that is to be evaluated (Schiff and Möller, 2021).

(6d) Synthetization of relevant document and query pairs Most existing evaluation methodologies that fit the evaluation of our human-aware IR agent best rely on data that is not freely available. We present a new approach based on wiki data from the Wikimedia foundation that is freely available.

1.5. Structure

Chapter 2 introduces commonly used terms in data management and tools we have found, being used in practice by researchers, for managing research data stored in a project's repository. In Chapter 3 we show how to transform research data from one format into others that are either readable by humans or interpretable by machines. Our transformation approach is compared with the most similar tools we could find, such as Pandoc. If research data is available in a machine interoperable format, more opportunities are offered, which we present in Chapter 4. Among others, we show how research data can be, depending on the use case, automatically linked or combined with each other, and then visualized by a viewer. Data visualized by a viewer helps to support deciding whether data is relevant or not. However, it lacks of the possibility to uniquely refer to specific locations in the data. Hence, we present in Chapter 5 a solution for sharing data visualized within a viewer. Chapter 3, Chapter 4 and Chapter 5 were mainly published in:

> Simon Schiff, Felix Kuhr, Sylvia Melzer, and Ralf Möller (2020). "AI-based Companion Services for Humanities". en. In: *KI2020*. URL: https://www.ifis.uni-luebeck.de/uploads/tx_wapublications/SchiffKuhrSylviaMoeller-Extended-Abstract_01.pdf (visited on 07/31/2023)
>
> Simon Schiff, Sylvia Melzer, Eva Wilden, and Ralf Möller (May 22, 2022). "TEI-Based Interactive Critical Editions". en. In: *Document Analysis Systems*. Ed. by Seiichiand Barney Uchida and Véronique Elisaand Eglin. Springer International Publishing, pp. 230–244. ISBN: 978-3-031-06555-2. DOI: 10.1007/978-3-031-06555-2_16
>
> Simon Schiff, Marcel Gehrke, and Ralf Möller (2018). "Efficient Enriching of Synthesized Relational Patient Data with Time Series Data". en. In:

Procedia Computer Science 141, pp. 531–538. DOI: 10.1016/j.procs.2018.10.130

So far, our solutions solve problems that arise at managing data at a repository which does not prevent in the long run a mesh of data. Hence, we present in Chapter 6 an IR agent that cooperatively searches for information with a human in a mesh of data. Chapter 6 was mainly published in:

Simon Schiff and Ralf Möller (Sept. 28, 2021). "On Human-Aware Information Seeking". en. In: *Proceedings of the Workshop on Humanities-Centred Artificial Intelligence (CHAI 2021)*. CEUR Workshop Proceedings, pp. 31–39. URL: https://ceur-ws.org/Vol-3093/paper4.pdf (visited on 07/31/2023)

Simon Schiff, Magnus Bender, and Ralf Möller (Sept. 19, 2022). "Embodiment of an Agent by a Pepper Robot for Explaining Retrieval Results". en. In: *Proceedings of the Workshop on Humanities-Centred Artificial Intelligence (CHAI 2022)*. CEUR Workshop Proceedings, pp. 29–37. URL: https://ceur-ws.org/Vol-3301/paper4.pdf (visited on 08/01/2023)

Magnus Bender, Tanya Braun, Marcel Gehrke, Felix Kuhr, Ralf Möller, and Simon Schiff (2021). "Identifying and Translating Subjective Content Descriptions Among Texts". en. In: *International Journal of Semantic Computing* 15.04, pp. 461–485. DOI: 10.1142/S1793351X21400122

Finally, in Chapter 7 we conclude this work and given an outlook for further research directions. Chapter 7 was partially published at a workshop:

Simon Schiff and Ralf Möller (2023). "Persistend Data, Sustainable Information". en. In: *Proceedings of the Workshop on Humanities-Centred Artificial Intelligence (CHAI 2023)*. to appear. CEUR Workshop Proceedings

2. Preliminaries

A *desired* research data life cycle should contain at least six steps, as depicted in Figure 2.1. Often, *bad* research data life cycles contain only three of these steps: The creation of the research data from scratch, the publication of results, and research data no longer being accessible for verification of results or reuse. Although a *bad* research data life cycle is cheaper at first sight than aiming to realize a *desired* one, there are many other reasons why research data often passes only three steps of the research data life cycle. One of the reasons for this is a *bad* research data life cycle often not being considered as bad or the realization that a *desired* one is possibly not viable. The latter is often caused by a lack of IT-knowledge, resulting in an overhead of work needing to be done. For example, documents required to be published in different formats by funding agencies are often formatted manually instead of automatically due to a lack of IT-knowledge. It might be mandatory for metadata to be created and associated with research data. Deciding which format to use for widely accepted, sustainable, and easy-to-process metadata is challenging for non-IT-experts and is created manually if not supported by a program. Anything done manually costs time – thus, it is oftentimes avoided or only done to a minimal extent. As we have observed, many problems involved in converting a *bad* research data life cycle to a *desired* one can be solved by an IT-expert and are partially implementable by the composition of freely available libraries.

Before we start identifying problems that are solvable by an IT-expert and implementing and evaluating them with respect to a real-world research data life cycle from the humanities, we begin by defining research data in more depth. After, we present existing standards, data formats, and implementations to move towards a good research data life cycle and show why these are only partially used in practice. Finally, we present libraries and frameworks that we use in our implementations to enable humanities scholars to realize a *desired* research data life cycle by making existing standards and data formats accessible to them.

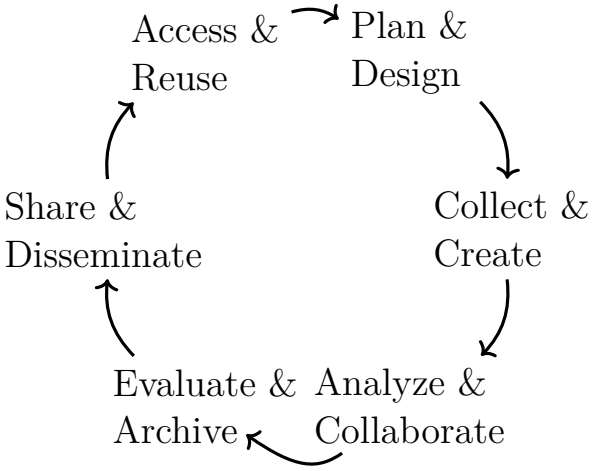


Figure 2.1.: Research Data Life Cycle (Longwood Research Data Management, 2022)

2.1. Research Data

Research data is produced either by digitization or born digital. Digitization is the process of transforming an arbitrary object into a digital representation encoded binary from a low-level perspective (Saima Khan, Shazia Khan, and Aftab, 2015). Sampling is required for digitizing objects, as binary encoding is discrete and not continuous. Data that stems from the digitization of objects should represent them as well as possible in case of the objects no longer being accessible. As sampling is required, digitization is not loss-free and needs to be done with care. For instance, without any specific goal, it is challenging to digitize an object appropriately and in a way that prevents it from becoming inaccessible later, as it may degrade over time. It might be sufficient and cheap to take low-resolution pictures of an object, allowing text written on it to be readable. Later, the object may no longer be available and one may be interested in the material of the object – it will, however, be lost as it was not examined during the digitization. Hence, it is important to thoroughly plan the digitization of objects in advance (K. Thiemann, 2019). Additionally, choosing an appropriate format for digitized objects is important from the beginning, as other users may otherwise be unable to use the data or spend time on data formatting before they can use the data for their own purposes. In case of text, one format amongst the many existing and frequently used ones is XML. Data formatted in XML is easier to process than data formatted in an arbitrary format, as XML is standardized by the World Wide Web Consortium (W3C) (World Wide Web Consortium, 2022a).

2.2. Extensible Markup Language

XML is designed to exchange information via the web by encoding information to be readable by machines and humans. It is widely accepted and most programming languages are shipped with libraries for producing and reading documents encoded in XML. Exchanging structured documents in specific fields requires not only documents encoded with respect to a standard, such as XML, but additionally needs the documents to have a specific schema that can be validated. For instance, a system reads metadata encoded in XML associated with a publication to display information such as the publication's authors. Without any schema provided for the metadata encoded in XML, the system is unable to distinguish tags possibly surrounding the name of an author, making it impossible to distinguish tags such as `<author>` from `<humba>` or `<kalumba>`. Thus, xml schema definition (XSD) is used to specify the schema of a document, itself encoded in XML, that can be validated. Various XSD schemes have been invented in the last decades, such as Office Open XML (OOXML) or Metadata Encoding and Transmission Standard (METS) and billions of documents worldwide are formatted accordingly. Research data, as well as associated metadata, is often encoded in XML. The former by, for instance, Text Encoding Initiative (TEI) and the latter by METS.

2.3. Metadata Encoding & Transmission Standard

The Metadata Encoding and Transmission Standard (METS) standard is developed by the METS board in collaboration with the Library of Congress. Metadata is encoded in XML and its schema is specified according to METS for interoperability. METS is recommended when research data is uploaded as archives at a RDR for long-term preservation and is mainly composed of seven sections (The Library of Congress, 2022):

METS Header The METS header `<metsHdr>` contains metadata about the METS file, such as the last modification date. Additionally, it contains a list of authors and one or more unique identifiers. Each author in the list has an associated role and a type which is important for distinguishing them. Roles are restricted to a certain set of predefined ones such as ARCHIVIST or CREATOR, where ARCHIVIST is the creator of the METS file and CREATOR is the creator of the actual research data that the METS file is associated with. Each role has an associated type that is restricted to either INDIVIDUAL, ORGANIZATION, or OTHER. Unique identifiers help to find and therefore access the METS file.

Descriptive Metadata Zero or more descriptive metadata `<descMD>` sections are al-

lowed and each contains <dmdSec> elements that are either pointers to external metadata <mdRef> or internally embedded metadata <mdWrap>. External metadata can be formatted in various formats such as dublin core (DC) or data documentation initiative (DDI) and is not limited to being encoded in XML. Internal metadata is encoded in either XML or in any binary form. Each <dmdSec> element has a unique id ID to link descriptive metadata with parts of data associated with the METS file and the contents of a <dmdSec> element are used for discovery and identification.

Administrative Metadata Zero or more administrative metadata <amdSec> sections are allowed and each contains either technical metadata <techMD> about the data such as formatting, intellectual property rights <rightsMD> such as licensing, source metadata <sourceMD> about objects that are associated with the data, or digital provenance metadata <digiprovMD> to trace back from where data stems from.

File Section The file section <fileSec> contains one or more file group <fileGrp> elements. Each <fileGrp> is used to separate file <file> elements into meaningful groups such as by format or collection. A <file> element itself has a unique id ID and either a <FLocat> or <FContent> element. The former is a reference to the location of an object encoded as, for instance, a Uniform Resource Locator (URL), and the latter contains the actual content of the object either encoded in XML or binary. The latter is not recommended by the FAIR guidelines as metadata should remain available even if the associated data is missing. Hence, data and metadata should be stored separately.

Structural Map The structural map <structMap> section contains a hierarchy of nested elements, each having a unique ID. It determines how data is presented to users, for instance, as a file browser, and for navigating through the data that is associated with the METS file. Each element may contain either again a element, a METS pointer <mptr>, or a file pointer <fptr>. A METS pointer points to another METS file and the file pointer points to a <file> element via its ID. Pointers to other METS files are useful to keep a METS file small if data associated with it contains many objects.

Structural Links The structural links <structLink> section contains zero or more link elements <smLink>. Each link element has, among others, the attributes xlink:from and xlink:to pointing to the source and the target ID of elements in the structural map section.

Behavior The behavior section `<behaviorSec>` element contains one or multiple `<behavior>` elements linking executable code with data linked in the structural map of the METS document. For instance, if a user clicks on an image in a viewer of the contents as defined in the structural map, the behavior is defined by the `<behavior>` element. In case of an image, a viewer displaying the image could pop up. Each `<behavior>` element contains a `<interfaceDef>` that is an abstract interface definition of the behavior and a `<mechanism>` element. The `<mechanism>` element points to a module of executable code that implements and runs the behavior defined abstractly by the interface definition. Each `<behavior>` element has the attribute STRUCTID that links to one or more `<div>` elements within a structural map.

2.4. Text Encoding Initiative - Format

Research data should be encoded in a format that is easy to be processed and analyzed for extracting information or decision-making and reuse. Deciding on how data should be encoded during its creation is challenging as it is often impossible to initially identify all use cases and requirements by later users. Thus, using widely used and accepted standards such as XML with an associated XSD schema maintained by a large community is beneficial. One of the schemata for encoding research data is Text Encoding Initiative (TEI). TEI is mostly used by scholars and librarians and tools exist to produce, manipulate and read TEI documents. Among many XSD schemes, TEI is shipped with schemes for dictionaries, critical apparatuses, and language corpora (Text Encoding Initiative, 2022b). Throughout this dissertation, we present examples of TEI documents that are validated against the TEI schema encoded in XSD.

2.5. Unique Identifier

Unique identifier (UID) are primarily used to discriminate objects from each other and to refer to them via the associated UID. For instance, a database may contain a large relation of patients who have visited a hospital. The schema of the relation contains the name and gender of the patients. It is impossible to refer to one hospital stay in the relation without a UID, as name and gender of a patient are identical for each hospital stay. This can be solved either by adding the date of the hospital stay, assuming that there is no time point at which two different patients with the same name and gender visit the hospital or by adding a UID to the schema of the relation. The former might

work if the assumption is correct; however referring to a hospital stay of a patient requires name, gender and date instead of using a single UID. This is identical to the problem of discriminating publications and research data. Among other data points, a publication is likely to have a title, list of authors, abstract, proceedings, and an optional revision number. A lot of information needs to be provided to refer to a specific publication. In the case of research data that usually has a title, description and content, only the content may differ from other research data with an identical title and description. For instance, a very popular research data set for ML may be downloaded, modified, and re-uploaded somewhere else without referring to the original data due to the lack of a UID. Re-uploaded data could then possibly be confounded with the original data. Standards, such as DOI or the well-known ISBN, have emerged to discriminate objects. ISBN is used to discriminate books worldwide, and everyone is able to refer uniquely to a book without risk of confusion. DOI is generally used to create unique identifiers for any object and is widely accepted. It is resolved via a name resolver and a table is returned containing keys and values in a one-to-one relation. Keys are, for instance, XML, TEI, URL, or DOI itself. Depending on the use case, the table is returned or one of the keys is used for a specific purpose. A publication could contain a DOI referring to a dataset. The DOI is linked to a table that contains both an URL key referring to the dataset and a XML key with a XML document as value that contains metadata that is associated with the dataset. If a user hovers over the DOI with a mouse, a client decides to display metadata from the XML document, and if it is clicked, the dataset is downloaded. Additionally, a returned DOI key could point with its DOI as value to a newer revision of the dataset. It is mandatory that a DOI always refers to the same object. If an object changes, then the DOI also needs to change. Metadata is standardized, for instance, by METS, to be readable by any client.

2.6. Research Data Repository

The research data repository (RDR) is hosted at the Universität Hamburg and managed by the Center for Sustainable Research Data Management (RDM). It is a repository containing research data created and uploaded by, different research groups, e.g., humanities scholars, and is based on Zenodo. Zenodo itself is an open-source platform, based on the Invenio digital library that is also open source and developed by the European Organization for Nuclear Research (CERN) (CERN and OpenAIRE, 2013). The platform enables researchers worldwide to share their research data with others associated with a license and a DOI.

Uploading at the RDR is done by accessing it via a web browser and then uploading

ID	Property
1	Identifier (with mandatory type sub-property)
2	Creator (with optional given name, family name, name identifier and affiliation sub-properties)
3	Title (with optional type sub-properties)
4	Publisher
5	PublicationYear
10	RessourceType (with mandatory general type description sub-property)

Table 2.1.: Mandatory fields of the DataCite.org scheme (DataCite Metadata Working Group, 2021)

research data from local storage. Uploading files requires providing additional metadata that is associated with the uploaded research data. Metadata is encoded in XML and its schema is standardized by DataCite.org (Sicilia, García-Barriocanal, and Sánchez-Alonso, 2017). Some metadata information is mandatory as listed in Table 2.1, while other data points are optional. Mandatory metadata is used by providers of repositories, such as the RDM, for representing uploaded datasets and to enable others to find them. Each uploaded dataset at the RDR has an automatically created DOI that is associated with it. It is impossible to change the uploaded data afterwards without changing the DOI. Thus, a DOI is always associated with the same immutable data and therefore citable. Otherwise, published results based on public research data are not verifiable.

The RDR provides a very useful function by enabling an uploader of research data to link it to other uploaded data via, among others, a DOI, ISBN, Uniform Resource Name (URN), or URL. Links, such as “cites this upload”, “compiles/create this upload”, or “is identical to this upload”, are very informative and useful in several aspects. One can determine the research data life cycle from its origin to its current state. This helps to retrace how data is generated and potentially assess its trustworthiness. Additionally, search engines can benefit from how data are linked with each other for computing retrieval results given a query. A system can recommend other data related to the data currently displayed.

As depicted in Figure 2.2, each dataset at the RDR is displayed by its metadata, including a preview that lists the files that are part of uploaded archives. Archives contain research data of arbitrary formats, such as CSV, TEI, Epigraphic Documents in TEI XML (EpiDoc), Microsoft Word DOCX, or International Image Interoperability Framework (IIIF). The preview function in Figure 2.2 is a generic one and additional

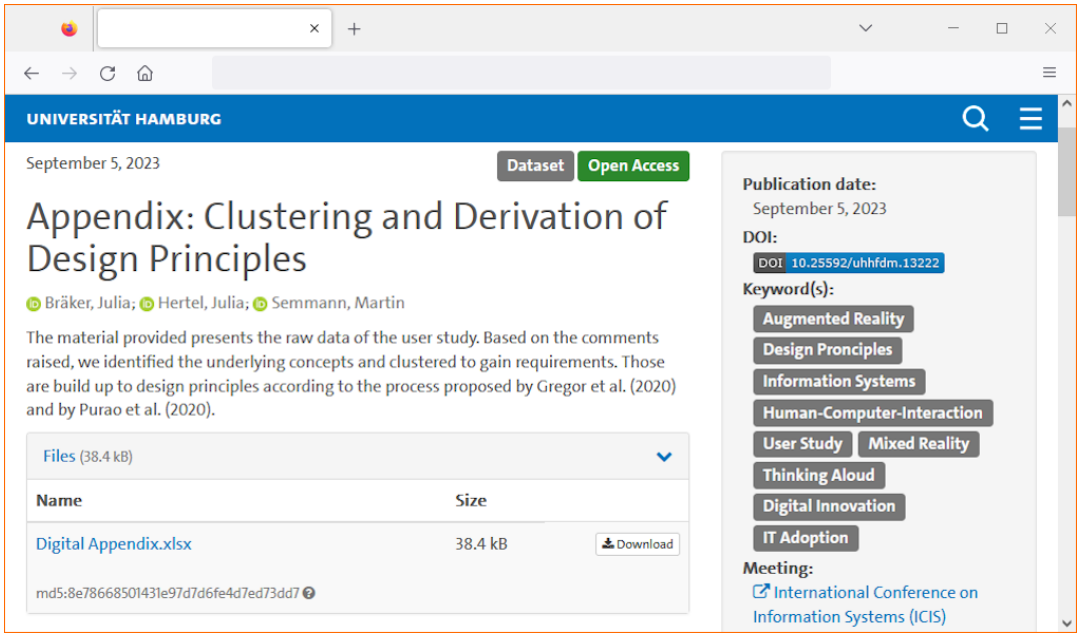


Figure 2.2.: Uploaded dataset at the RDR

knowledge is required about the data to ensure it is presented in a manner that meets the expectations of the uploader. Currently, the RDR does not provide any functionality to extend available previewer without changing and recompiling the underlying code. Hence, uploaded archives are previewed by listing only all files they contain. If one wants to access one of these listed files to decide whether it is relevant, then one needs to download the whole archive.

2.7. Heurist

Heurist is a research-driven data management system that allows non-IT-experts to first design databases based on their research data, then modify the database by, e.g., importing existing data or inserting data manually, and finally publish it. It is designed, in particular, for humanities scholars, as it is shipped with various predefined entities, data types, data formats, and relationships between entities. Research data of tabular form, such as XML, CSV, JavaScript Object Notation (JSON), IIIF, or Keyhole Markup Language (KML) can be imported. Entities that are part of the uploaded data can be mapped to predefined or manually created entities. MySQL is used as a back-end by Heurist for storing data, as it is open source, widely used, and data remains accessible

even if Heurist is no longer supported. Databases are publishable through the creation of a database dump, by using a website builder for the creation of a website on which the data is accessible, or by exporting the data in various tabular formats (HEURIST, 2022).

3. Multi-Target Publishing

Research data is generated either by digitization or virtually as born digital data (Society of American Archivists, 2023). Depending on the objects that are to be digitized or programs that produce data from scratch, research data can take many forms, such as a text file (TXT), CSV, Microsoft Word DOCX, or JPEG. In a bad research data life cycle, data is produced and analyzed to make decisions or finding out things, results are published, and the underlying data is kept on local storage where it cannot be reused by others. Sharing reusable research data by, for instance, uploading it to an RDR has several advantages. Other researchers can verify results by accessing associated research data used for producing the results or reusing it themselves. If research data is reused, those reusing it save resources as they do not have to produce the data by themselves while also citing the data, which is beneficial to those who upload it. However, those who produce the research data mostly decide to use formats that fit their own purposes or that they are familiar with. These formats may not be ideal for reuse by others and thus may need to be transformed into other formats prior to being uploaded to an RDR. Transformation depends on, among other things, the original format of the research data that is to be transformed, whether the transformation needs to be lossless, and whether additional information is needed. For instance, transforming images from TIFF to JPEG is not loss-free and additional information is required when transforming a TXT document into HTML. The latter is necessary, as HTML is a markup language while TXT is not. Those who aim to publish their research data must, right from the beginning, choose a data format that satisfies their needs and those of researchers who may wish to reuse the data. However, one might have no expertise in choosing the right format or no control over how data is generated. The challenge lies in choosing a source format that is suitable for one's own purposes and can later be transformed into different target formats. Ideally, the transformation is not conducted manually, as such a process can be very costly, error-prone, and laborious. Publishing results that are automatically formatted in different formats is called multi-target publishing.

One proposed solution is to use Pandoc for multi-target publishing (Kielhorn, 2011). The idea, in a nutshell, is to limit the creation of research data to a format that is fully supported by Pandoc. By our observation, it is mostly not applicable to limit the creation of research data in such a way; however, if it is applicable, then Pandoc is a

very powerful method to publish research data in many formats. Our observations stem from a real-world research data life cycle from humanities scholars that we present in Section 3.1. Problems are identified, and solutions are implemented and evaluated.

3.1. Real World Research Data Life Cycle from the Humanities

In the data life cycle we studied, we worked together with Eva Wilden, a humanities scholar with interests in classical Tamil literature and poetics, interaction between Tamil and Sanskrit, and manuscript studies. The goal by Eva Wilden, is to produce a critical edition, such as Wilden (2018) we present in Section 3.2, from palm leave bundles and paper manuscripts stored at the U. V. Swaminatha Iyer Library (UVSL), as depicted in Figure 3.1. Pictures are imported into the RDR managed by the RDM at the Universität Hamburg. Metadata is associated with the pictures and encoded in XML with a schema from DataCite.org. From these pictures, Eva Wilden creates a critical edition stored at the RDR. Associated metadata contains, among other data points, links to the pictures from which the critical edition is created. Finally, the critical edition is exported either as a PDF or printed as a book. We argue that this violates the FAIR principles and propose a solution, as depicted in Figure 3.2, which does not change the existing ETL presented in Figure 3.1. Changing the existing ETL forces users to change their preferred tools and document formats. Hence, solutions may not be accepted and therefore not used. Our ETL extension, we present in Section 3.3, starts by loading documents stored at the RDR and then extracting and loading the contents into a PostgreSQL database (The PostgreSQL Global Development Group, 2023). The parsing and extracting are done by an automatically created Python program from an Antlr4 grammar. As we show in Section 3.4, data stored in a database can mainly be exported into any format by using either structured query language (SQL), existing libraries, or custom-implemented programs that access the data stored in the database via an API. Hence, our ETL extension enables scholars in the humanities to continue working with their preferred tools and documents and export their research data into various formats. Exported data could then be reused beyond the field of humanities. Additionally, the data could be used in data sciences as training data and thus provide new solutions for humanities scholars working on a critical edition (Bender et al., 2021). Exported data is written back to the RDR and associated with a METS document. The METS document contains, among others, a mapping between TEI documents, and executable code that displays the document so that it is presented in a human-readable form. Thus, our ETL extension

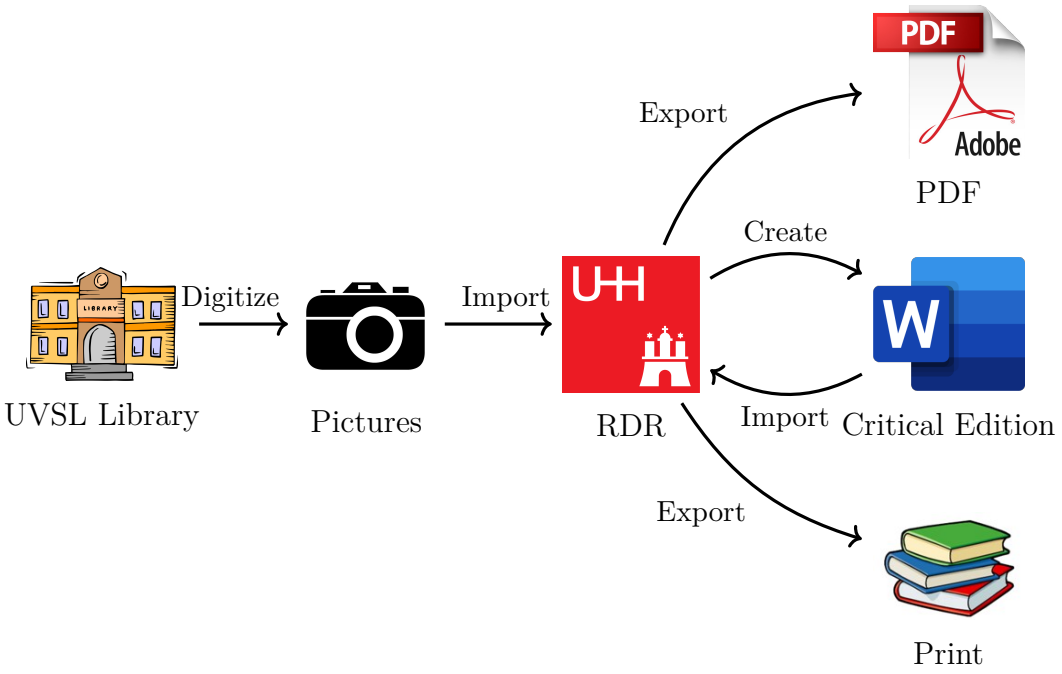


Figure 3.1.: Existing extract transform load (ETL)-pipeline

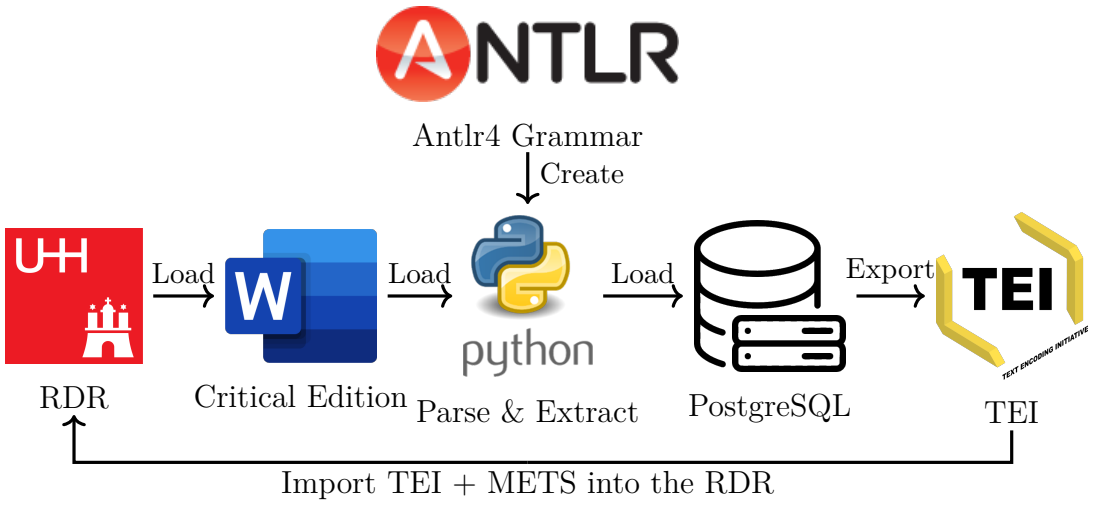


Figure 3.2.: Proposed ETL extension

converts the existing data ETL pipeline into a life cycle as it becomes more likely that others will reuse the data. Finally, we compare our ETL extension in Section 3.5 to Pandoc.

3.2. Critical Edition Created by Eva Wilden

The U. V. Swaminatha Iyer Library (UVSL) library in Chennai contains 2.200 palm leaf bundles, 850 volumes of paper manuscripts and 33.000 books. Each palm leaf bundle consists of 10 to 1.000 palm-leaf manuscripts (Subramanian, 2013). Palm leaves were one of the earliest forms of writing material in South and Southeast Asia (Shi, Setlur, and Govindaraju, 2004). A writer uses a stylus to carve letters into the leaves and soot is applied to the surface to stick in the grooves. The excess soot is wiped off, and the palm leave manuscript is complete (Padmakumar et al., 2003; Kumar, Sreekumar, and Athvankar, 2009). Humanities scholars are interested in these manuscripts for various reasons:

Access Manuscripts stored at the library are only accessible on-site as they are not digitized. Not everyone is allowed to access them on-site and gaining access depends on whether one has a good reason to do so. However, even a good reason may not guarantee access. Humanities scholars are interested in making the texts accessible by publishing a transcription.

Understanding The texts on the palm leaves are traditional poems written in the Tamil language. Both the font and the language have changed over time, making it difficult for beginners to understand the texts. Often, no literature exists that helps learn the language which was used centuries ago. Humanities scholars translate the texts into English, allowing everyone to understand the contents of the texts written on the palm leaves.

Preservation Palm leaves inscribed with soot age over time. The soot needs to be renewed, and oil and lemongrass are to be applied regularly. Renewing the soot helps ensure the readability of the carved letters, and applying oil and lemongrass prevents the manuscripts from being eaten by worms or silverfish. However, it is necessary to do so regularly; something that is not always done. Humanities scholars preserve the texts by transcribing them.

Reconstruction If parts of a palm leaf are damaged, one can consult an edition created in the past that shows the palm leaves in a previous, undamaged state, as depicted

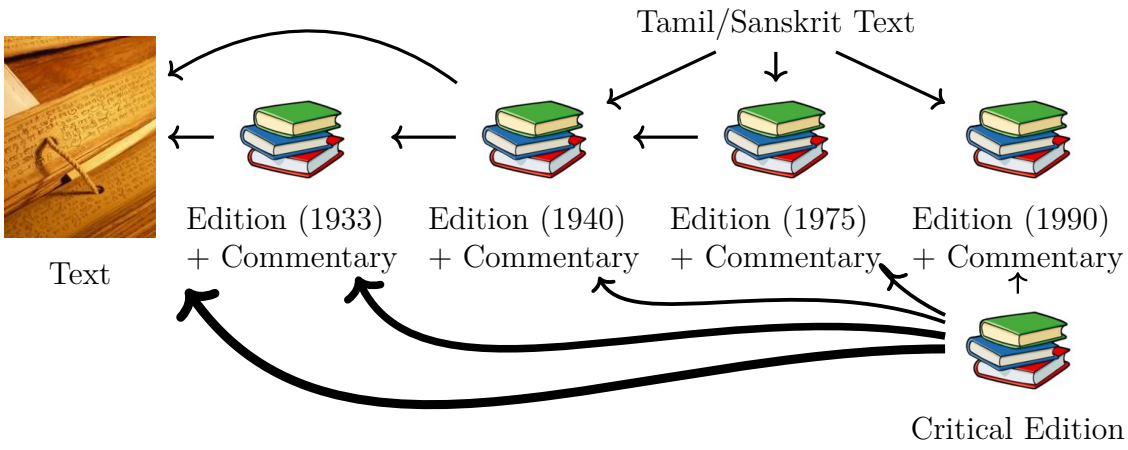


Figure 3.3.: Contents of a critical edition

in Figure 3.3. The left-hand side of Figure 3.3 depicts a palm leaf bundle containing the original texts. Four editions, created around 1933, 1940, 1973, and 1990, have been written and give evidence of the original texts. Old editions are preferred over newer ones, as they were written at a time when the palm leaves were less damaged than today and the language they are written in is not mixed up with other languages such as Sanskrit. Not all editions of the same lost text are consistent, so a unique, original text no longer exists. Humanities scholars "critically" go through all manuscripts and editions and document them as a critical edition.

A critical edition is the reconstruction of a specific text based on all variations of it found on objects created in the past (i.e., witnesses). The reconstruction of the texts written on the palm leaves is usually conducted by humanities scholars, is very cost-intensive and can take several years. The effort, however, pays off, as a critical edition is as complete as possible and helps humanities scholars gain a deep understanding of the underlying texts. In addition, as the scholar is able to go through all variants of the texts, including guidance from comments from the authors of the critical edition, the scholar is able to form her or his own opinion about the most probable variant of the text. A completed critical edition needs to be printed as a book, as humanities scholars often prefer libraries of printed books while looking for new information (Al-Shboul and Abrizah, 2014). Printing a book requires a document of any format in which headings etc., are well formatted. One example of a format that is not suitable for the printing of documents is TXT. The heading of a chapter is not differentiable from the rest of

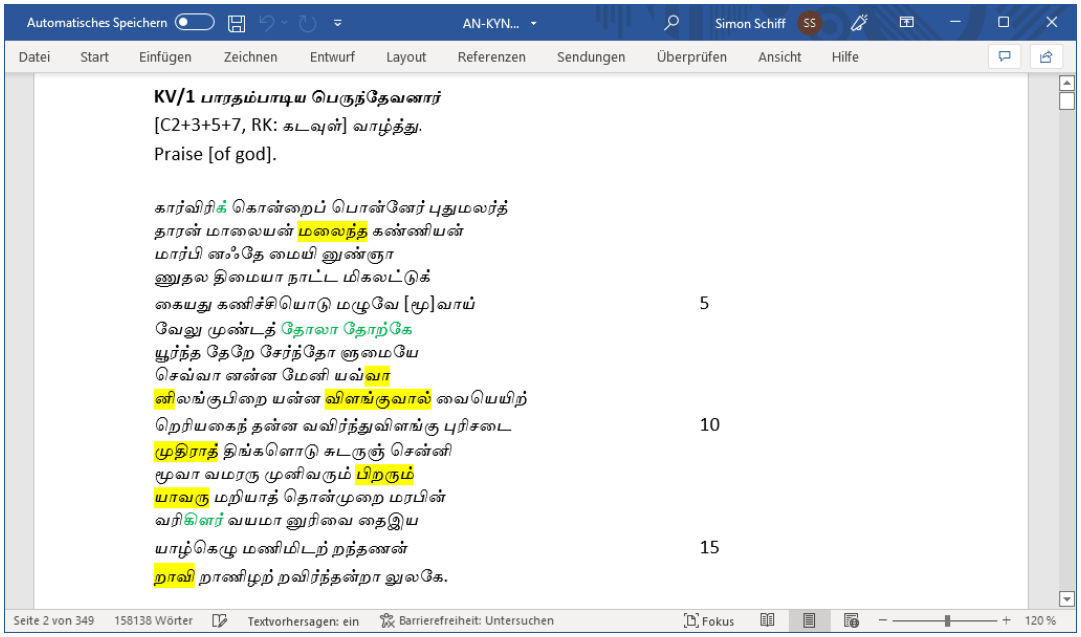


Figure 3.4.: Tamil poem as part of a critical edition written in Microsoft Word

the text of the chapter in a TXT document. Therefore, markup languages such as XML, in which headings are highlightable, are preferred. Manually producing an XML document is laborious and error-prone. If the XML schema is not supported by any program, then it should not be printed plain as reading an XML document is normally intended for maschines rather than for end users. A program that can produce printable documents, such as a PDF from XML by rendering it, is Microsoft Word. The schema of the documents in DOCX format is OOXML which is supported by Microsoft Word as well as other open-source programs. An advantage of these kinds of programs is that their graphical user interface (GUI) provides a WYSIWYG editor. Equipped with a WYSIWYG editor, the author sees a preview of the printed version of her or his text during the writing process.

The examined critical edition, created by Eva Wilden, was produced using Microsoft Word (Wilden, 2018). It contains 120 poems transcribed from images of palm leaves and paper editions described in Tamil, including (i) a transliteration, (ii) an apparatus containing variations in the texts, (iii) an interlinear translation into English, (iv) and a translation into English. The palm leaves stem from the UVSL in Chennai.

Figure 3.4 contains the first poem KV/1 in Wilden (2018). Some parts of the poem are high-lighted in colors, namely in green (primary variant) and in yellow (secondary

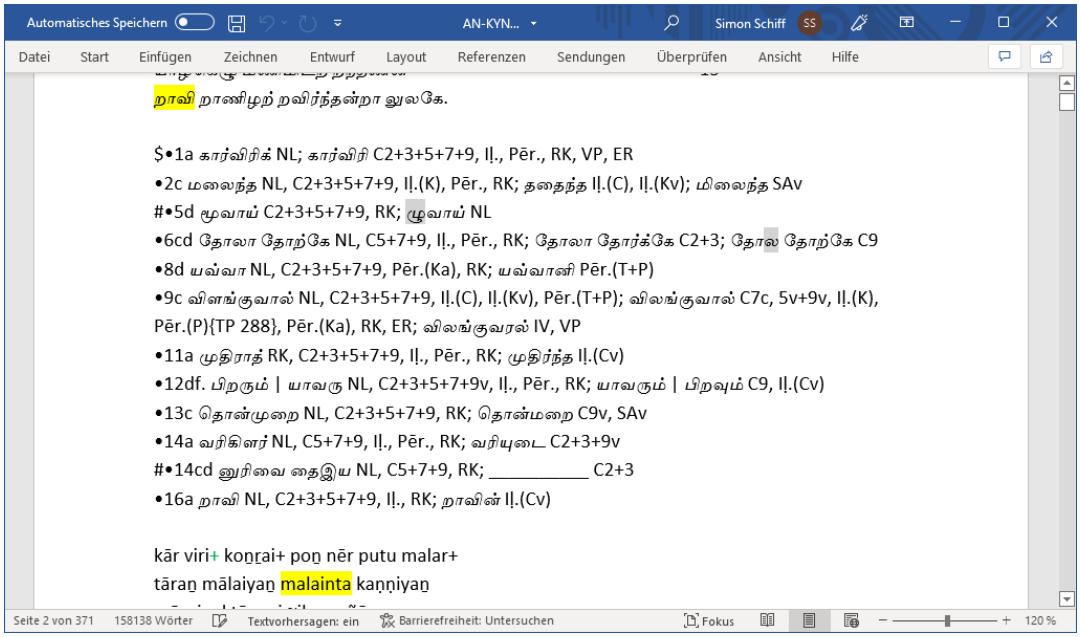


Figure 3.5.: Apparatus part of a critical edition written in Microsoft Word

variant). The primary variant shows a variation between different palm leaves containing the same poem, while the second variant means that there is a variation between paper manuscripts. Even if not depicted in Figure 3.4, words with variations between different palm leaves and paper manuscripts exist, too.

Figure 3.5 contains the apparatus for the first poem KV/1 in Wilden (2018). While the poem depicted in Figure 3.4 is the most likely variant of the poem, one can reconstruct all possible variants of the poem with the apparatus. For instance, the first line of the critical apparatus in Figure 3.5

$\$ \bullet$ 1a **கார்விரிக்** NL; **கார்விரி** C2+3+5+7+9, Iḷ., Pēr., RK, VP, ER

has a specific syntax, is located at a specific word in the poem, and means:

$ It is sandhi and/or a metrical variant.

- The beginning of a variant.

1a The first word in the first line of the poem.

கார்விரிக் The word in the poem that is the most accurate one.

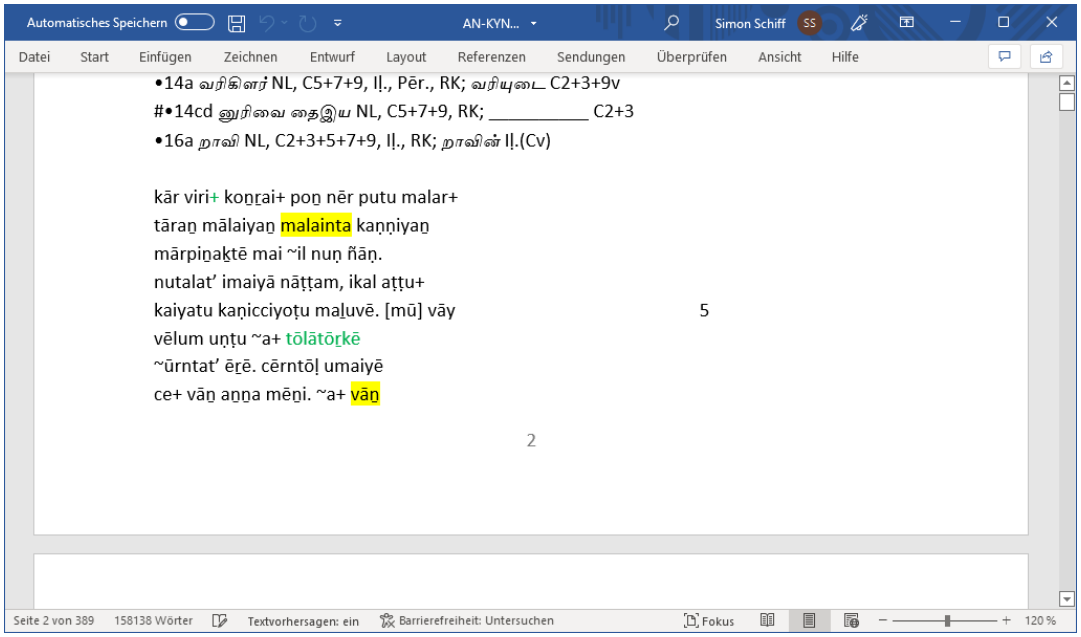


Figure 3.6.: Transliteration part of a critical edition written in Microsoft Word

NL A palm leaf manuscript from the National Library in Kolkata in which the first word in the first line of the poem is **கார்விரிக்**.

; End of the variant and the start of another one.

கார்விரி Another possible variant for the first word in the first line of the poem.

C2+3+5+7+9 is an abbreviation for C2, C3, C5, C7 and C9. They refer each to a palm leaf manuscript bundle that are stored at the UVSL in Chennai. One can find the variant **கார்விரி** in these manuscript bundles.

Ilˍ., Pēr., RK, VP, ER In these editions, the word is illegible.

As shown in Figure 3.6, the apparatus is followed by a transliteration. The transliteration is the transformation of one text into another containing only Latin letters with the use of diacritics. Usually, it is possible to transform a transliteration back into its original, which, in the case of this poem, is a text composed of Tamil characters.

The transliteration is followed by the interlinear translation into English, as depicted in Figure 3.7. The interlinear translation is a reading aid for students and helps to understand the meaning of the variants in the poems.

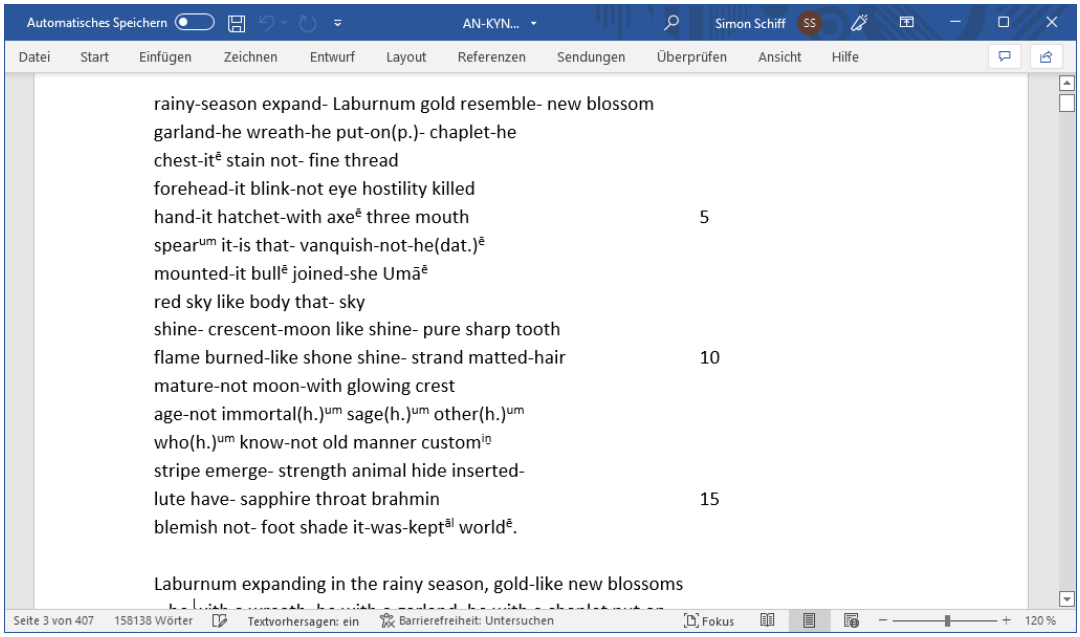


Figure 3.7.: Interlinear translation part of a critical edition written in Microsoft Word

The last part of each chapter in the critical edition is the translation into English, as depicted in Figure 3.8. The translation is not interlinear, as it should help the reader to understand the poem itself. Translating the poem, written in Tamil, into English is very difficult and not always exact. Sometimes new words that do not match the surface structure of the original need to be introduced (Wilden, 2018). These words are highlighted using square brackets.

For each abbreviation, such as C2 or C3, a description can be found at the head of the critical edition, as depicted in Figure 3.9.

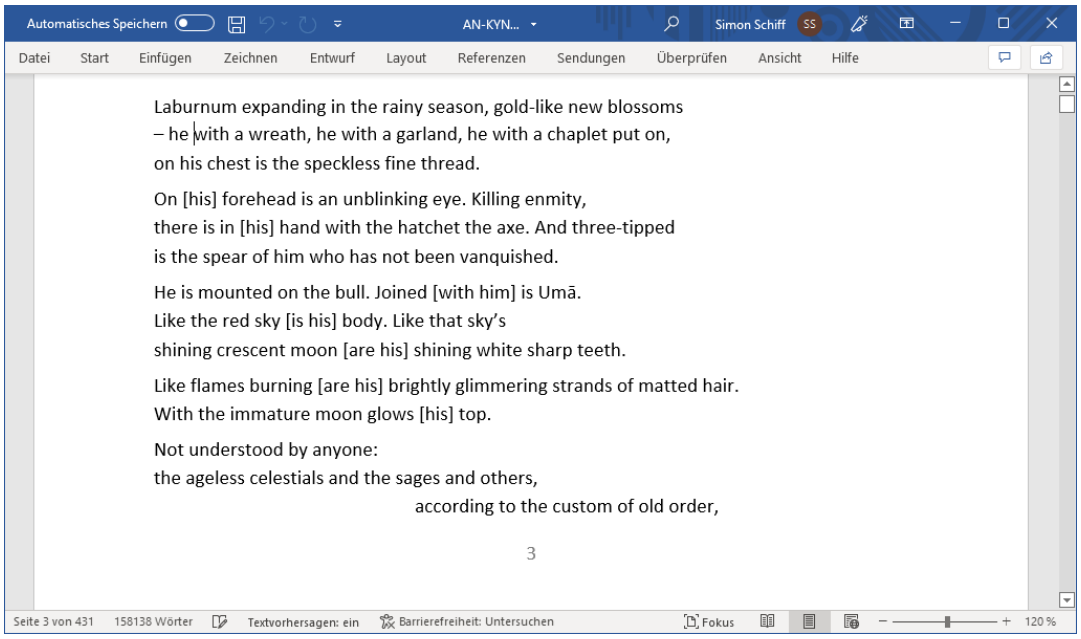


Figure 3.8.: Translation into English as part of a critical edition written in Microsoft Word

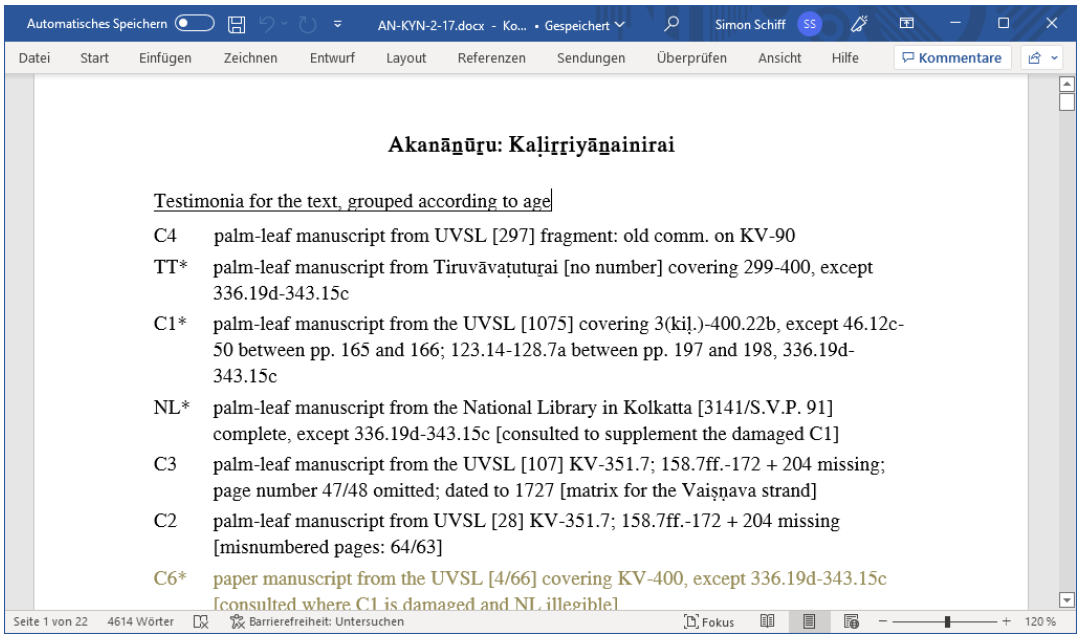


Figure 3.9.: Testimonia part of a critical edition written in Microsoft Word

3.3. Load Word Documents into a Database

Even if using a program with a WYSIWYG editor is beneficial, new problems can arise that can be very costly and time intensive to solve manually. A critical edition and a "common textbook," such as a novel may both be in Microsoft Word DOCX format. From a technical perspective, documents in DOCX format are ZIP archives containing multiple XML documents. Some of the XML documents contain the contents of the document that the author of the document can see using the WYSIWYG editor in Microsoft Word. Paragraphs, for instance, are highlighted using XML and shown to be the title of a chapter, have a specific color, be bold, etc. The schema for the XML documents is OOXML. Syntactically, there is no difference between the XML documents of a critical edition and a "common textbook", as both are encoded in XML with the same schema. That holds for most of the languages being used to highlight specific parts of texts for visualization purposes.

However, critical editions differ syntactically from "common textbooks" with respect to the textual content, as "common textbooks" are usually sequences of chapters consisting of a title and textual paragraphs. In contrast, critical editions consist of more finely-grained structures.

The critical edition written by Eva Wilden in Microsoft Word is not only a sequence of chapters. Most of the chapters in the document contain, among other points, a poem written in Tamil that stems from the reconstruction of various palm leaves and editions. Each of these chapters is syntactically identical and is a sequence of paragraphs divided by an empty new line, that each have an identical syntax across chapters:

Kiḷavi Miniature commentary added to each poem written in Tamil.

Tamil poem Poem written in Tamil. Approximately ten to 30 lines. At every 5th line, a tab separated line number is appended.

Apparatus An apparatus for the poem. It contains all variations across manuscripts and editions. Each line is written in a specific syntax across all chapters.

Transliteration The transliteration of the poem written in Tamil. Approximately ten to 30 lines. In every fifth line, a tab-separated line number is appended.

Interlinear translation An interlinear translation of the poem written in Tamil. Approximately ten to 30 lines. In every fifth line, a tab-separated line number is appended.

Translation A translation of the Tamil poem into English.

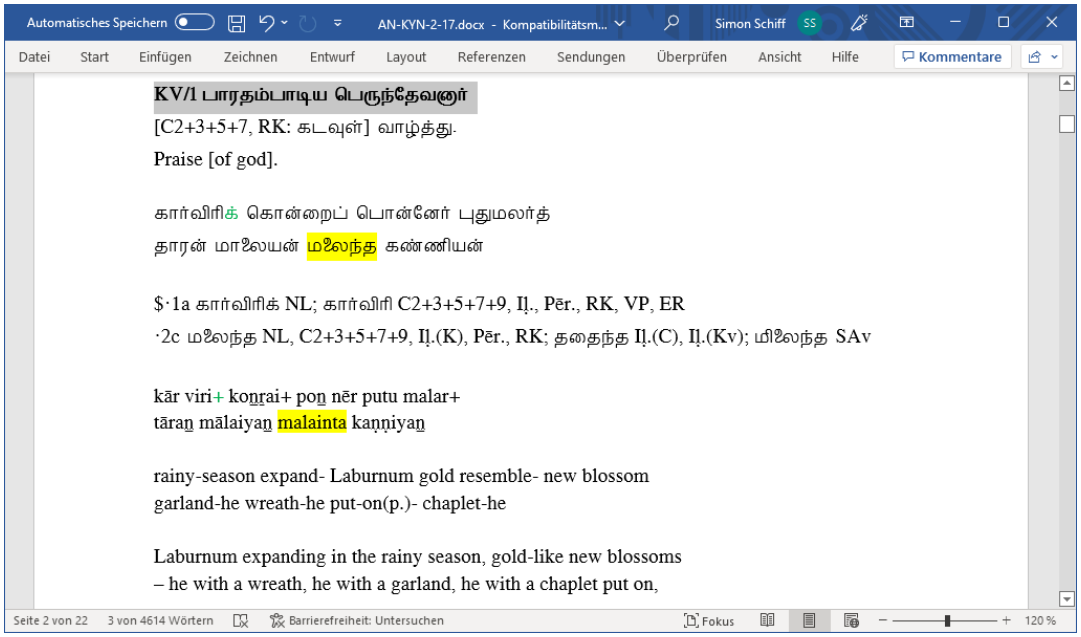


Figure 3.10.: Example of a tiny DOCX document written in Microsoft Word

If one ignores the markup language, the syntactical difference between a critical edition and a "common textbook" is a crucial advantage. In a textbook, each paragraph in a chapter has an arbitrary semantic, whereas this is not the case in a critical edition. Even if not explicitly highlighted, the paragraphs have a specific semantic given their specific position in the critical edition relative to the chapter they are located in. The advantage is that the critical edition can be processed automatically, not only given the markup language that highlights paragraphs in the text. Additionally, the syntax of the document's contents can be used to identify paragraphs having specific semantics, even if these are not highlighted with the markup language they are encoded in.

A greatly reduced critical DOCX document for demonstration purposes, opened in Microsoft Word, is depicted in Figure 3.10. One can view the document from different perspectives.

From a semantical perspective, the document is a critical edition that contains only one poem. It is titled "AN-KYN-2-17.docx" and consists of a chapter containing the poem "KV/1 பாரதம்பாடிய பெருந்தேவனார்". The poem is a sequence of Kiḷavi, a Tamil poem, an apparatus, a transliteration, an interlinear translation, and a translation. Printed or exported as a PDF, it would look identical to the document in Figure 3.10 where it is opened in the WYSIWYG editor Microsoft Word.

From a syntactical perspective, with respect to the content of the document listed in Listing 1, nothing is highlighted.

```
KV/1 பாரதம்பாடிய பரொந்தவேனார்
[C2+3+5+7, RK: கடவுள்] வாழ்த்து.
Praise [of god].

 கார்விரிக் கொன்றைப் பொன்னரே் பூதுமலர்த்
 தாரன் மாலையன் மலைந்த கண்ணியன்

$•1a கார்விரிக் NL; கார்விரி C2+3+5+7+9, Iḷ., Pēr., RK, VP, ER
•2c மலைந்த NL, C2+3+5+7+9, Iḷ.(K), Pēr., RK; ததைந்த Iḷ.(C),
↪  Iḷ.(Kv); மிலைந்த SAv

kār viri+ koṉṟai+ poṉ nēr putu malar+
tāraṉ mālaiyaṉ malainta kaṇṇiyaṉ

rainy-season expand- Laburnum gold resemble- new blossom
garland-he wreath-he put-on(p.)- chaplet-he

Laburnum expanding in the rainy season, gold-like new blossoms
- he with a wreath, he with a garland, he with a chaplet put on,
```

Listing 1: Content of a DOCX document

In Listing 1, a sequence of paragraphs, each separated by an empty new line, can be seen. Thus, one can differentiate syntactically between these paragraphs, but not semantically as each of the paragraphs' first line might be the title of a chapter. A paragraph in the critical edition is the poem written in Tamil if it is separated by an empty new line from the Kiḷavi which itself directly follows after the title of a chapter. However, the title of a chapter is not syntactically identifiable. To identify the title of a chapter, one has to analyze the XML document's part of the ZIP archive of the DOCX document. The ZIP archive has the file structure as listed in Listing 2.

```
+-- document.docx
|   +-- [Content_Types].xml
|   +-- _rels
|   +-- docProps
|   |   +-- app.xml
|   |   +-- core.xml
|   +-- word
|       +-- _rels
|       |   +-- document.xml.rels
```

```
|       +-- document.xml
|       +-- fontTable.xml
|       +-- settings.xml
|       +-- styles.xml
|       +-- theme
|       |   +-- theme1.xml
|       +-- webSettings.xml
+-- document.docx
+-- document.png
```

Listing 2: File structure of a DOCX document

The file `[Content_Types].xml` in Listing 2 contains a list of all files in the archive and their types (Dick, 2020). Among others, it contains a line, as listed in Listing 3, in which one can find the body of the DOCX document, namely `/word/document.xml`.

```
<Override PartName="/word/document.xml" ContentType="application/vnd.o
↪  penxmlformats-officedocument.wordprocessingml.document.main+xml"/>
```

Listing 3: Location of the body of the DOCX document

The content of `/word/document.xml` contains the missing information that the first line of a paragraph is the title of a chapter, as listed in Listing 4.

```
...
<w:p w14:paraId="4582A05F" w14:textId="77777777" w:rsidR="00030866"
↪  w:rsidRPr="00180C92" w:rsidRDefault="00030866" w:rsidP="00030866">
  <w:pPr>
    <w:pStyle w:val="berschrift1"/>
    <w:spacing w:line="276" w:lineRule="auto"/>
    <w:jc w:val="left"/>
    <w:rPr>
      <w:rFonts w:ascii="Gandhari Unicode" w:hAnsi="Gandhari Unicode"/>
      <w:b/>
      <w:bCs/>
    </w:rPr>
  </w:pPr>
  <w:r w:rsidRPr="00180C92">
    <w:rPr>
      <w:rFonts w:ascii="Gandhari Unicode" w:hAnsi="Gandhari Unicode"/>
      <w:b/>
      <w:bCs/>
```

```
18        </w:rPr>
19        <w:lastRenderedPageBreak/>
20        <w:t>KV</w:t>
21      </w:r>
22      <w:r w:rsidRPr="00180C92">
23        ...
24        <w:t>/1</w:t>
25      </w:r>
26      <w:r w:rsidRPr="00180C92">
27        ...
28        <w:t xml:space="preserve"> </w:t>
29      </w:r>
30      <w:r w:rsidRPr="00180C92">
31        ...
32        <w:t>பாரதம்பாடிய பரைந்தவேனார்</w:t>
33      </w:r>
34      ...
35    </w:p>
36    ...
```

Listing 4: Excerpt of `/word/document.xml`

As listed in Listing 4, `berschrift1` in line 4 refers to the German word "Überschrift"; title in English. Everything enclosed within `<w:p>` in line 2 to `</w:p>` in line 35 is emphasized as being a title. Hence, "KV/1 பாரதம்பாடிய பரைந்தவேனார்", divided in line 20, 24, 28, and 32, is identifiable as being the title of a chapter. The next paragraph in the XML is the Kiḷavi, followed by the poem in Tamil, separated by an empty new line from the Kiḷavi. Thus, all paragraphs that differ from a semantic point of view are syntactically separable if one does not ignore the XML markup of a Microsoft Word DOCX document.

A document, such as a DOCX document, in which specific parts are highlighted, can be transformed into a single intermediate document, that, in addition to the content, contains specific parts of the markup language in use that help to separate all parts of the document that are different from a semantic perspective. The specific parts are, for instance, highlighted concerning whether they are titles or which color they have. In case of OOXML, one can use an XML parser such as ElementTree, usually shipped with a Python distribution (Python Software Foundation, 2022), to transform a `document.xml` file part of a DOCX document into an intermediate one.

The document listed in Listing 4 can be transformed into an intermediate one in which all semantic parts are syntactically differentiable. Transforming the document `document.xml` part of a DOCX document can be seen as a data cleaning step where

XML tags that do not separate anything semantically are filtered out. Listing 18 in Appendix A contains a Python program that uses the ElementTree library for parsing a `document.xml` with an OOXML schema. Unnecessary XML tags are ignored, and the resulting document is written out as listed in Listing 5.

```
<!CHAPTER!><b>Akanāṉūṟu</b><b>: </b><b>Kaḷiṟṟiyāṉainirai</b>

Testimonia for the text, grouped according to age
C4  palm-leaf manuscript from UVSL [297] fragment: old comm. on KV-90
NL*  palm-leaf manuscript from the National Library in Kolkatta
↪  [3141/S.V.P. 91] complete, except 336.19d-343.15c [consulted to
↪  supplement the damaged C1]
C3  palm-leaf manuscript from the UVSL [107] KV-351.7; 158.7ff.-172 +
↪  204 missing; page number 47/48 omitted; dated to 1727 [matrix for
↪  the Vaiṣṇava strand]
C2  palm-leaf manuscript from UVSL [28] KV-351.7; 158.7ff.-172 + 204
↪  missing [misnumbered pages: 64/63]
C9  paper manuscript from the UVSL [11/73] KV-399 (complete except for
↪  7, 317, 322) [pencil corr./add. = C9c/v; based on strand 2 up to
↪  17, then on strand 1]
C7  paper manuscript from the UVSL [5/67 ] KV-120, 301-400, [10/71]
↪  121-300 (except 147, 149, 152, 230, 300) [pencil
↪  corrections/additions = C7c/v]
C5*  paper manuscript from the UVSL [7/69, 8/70, 9/72] complete in 3
↪  vol.; dated to 1903/04 [matrix for the Vulgata strand]
* misnumbered from 107 onwards by one downwards
RK  Rājakopālāryaṉ/Irākavaiyaṅkar edition (1933?)
  [= rep. of the editio princeps 1920?]
VP  Vaiyāpurip Piḷḷai edition (1940) reported when deviating
ER  Rājam edition (1957) reported when deviating
SAv  variants listed in the unpublished translation by Cupiramaṇiya
↪  Aiyar (1975)
<!CHAPTER!><b>KV</b><b>/1</b><b> </b><b>பாரதம்பாடிய பெருந்தவேனார்</b>
[C2+3+5+7, RK: கடவுள்] வாழ்த்து.
Praise [of god].

கார்விரி<span style="color:#00B050;">க்</span> கொன்றைப் பொன்னேர் பூமலர்த்
    தாரன் மாலையன் <span
↪  style="background-color:yellow;">மலைந்த</span> கண்ணியன்

$•1a கார்விரிக் NL; கார்விரி C2+3+5+7+9, Iḷ., Pēr., RK, VP, ER
•2c மலைந்த NL, C2+3+5+7+9, Iḷ.(K), Pēr., RK; ததைந்த Iḷ.(C), Iḷ.(Kv);
↪  மிலைந்த SAv

```

```
26 kār viri<span style="color:#00B050;">+</span> koṉṟai+ poṉ nēr putu
   ↪ malar+
27 tāraṉ mālaiyaṉ <span style="background-color:yellow;">malainta</span>
   ↪ kaṇṇiyaṉ
28
29 rainy-season expand- Laburnum gold resemble- new blossom
30 garland-he wreath-he put-on(p.)- chaplet-he
31
32 Laburnum expanding in the rainy season, gold-like new blossoms
33 - he with a wreath, he with a garland, he with a chaplet put on,
```

Listing 5: Transformed `document.xml` DOCX document

XML tags such as `<w:pStyle w:val="berschrift1"/>`, where `berschrift1` stands for title, are replaced with `<!CHAPTER!>` highlighting that a title and a chapter follow, as depicted in Line 17 of Listing 5.

The intermediate document, as listed in Listing 5, contains everything necessary to syntactically separate all things semantically different. A parser, such as an already implemented XML parser, is most likely unavailable for parsing the intermediate document. Only the author of the DOCX document, the humanities scholar, has control of the controlled language and the structure of her or his documents. The controlled language, specified implicitly by the author by creating the DOCX document, can be explicitly specified using Antlr4. Antlr4 is a tool used for transforming a grammar and a lexer written in Backus–Naur form (BNF), a formal language description, into the source code of a programming language, such as Java, Python, or C++. During parsing, the parser executes application-specific code that one can include in the grammar (Parr, 2013). All languages that can be formally described with a non-left-recursive context-free grammar are supported (Parr, Harwell, and Fisher, 2014). The language of intermediate documents produced by the Python program listed in Listing 18 in Appendix A is one of them.

Using Antlr4 instead of implementing a parser manually with any programming language, e.g., Python, has several advantages. Implementing a parser manually without any parser generators such as Antlr4 is non trivial and not easy to maintain if the controlled language defined by the author changes regularly. Parsing can be easily extended to structure specific sections with even finer granularity as needed. Grammar and lexer rules are readable even by non-IT-experts if rules have meaningful names. Hence, researchers can decide to produce research data matching an existing Antlr4 grammar part of an application that can export research data into other formats or visualize them on the web. Additionally not-IT-experts are possibly able to modify the grammar a little

bit as needed.

The Antlr4 lexer and grammar listed in Listing 19 and Listing 20 in Appendix A can be automatically translated into a Python program that parses the intermediate document listed in Listing 5 and returns a set of objects containing the data of the document. A document, such as that listed in Listing 5, is syntactically a sequence of characters. Characters that belong together, such as `<!CHAPTER!>`, are referred to as tokens. A Python program generated from an Antlr4 lexer combines characters in a document into a sequence of tokens. For instance, the lexer rule listed in Listing 6 is translated into a Python program that uses a deterministic finite automaton (DFA) for combining the sequence of characters $\langle <, !, C, H, A, P, T, E, R, !, > \rangle$ to the token `<!CHAPTER!>`.

```
3  CHAPTER
4    :  '<!CHAPTER!>'
5    ;
```

Listing 6: Antlr4 lexer rule for the start of a chapter

A sequence of tokens is passed to a Python program that is generated from an Antlr4 grammar. Such a grammar consists of a rule specifying the complete language and itself consists of sub-rules and tokens. Every time a rule matches the input token stream, Python code is executed returning the matched tokens. For instance, the grammar rule for a chapter of a critical edition is listed in Listing 7.

```
79  translation_chapter returns [item]
80    @init {
81  $item = TranslationData()
82    }
83    :  (CHAPTER | SUBCHAPTER) title=text
84    {
85  $item.title = $title.item
86    }
87    NEWLINE (kilavi_tamil=text
88    {
89  if $item.kilavi_tamil is None:
90    $item.kilavi_tamil = $kilavi_tamil.item
91  else:
92    $item.kilavi_tamil += $kilavi_tamil.item
93    }
94    NEWLINE)+ NEWLINE tamil_block
95    {
96  $item.tamil = $tamil_block.items
97    }
```

```
98    NEWLINE (apparatus_block
99    {
100 $item.apparatus = $apparatus_block.items
101   }
102   NEWLINE)* transcription_block
103   {
104 $item.transcription = $transcription_block.items
105   }
106   (NEWLINE+ first_translation
107   {
108 $item.first_translation = $first_translation.items
109   }
110   )? (NEWLINE+ second_translation
111   {
112 $item.second_translation = $second_translation.items
113   }
114   )? NEWLINE*
115   ;
```

Listing 7: Antlr4 grammar rule for a poem written in Tamil

If it matches a sequence of tokens or combined tokens by sub-rules, then a TranslationData object, listed in Listing 8, containing the contents of the corresponding chapter is returned, as listed in Listing 9.

```
13 class TranslationData:
14   def __init__(self):
15     self.title = None
16     self.kilavi_tamil = None
17     self.tamil = list()
18     self.apparatus = list()
19     self.transcription = list()
20     self.kilavi_english = list()
21     self.first_translation = list()
22     self.second_translation = list()
```

Listing 8: TranslationData Python object

The rule, listed in Listing 7, in line 83 at first expects the tokens `CHAPTER` or `SUBCHAPTER` followed by a text. Rule `text` matches, e.g., numbers, Latin characters, tabulators, spaces, Tamil characters, and diacritics. Hence, TranslationData.title is set to the sequence of tokens matching the text. Finally, if the rule

`translation_chapter` has the sequence of characters from the intermediate document listed in Listing 5 as input, a Python object is returned. The Python object is listed in Listing 9 serialized as JSON.

```
{
    "title": "<b>KV</b><b>/1</b><b> </b><b>பாரதம்பாடிய
    ↪ பெருந்தேவனார்</b>",
    "kilavi_tamil": "[C2+3+5+7, RK: கடவுள்] வாழ்த்து.Praise [of
    ↪ god].",
    "tamil": [
        "கார்விரி<span style=\"color:#00B050;\">க்</span> கொன்றைப்
        ↪ பொன்னேர் பூதுமலர்த்",
        "தாரன் மாலையன் <span
        ↪ style=\"background-color:yellow;\">மலைந்த</span>
        ↪ கண்ணியன்"
    ],
    "apparatus": [
        "$•1a கார்விரிக் NL; கார்விரி C2+3+5+7+9, Il̤., Pēr., RK, VP,
        ↪ ER",
        "•2c மலைந்த NL, C2+3+5+7+9, Il̤.(K), Pēr., RK; ததைந்த
        ↪ Il̤.(C), Il̤.(Kv); மிலைந்த SAv"
    ],
    "transcription": [
        "kār viri<span style=\"color:#00B050;\">+</span> koṉṟai+ poṉ
        ↪ nēr putu malar+",
        "tāraṉ mālaiyaṉ <span
        ↪ style=\"background-color:yellow;\">malainta</span>
        ↪ kaṇṇiyaṉ"
    ],
    "kilavi_english": [],
    "first_translation": [
        "rainy-season expand- Laburnum gold resemble- new blossom",
        "garland-he wreath-he put-on(p.)- chaplet-he"
    ],
    "second_translation": [
        "Laburnum expanding in the rainy season, gold-like new
        ↪ blossoms",
        "- he with a wreath, he with a garland, he with a chaplet put
        ↪ on,"
    ]
}
```

Listing 9: Serialized JSON object

One can see the contents of Listing 5 in a structured format. In this way, the complete

critical edition, created by Eva Wilden, can be transformed into a set of JSON objects each containing a chapter. The apparatus consists of a JSON array with two string elements. Each string represents a complete list of variations for a specific word and is not structured any further in the JSON. By adapting the Antlr4 grammar listed in Listing 19 in Appendix A accordingly, the apparatus could be further structured to access all witnesses.

3.4. Export Data from a Database into Various Formats

For further processing, the JSON objects, such as those listed in Listing 9, are loaded into a PostgreSQL database. The database contains the relations `document` listed in Table 3.1a, `critical` listed in Table 3.1b, and `testimonia` listed in Table 3.1c. The relation `document` contains metadata on the documents of which content is parsed, extracted, and loaded into the database. Each document has an associated `id` uniquely referring to it. The attribute `content` contains the text of the corresponding Word document without any markup. The content is used to index the document for IR, where the content is compared with a query. Relation `critical` and `testimonia` have the attribute `document_id` to refer to `id` in the relation `document`, and as well as `document` have a unique `id` for each entry. The former contains the contents of the JSON objects, such as those listed in Listing 9, and the latter the testimonia, line by line, of the critical edition, created by Eva Wilden. Storing data in a relational database management system (RDBMS) has several advantages: Data can be accessed by an API, directly via SQL, integrated and combined with other data, and thus exported into any format in which the data could be encoded. One of these formats is the XML schema specification TEI used for encoding, for example, critical editions. TEI is widely used by humanities scholars and, thus, a good choice for encoding a critical edition. Data stored in the database, as listed in Table 3.1, could be transformed into TEI by using SQL as listed in line 33 to line 368 of Listing 21 in Appendix A. While the SQL query is complex at first sight, it can be easily modified to change the resulting TEI documents, as listed in Listing 10. The document in Listing 10 is valid with respect to the XML schema of TEI.

Column	*Type*
id	text
docfile	text
docname	text
content	text
doctype	text
docformat	text

(a) Schema of the relation `document`

Column	*Type*
document_id	text
id	integer
title	text
kilavi_tamil	text
tamil	text[]
source	text[]
transcription	text[]
kilavi_english	text[]
first_translation	text[]
second_translation	text[]

(b) Schema of the relation `critical`

Column	*Type*
document_id	text
id	integer
key	text
value	text

(c) Schema of the relation `testimonia`

Table 3.1.: Schema of a PostgreSQL database for managing data of humanities scholars

```
<?xml version="1.0" standalone="yes"?>
<TEI xmlns="http://www.tei-c.org/ns/1.0">
  <teiHeader>
    <fileDesc>
      <titleStmt>
        <title>KV/1 பாரதம்பாடிய பரெுந்தவேனார்</title>
        <author>Eva Wilden</author>
        <respStmt>
          <resp>compiled by</resp>
            <name>Simon Schiff</name>
        </respStmt>
      </titleStmt>
      <publicationStmt>
        <p></p>
      </publicationStmt>
      <sourceDesc>
        <listWit>
          <witness xml:id="C4">palm-leaf manuscript from UVSL [297]
          ↪  fragment: old comm. on KV-90</witness>
          ...
          <witness xml:id="SAv">variants listed in the unpublished
          ↪  translation by Cupiramaṇiya Aiyar (1975)</witness>
        </listWit>
      </sourceDesc>
    </fileDesc>
  </teiHeader>
  <text>
    <body>
      <div n="KV/1" type="kilavi_tamil" xml:lang="ta">
        <head>kiLavi Tamil</head>
        <l n="1">[C2+3+5+7, RK: கடவுள்] வாழ்த்தை.Praise [of god].</l>
      </div>
      <div n="KV/1" type="tamil" xml:lang="ta">
        <head>Tamil Poem</head>
        <l n="1">கார்விரி&lt;span
        ↪  style="color:#00B050;"&gt;க்&lt;/span&gt; கொன்றைப்
        ↪  பொன்னரே ூதுமலர்த்</l>
        <l n="2">தாரன் மாலையன் &lt;span
        ↪  style="background-color:yellow;"&gt;மலைந்த&lt;/span&gt;
        ↪  கண்ணியன்</l>
      </div>
      <div type="apparatus">
        <head>Apparatus</head>
```

```
38          <l n="1">$•1a கார்விரிக் NL; கார்விரி C2+3+5+7+9, Iḷ., Pēr.,
            ↪  RK, VP, ER</l>
39          <l n="2">•2c மலைந்த NL, C2+3+5+7+9, Iḷ.(K), Pēr., RK;
            ↪  ததைந்த Iḷ.(C), Iḷ.(Kv); மிலைந்த SAv</l>
40        </div>
41        <div type="transliteration" xml:lang="ta">
42          <head>Transliteration</head>
43          <l n="1">kār viri&lt;span
            ↪  style="color:#00B050;"&gt;+&lt;/span&gt; koṉ̲rai+ poṉ nēr
            ↪  putu malar+</l>
44          <l n="2">tāraṉ mālaiyaṉ̲ &lt;span
            ↪  style="background-color:yellow;"&gt;malainta&lt;/span&gt;
            ↪  kaṇṇiyaṉ̲</l>
45        </div>
46        <div type="word_by_word_translation" xml:lang="en">
47          <head>Word-by-word translation into English</head>
48          <l n="1">rainy-season expand- Laburnum gold resemble- new
            ↪  blossom</l>
49          <l n="2">garland-he wreath-he put-on(p.)- chaplet-he</l></div>
50        <div type="translation" xml:lang="en">
51          <head>Translation into English</head>
52          <l n="1">Laburnum expanding in the rainy season, gold-like
            ↪  new blossoms</l>
53          <l n="2">- he with a wreath, he with a garland, he with a
            ↪  chaplet put on,</l>
54        </div>
55      </body>
56    </text>
57  </TEI>
```

Listing 10: Exported chapter of a critical edition as TEI

Research data at the database listed in Table 3.1 is sufficiently finely grained structured in accordance with the requirements of the author of the data. Hence, one can use SQL to export the research data into any desired format or access the data directly via an API. As the data is accessible in any format it can be reused, as proposed by the FAIR principles, for any purpose such as for training a model, visualization, printing, combining with other data or linking.

3.5. Comparison with Pandoc

Pandoc is open-source and can read documents of many source formats and parses and extracts their contents before writing documents into many target formats. It can transform Microsoft Word DOCX documents into XML valid with respect to the TEI schema. Instead of using our proposed ETL pipeline, depicted in Figure 3.2, one could simply use Pandoc to transform a critical edition, such as that presented in Section 3.2 and written in Microsoft Word DOCX into TEI. Pandoc can be used via a terminal with the command listed in Listing 11.

```
pandoc AN-KYN-2-17.docx -f docx -t tei -s -o critical.xml
```

Listing 11: Transformation of a Microsoft Word DOCX document into TEI with Pandoc

Pandoc transforms the document `AN-KYN-2-17.docx` of `docx` format into `critical.xml` in `tei` format. The resulting document `critical.xml` is listed in Listing 12. It is valid with respect to the TEI schema; however, it is crucially different from the TEI document, listed in Listing 5.

```
<?xml version="1.0" encoding="utf-8"?>
<TEI xmlns="http://www.tei-c.org/ns/1.0">
<teiHeader>
  <fileDesc>
    <titleStmt>
      <title><hi rendition="simple:bold">Akanāṉūṟu:
Kaḷiṟṟiyāṉainirai</hi></title>
    </titleStmt>
    <publicationStmt>
    </publicationStmt>
    <sourceDesc>
      <p>Produced by pandoc.</p>
    </sourceDesc>
  </fileDesc>
</teiHeader>
<text>
<body>
<p><hi rendition="simple:underline">Testimonia for the text, grouped
according to age</hi></p>
<p>C4 palm-leaf manuscript from UVSL [297] fragment: old comm. on
KV-90</p>
...
<p>SAv variants listed in the unpublished translation by Cupiramaṇiya
Aiyar (1975)</p>
```

```
<div type="level1" xml:id="kv1-பரதமபடய-பரநதவனர">
  <head><hi rendition="simple:bold">KV/1 பாரதம்பாடிய
       பரொந்தவேனார்</hi></head>
  <p>[C2+3+5+7, RK: கடவுள்] வாழ்த்து.</p>
  <p>Praise [of god].</p>
  <p>கார்விரிக் கொன்றைப் பொன்னரே் பூதுமலர்த்</p>
  <p>தாரன் மாலையன் மலைந்த கணண்ியன்</p>
  <p>$•1a கார்விரிக் NL; கார்விரி C2+3+5+7+9, Iḷ., Pēr., RK, VP, ER</p>
  <p>•2c மலைந்த NL, C2+3+5+7+9, Iḷ.(K), Pēr., RK; ததைந்த Iḷ.(C),
  ↪  Iḷ.(Kv);
  மிலைந்த SAv</p>
  <p>kār viri+ koṉṟai+ poṉ nēr putu malar+</p>
  <p>tāraṉ mālaiyaṉ malainta kaṇṇiyaṉ</p>
  <p>rainy-season expand- Laburnum gold resemble- new blossom</p>
  <p>garland-he wreath-he put-on(p.)- chaplet-he</p>
  <p>Laburnum expanding in the rainy season, gold-like new
  ↪  blossoms</p>
  <p>- he with a wreath, he with a garland, he with a chaplet put
  on,</p>
</div>
</body>
</text>
</TEI>
```

Listing 12: Transformed Microsoft Word DOCX document into TEI by Pandoc

The `<body>` element of the document listed in Listing 12 contains only nested `<p>` paragraph elements, including a `<div>` element that contains the chapter. There is no distinction between the Kiḷavi, the poem written in Tamil, apparatus, transliteration, interlinear translation, and the translation into English, and the order of `<p>` elements is not preserved. Attributes for the language of the text are missing as well as whether text segments are, for example, bold, have a specific color or background color, or are crossed out. Colors are very important as they highlight variations between palm leaves as well as paper manuscripts and a lot of work was done to color-code the text, something that is completely lost by using Pandoc. Instead of `<witness>` elements, Pandoc has created `<p>` elements and does not distinguish between the `id` of a witness and its description.

4. Linking and Combining Research Data

Research data is often generated across geographical and chronological borders by many different people. In some cases, such data could be combined with other data stored in different locations and created in the past, as it may have been created within the same project or belong to the same domain. For instance, weather forecast data always stems from the same domain but is often stored in different locations and formats. Transforming the data into the same format and combining it is beneficial for, e.g., training a model for predicting the weather. The more data is available through combining, the more precise the model becomes. It is, however, not only training data intended only for machines that could be usefully combined. Combining knowledge, regardless of how it is encoded, helps reduce the time required for humans to find information. If two encyclopedias are combined, readers are not required to search for information in both.

However, combining everything related to each other is not useful. Research data can become too large and humans might have to go through too much information to find what they are searching for. Nevertheless, if data is related to each other, then one could link it with each other. Links are either implicit or explicit, and various standards have emerged. An implicit link is a description of a resource in another location and is not intended to be readable by machines and an explicit one, on the other hand, is readable by machines and mostly standardized. In the World Wide Web (WWW), HTML documents are linked with each other via URLs that are readable by machines. Apart from not being technically feasible, combining all HTML documents on the web into one is not useful. Instead, links help machines and humans find related documents. If links are only implicitly available, more sophisticated methods would be required to find mentioned documents with a machine and humans would have to use an IR system.

In Section 4.1, we present word indices that need to be combined before being published and in Section 4.2 medical data that need to be combined to obtain medical data containing both static and temporal data that can be shared and reused without restrictions. In Section 4.3, we examine the linking of critical editions, word indices, commentaries, and external resources on the web for cases in which it does not make sense to combine everything. Implicitly available links are extracted and analyzed to

make them explicit so that humans can click on and machines can follow them. Links point either to a viewer for TEI documents or to external resources. In Section 4.4, we demonstrate our viewer that displays TEI documents in a human-readable format. Users can directly access the contents of the documents on the web, the viewer helps users to decide whether to download a dataset, and one can follow links, pointing to other documents on the web, by clicking on them.

4.1. Combining Research Data Created by Humanities Scholars

Humanities scholars create research data in the form of documents based on the critical edition, such as word indices. Word indices contain words used in a critical edition, a reference to their occurrence in the texts, such as poems written in Tamil, and a translation into English. Indices are transformed into TEI as described in Section 3.1 and loaded into the RDR at Universität Hamburg. As indices are very time-consuming to create, the creation is done by many researchers, resulting in multiple word indices in the form of Microsoft Word DOCX documents on the same texts. As the documents concern the same texts, potential readers would have to go through all those word indices to retrieve all information available on a specific word. Hence, word indices must be combined before being published. Doing so manually would take up to several weeks or even months of work and may be error-prone. We import the word indices to be merged into a PostgreSQL database, combine these, and finally write them back into the RDR in TEI format, in which they are linked with the indices they originate from, as depicted in Figure 4.1. In Section 4.1.1, we present word indices and in Section 4.1.2, we show how to combine them automatically, thus reducing the time required to do so manually by weeks or even several months.

4.1.1. Word Indices

Humanities scholars create, amongst other things, word indices and textbooks for students alongside a critical edition. A word index contains a list of the words used in a critical edition, including their location in the texts, derivations, and translation into English. For instance, a small excerpt of the **Akanāṉūṟu** Index, created by Eva Wilden, is listed in Figure 4.2. Its first word is “acai” and it can be found in the Dravidian etymological dictionary (DEDR) 37$^{\text{th}}$ entry (Burrow and Emeneau, 1984). It is a verb, meaning “to move” in English, it is the verbal root of the vocabulary entries following not separated by an empty new line from it, and can be found by word references, such

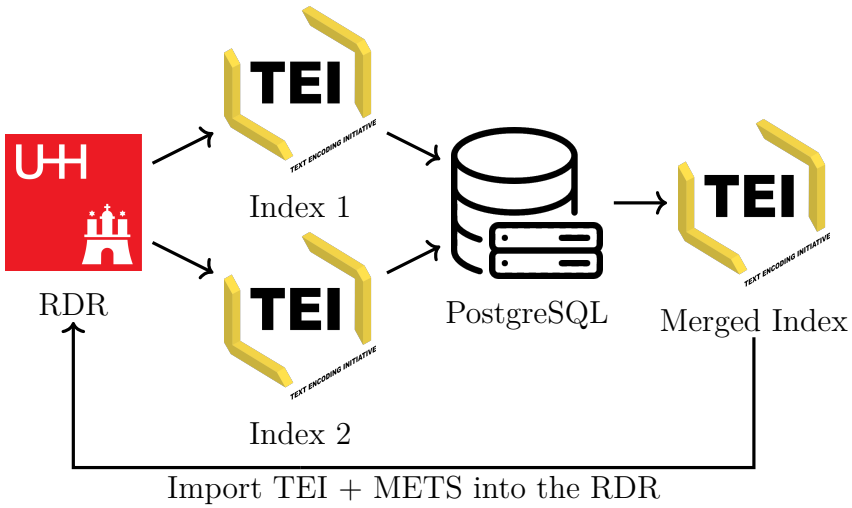
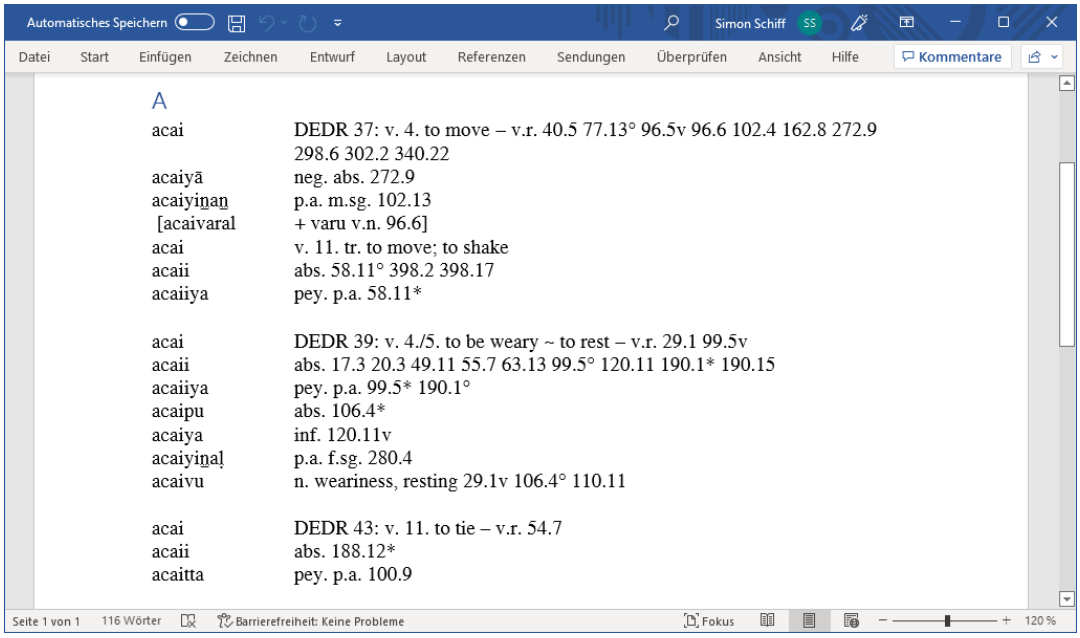


Figure 4.1.: Merge Word Indices



A

acai	DEDR 37: v. 4. to move – v.r. 40.5 77.13° 96.5v 96.6 102.4 162.8 272.9 298.6 302.2 340.22
acaiyā	neg. abs. 272.9
acaiyiṉaṉ	p.a. m.sg. 102.13
[acaivaral	+ varu v.n. 96.6]
acai	v. 11. tr. to move; to shake
acaii	abs. 58.11° 398.2 398.17
acaiiya	pey. p.a. 58.11*
acai	DEDR 39: v. 4./5. to be weary ~ to rest – v.r. 29.1 99.5v
acaii	abs. 17.3 20.3 49.11 55.7 63.13 99.5° 120.11 190.1* 190.15
acaiiya	pey. p.a. 99.5* 190.1°
acaipu	abs. 106.4*
acaiya	inf. 120.11v
acaiyiṉaḷ	p.a. f.sg. 280.4
acaivu	n. weariness, resting 29.1v 106.4° 110.11
acai	DEDR 43: v. 11. to tie – v.r. 54.7
acaii	abs. 188.12*
acaitta	pey. p.a. 100.9

Figure 4.2.: Akanāṉūṟu Index

short	long
*	primary variant Vulgate
°	primary variant strand two
“	primary variant from the old commentary
v	secondary variant
R	variant found in Rajam (not attested in any surviving manuscript)
’	additional line

Table 4.1.: Abbreviations used in the **Akanāṉūṟu** Index

as `40.5` or `96.6` in poem 40 in line five or poem 77 in line 13, respectively, in the corresponding critical edition. Poems can be found by their numbers in the critical edition AN-KYN-2-17.docx, as presented in Section 3.2. Linking the words in the index with their position in a critical edition has a crucial advantage. Readers can see the context in which the word was used centuries ago and thus have a deeper understanding of the translation. Some word references, such as `77.13°`, have a suffix, as listed in Table 4.1. Each line below the word “acai” that is not separated from it by an empty new line is a derivative of it. Hence, word index entries belong together if they are not separated from each other by an empty new line. The order of the entries is not in the Latin word order as the words are transliterations of Tamil words and therefore contain diacritical characters. The correct ordering of characters, including vowels and consonants, for words written in Tamil, is listed in Listing 13.

```
1 a ā i ī u ū e ē ai o ō au k ṅ c ñ t n p m y r l v ḻ ḷ ṟ ṉ
```

Listing 13: Tamil alphabetical order

4.1.2. Combine Word Indices

Word indices, such as those depicted in Figure 4.2, are processed exactly the same way as critical editions, as described in Section 3.1. The only difference is that a different Antlr4 grammar is used to generate a Python program that parses and extracts the contents. Hence, word indices are transformed into TEI, loaded into the RDR and linked with the word indices from which they stem. The word index, depicted in Figure 4.2, is transformed into TEI as listed in Listing 14.

```
1 <?xml version="1.0"?>
2 <TEI>
3   <teiHeader>
```

```
    <fileDesc>
      <titleStmt>
        <title>Akanānūṟu Index</title>
      </titleStmt>
      <publicationStmt>
        <p/>
      </publicationStmt>
      <sourceDesc>
        <p/>
      </sourceDesc>
    </fileDesc>
  </teiHeader>
  <text>
    <body>
      <div>
        <head>A</head>
        <superEntry>
          <entry>
            <form>
              <orth>acai</orth>
            </form>
            <def>DEDR 37: v. 4. to move  v.r. 40.5 77.13&#xB0; 96.5v
            ↪  96.6 102.4 162.8 272.9 298.6 302.2 340.22</def>
          </entry>
          <entry>
            <form>
              <orth>acai</orth>
            </form>
            <def>v. 11. tr. to move; to shake</def>
          </entry>
          ...
        </superEntry>
        ...
      </div>
    </body>
  </text>
</TEI>
```

Listing 14: Excerpt of Akanāṉūṟu index transformed into TEI

TEI provides a schema for word indices and, thus, one can use specific elements such as `<superEntry>`, `<entry>`, or `<def>` (Text Encoding Initiative, 2022a). The former is for grouping `<entry>` elements belonging together as, for instance, word entries that are not separated by an empty new line in the Akanāṉūṟu index. Each entry contains

Document	Number of words
Akanāṉūṟu Index.docx	8756
AN-Palai-Index.docx	6894
Kuṟiñcippāṭṭu Glossary.docx	1339
Tirumuruku Glossary.docx	1344

Table 4.2.: Overview of word indices created by Eva Wilden

a `<form>` element that itself contains an `<orth>` element with the actual word and a `<def>` element containing a translation and a set of references to where the word can be found in the associated critical edition. Entries are automatically sorted as listed in Listing 13.

Creating a vocabulary index is painstaking work as it is done by hand. Each word in the texts is manually copied into the Microsoft Word DOCX document in the position it belongs by its ordering and translated, and all references in the texts are written down. Copying, sorting, and creating references could be bootstrapped automatically; however, the careful translation of the words given their surrounding words in the texts and categorizing them needs to be done by humanities scholars. As texts contain thousands of words, the work is shared among many humanities scholars. Hence, more than one Microsoft Word DOCX document emerges, as listed in Table 4.2.

As many humanities scholars create critical editions, some word indices are associated with the same edition, however, they differ slightly. Readers have to consider all word indices when gathering information on a certain word in the indexes. Hence, humanities scholars combine all word indices associated with the same edition manually by comparing all words and their occurrences before publishing them. Doing so manually takes up to several weeks or months of work and might be error-prone. Editions Akanāṉūṟu Index.docx and AN-Palai-Index.docx and the editions Kuṟiñcippāṭṭu Glossary.docx and Tirumuruku Glossary.docx are each associated with the same edition. The former is depicted in Figure 4.3, in which a tiny excerpt of Akanāṉūṟu Index.docx is pictured above and AN-Palai-Index.docx below. Each excerpt of the word index contains three segments of words that are separated by an empty new line and “acai” is the first word. The word “acai” is translated either into “to move” and is DEDR’s 37$^{\text{th}}$ entry, “to be weary ~ to rest” and is DEDR’s 39$^{\text{th}}$ entry, or “to tie” and is DEDR’s 43$^{\text{rd}}$ entry. As one can see, these entries occur in both word indices; thus, the three segments of words in the word indices belong together. If the first word of a segment is equal to the first word in another segment of another document, all words in these segments need to be compared and combined if equal. Hence, entries of both word indices from Figure 4.3 are

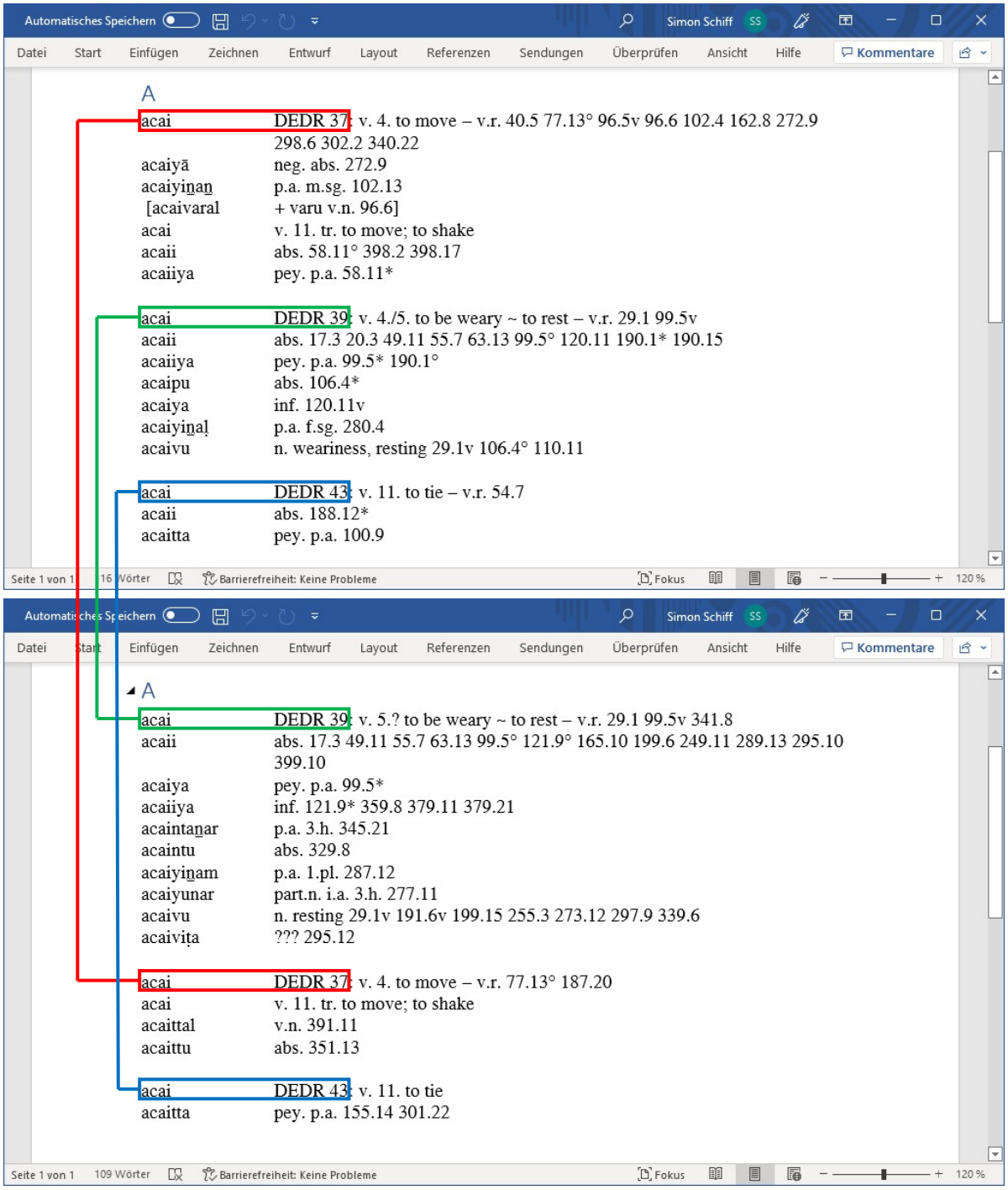


Figure 4.3.: Similar word indices that need to be combined before being published

loaded into a PostgreSQL database and joined into one relation as listed in Table 4.3, so entries that can be combined are listed above each other. Entries having the same *id* belong together in the same way as words belong together that are not separated by an empty new line in a Microsoft Word DOCX document. Attribute *doc* refers to the document from which the word stems and thus helps trace back which information stems from which document. *Header* is the chapter in which the entries were listed in the word indices to be merged.

doc	*id*	*key*	*value*	*header*
1	1	[acaivaral	+ varu v.n. 96.6]	A
1	1	**acai**	**DEDR 37: v. 4. to move v.r. 40.5 77.13° 96.5v 96.6 102.4 162.8 272.9 298.6 302.2 340.22**	A
2	1	**acai**	**DEDR 37: v. 4. to move v.r. 77.13° 187.20**	A
2	1	**acai**	**v. 11. tr. to move; to shake**	A
1	1	**acai**	**v. 11. tr. to move; to shake**	A
1	1	acaii	abs. 58.11° 398.2 398.17	A
1	1	acaiiya	pey. p.a. 58.11*	A
2	1	acaittal	v.n. 391.11	A
2	1	acaittu	abs. 351.13	A
1	1	acaiyi<u>n</u>a<u>n</u>	p.a. m.sg. 102.13	A
1	1	acaiyā	neg. abs. 272.9	A
1	2	acai	DEDR 39: v. 4./5. to be weary ~ to rest v.r. 29.1 99.5v	A
2	2	acai	DEDR 39: v. 5.? to be weary ~ to rest v.r. 29.1 99.5v 341.8	A
1	2	**acaii**	**abs. 17.3 20.3 49.11 55.7 63.13 99.5° 120.11 190.1* 190.15**	A
2	2	**acaii**	**abs. 17.3 49.11 55.7 63.13 99.5° 121.9° 165.10 199.6 249.11 289.13 295.10 399.10**	A
2	2	acaiiya	inf. 121.9* 359.8 379.11 379.21	A
1	2	acaiiya	pey. p.a. 99.5* 190.1°	A
2	2	acainta<u>n</u>ar	p.a. 3.h. 345.21	A
2	2	acaintu	abs. 329.8	A
1	2	acaipu	abs. 106.4*	A

2	2	acaiviṭa	??? 295.12	A
2	2	acaivu	n. resting 29.1v 191.6v 199.15 255.3 273.12 297.9 339.6	A
1	2	acaivu	n. weariness, resting 29.1v 106.4° 110.11	A
1	2	acaiya	inf. 120.11v	A
2	2	acaiya	pey. p.a. 99.5*	A
2	2	acaiyi̲nam	p.a. 1.pl. 287.12	A
1	2	acaiyi̲nal̤	p.a. f.sg. 280.4	A
2	2	acaiyunar	part.n. i.a. 3.h. 277.11	A
2	3	**acai**	**DEDR 43: v. 11. to tie**	A
1	3	**acai**	**DEDR 43: v. 11. to tie – v.r. 54.7**	A
1	3	acaii	abs. 188.12*	A
1	3	**acaitta**	**pey. p.a. 100.9**	A
2	3	**acaitta**	**pey. p.a. 155.14 301.22**	A

Table 4.3.: Sorted and joined word indices in a PostgreSQL database

Entries are not sorted correctly with respect to their key as listed in Listing 13; however, they are sorted in a way that sees two entries listed above each other if they are combinable. Combinable entries are highlighted in bold in Table 4.3 and separated from others by a horizontal line. More specifically, two-word entries are combinable if:

- their associated *id*, *key* and *header* are equal, and
- the *value* of one entry is the prefix of the other's *value*, except for the references.

For instance, as depicted in Figure 4.4, the word "acai" in DEDR's 43$^{\text{rd}}$ entry can be found in both documents and its entry is combined as "DEDR 43: v. 11. to tie - v.r. 54.7." One entry is highlighted in blue, the other in yellow, and in the combined entry, combined parts are highlighted in green = blue + yellow. Although it would seem possible to do so, not all entries were merged automatically, such as "acai" which is DEDR's 39$^{\text{th}}$ entry. However, the values of both entries are slightly different and combining is left to a humanities scholar. Scholars do not have to search for such entries as most of them are easily identified by computing the edit distance of the corresponding values. If the edit distance is low, a humanities scholar is notified and entries are combined by hand, if possible. New combining rules can be easily implemented, such as combining "…v. 4./5.…" and "…v. 5.?…" into "…v. 4./5.…" to automatize the combination process further.

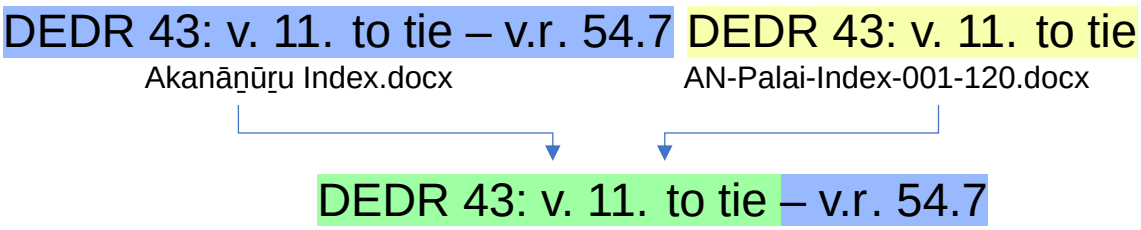


Figure 4.4.: Combining equal word entries

prefix	*prefix_cut*	*suffix*	*suffix_cut*
beauty	6		0
	0	*]	1
⋮	⋮	⋮	⋮

Table 4.4.: Tokenization rules for word index references

References in the entries have the format `[0-9]+.[0-9]+` and sometimes have a non-numeric prefix or suffix. They only have a prefix if a single space character was forgotten and have a suffix if an abbreviation follows, as listed in Table 4.1. In rare cases, after the abbreviation, characters follow that should be separated from the reference, including its abbreviation. For instance, the reference "beauty101.42*]" has the prefix "beauty", the abbreviation "*" and is followed by the character "]". The correct tokenization for "beauty101.42*]" would be "beauty", "101.42*", and "]". Reference "101.42*" is not split into "101.42" and "*", as "101.42" and "101.42*" are not combined. A humanities scholar must decide whether "101.42" and "101.42*" can be combined as either "101.42" or "101.42*". Tokenization needs to be done with care and a relation is created automatically that contains rules for the process, as listed in Table 4.4. The first row denotes that each reference with "beauty" as a *prefix* is split after "prefix_cut", namely six characters, the length of beauty. The second row denotes that every reference's suffix, with "*]" as *suffix*, is split after *suffix_cut*, namely one character. If *prefix* or *suffix* is empty, then *prefix_cut* or *suffix_cut* is 0 respectively. A humanities scholar can add or modify rules as needed that are applied to combining any pair of documents. Rules themselves are automatically added if a reference has a prefix or suffix not already added to the set of rules. The value for *prefix_cut* or *suffix_cut* is then set by a humanities scholar.

The final result of combining **Akanāṉūṟu** Index.docx and AN-Palai-Index.docx from Figure 4.3 is listed in Table 4.5. Entries that stem from **Akanāṉūṟu** Index.docx are high-

lighted in blue, those from AN-Palai-Index.docx are highlighted in yellow, and entries that stem from both and were combined are highlighted in green. All entries that belong together are separated from others by an empty new line and form a block of entries. Each block is sorted by its first entry, here acai, and within a block, all entries are sorted as well, as listed in Listing 13 in Tamil alphabetical order.

key	*value*
acai	DEDR 37: v. 4. to move v.r. 40.5 77.13° 96.5v 96.6 102.4 162.8 187.20 272.9 298.6 302.2 340.22
acai	v. 11. tr. to move; to shake
acaii	abs. 58.11° 398.2 398.17
acaiiya	pey. p.a. 58.11*
acaittal	v.n. 391.11
acaittu	abs. 351.13
acaiyā	neg. abs. 272.9
acaiyi̲na̲n	p.a. m.sg. 102.13
[acaivaral	+ varu v.n. 96.6]

acai	DEDR 39: v. 5.? to be weary to rest v.r. 29.1 99.5v 341.8
acai	DEDR 39: v. 4./5. to be weary to rest v.r. 29.1 99.5v
acaii	abs. 17.3 20.3 49.11 55.7 63.13 99.5° 120.11 121.9° 165.10 190.1* 190.15 199.6 249.11 289.13 295.10 399.10
acaiiya	pey. p.a. 99.5* 190.1°
acaiiya	inf. 121.9* 359.8 379.11 379.21
acaintana̲r	p.a. 3.h. 345.21
acaintu	abs. 329.8
acaipu	abs. 106.4*
acaiya	pey. p.a. 99.5*
acaiya	inf. 120.11v
acaiyi̲na̲m	p.a. 1.pl. 287.12
acaiyi̲na̲ḷ	p.a. f.sg. 280.4
acaiyunar	part.n. i.a. 3.h. 277.11
acaiviṭa	??? 295.12
acaivu	n. weariness, resting 29.1v 106.4° 110.11
acaivu	n. resting 29.1v 191.6v 199.15 255.3 273.12 297.9 339.6

acai	DEDR 43: v. 11. to tie – v.r. 54.7
acaii	abs. 188.12*

acaitta	pey. p.a. 100.9 155.14 301.22

Table 4.5.: Combined word indices Akanāṉūṟu Index.docx and AN-Palai-Index.docx

Combined word entries, such as those listed in Table 4.5, are written out as TEI documents, then stored at the RDR at Universität Hamburg, and linked with the word indices from which they were combined. The latter is important as improvements might require accessing the original word indices. Word indices Akanāṉūṟu Index.docx and AN-PalaiIndex.docx listed in Figure 4.3 are only tiny excerpts of larger word indices, as listed in Table 4.6. Excerpts contain only 17 and 16 entries, while the larger ones they

Document	Words	Total	Combined
1.A: Akanāṉūṟu Index.docx (Excerpt)	17	28	5
1.B: AN-Palai-Index.docx (Excerpt)	16		
2.A: Akanāṉūṟu Index.docx	8756	12557	3077
2.B: AN-Palai-Index.docx	6878		

Table 4.6.: Overview of combined word indices created by Eva Wilden at where "words" is the total number of word indices to be combined, "total" the number of words of the combined word index, and "combined" the total number of word pairs that were combined into one

are an excerpt of contain 8.756 and 6.894 word entries, respectively. The excerpts were combined into a total of 28 word entries, of which five were combined. The larger ones were combined into a total of 12.557 entries, of which 3.077 were combined. A human would need weeks or even months of work to achieve the same and results are possibly error-prone as thousands of word entries need to be compared, sorted, and combined. Additionally, 26.248 references need to be combined manually by first computing the intersection and then sorting them to ascend numerically. Instead of going through two-word indices, one can go through the combined one once to find all the required information. As we show in the following, combining research data is not only beneficial in the domain of humanities scholars and data to be combined do not necessarily need to have the same schema.

4.2. Combining Medical Data

Electronic health records (EHRs) contain relational and temporal patient data. More specifically, EHRs contain relations between patients, medications, procedures, diag-

noses, and streams of measurements over time. Analysing data from EHRs can help improving the overall health of patients and reduce healthcare costs (Bar-Dayan et al., 2013). The data can be used to generate prediction models (Khalilia et al., 2015; Marlin et al., 2012), for example to detect heart failures (Wu, Roy, and Stewart, 2010), and even unstructured treatment notes can be systemically analysed (Perera et al., 2013). Unfortunately, machine learning techniques require collections of data which are not easily accessible. Further, exploitation of existing health-related data is very restricted due to important privacy concerns (Dernoncourt et al., 2016). Therefore, in most cases the use of real datasets is not feasible as they are, e.g., only accessible inside a clinical organisation, if even, and in addition, scaling experiments can hardly be conducted on real data. Nonetheless, a few anonymized EHR databases for research purposes exist, such as MIMIC III (A. E. Johnson et al., 2016) and eICU (Goldberger et al., 2000). Those databases are accessible to researchers, and the patients are fully anonymized, even though Emam et. al were able to re-identify patients (Emam et al., 2011), which is forbidden by the data use agreement. Researchers can test new algorithms with those anonymized datasets, reproduce test results from others, and compare algorithms. Additionally, having a limited set of data leads to known problems in machine learning, such as

1. always reusing the same dataset for machine learning can lead to overfitting,
2. missing data to test for corner cases can lead to poor prediction models, and
3. there might not be enough data to train the prediction model well (Marlin et al., 2012).

A solution for obtaining data without such limitations is to synthesize realistic, but not real, patients. Hence, one can also make the data publicly available. Synthea (Walonoski et al., 2018) is a tool that synthesizes EHRs and models the whole lifespan of patients including relations such as drug usage, allergies, and observations of low frequency e.g. the body weight. A common EHR does not only consist of such relations, the largest part of an EHR consists of time series with measurements over time within the frequency of milliseconds. About 98% of the data in the MIMIC III database consists of measurements with high frequencies, having a very huge impact on the size of an EHR, which Synthea does not synthesize. Very often, these kinds of measurements are electrocardiographies (ECGs). FECGSYN (Behar et al., 2014) is a toolbox to generate ECGs based on several parameters. However, those parameters only have an impact on the ECG curves, but specific diseases or conditions are not specifiable. Thus, the generated ECG data does not match to the relational data of patients. Further, one

might be interested in influences of multiple diseases on ECGs. Hence, we propose to enrich synthesized relational data with time series. We match generated relational data by Synthea with ECGs from MIMIC III. For the match from the two sources, we use six features, which suffices for our scenario, as the use of more features results in a more precise match, but decreases the number of matched patients. Additionally, we can use FECGSYN, not matched time series, and time series with an offset to prevent overfitting. With our approach, we generate patients on demand, where time series are consistent with the patient's medical history.

Related approaches to synthesize relational medical data without time series exist and synthesized medical data for predications are used. Murray et al. describe a way how to extract features from real databases for the creation of simulated patient datasets (Murray, Ryan, and Reisinger, 2011). Cunningham and Ainsworth present a tool to simulate realistic enough patient datasets for the development and testing of medical systems, but they note that simulated datasets are not realistic in depth nor have a pedagogical value (Cunningham and Ainsworth, 2015). Johnson et al. demonstrate the usefulness of synthesized data to evaluate and compare algorithms in machine learning (M. L. Johnson et al., 2008). Watkins et al. use semi-synthetic data for disease surveillance with a hidden Markov model (Watkins et al., 2009). All these approaches have in common that either relational or time series data is generated, but lack of a combination of both. With our solution, we present an approach how to obtain relational patient data with appropriate time series data.

We begin by presenting EHR data and dictionaries we use, followed by how we solve our data integration problem by computing similarities between patients to enrich relational patient data with time series data. Finally, we evaluate our implementation and conclude how we could use the enriched patient data.

Data Dictionaries

A patient can have several diseases diagnosed by a doctor and based on diagnoses, procedures are performed. Diagnoses and procedures are typically described in natural language. Databases containing such descriptions without any classification have many disadvantages. Thousands of different diagnoses and procedures exist and there are several ways to describe each of them. Searching or comparing patients according to their diseases becomes nontrivial. Thus, a vocabulary for the classification of diagnoses and procedures is very useful; and indeed, several proposals for such a vocabulary exist. One of these is the international statistical classification of diseases and related health problems (ICD) maintained by the World Health Organization. ICD is currently the most widely used statistical classification system in the world. Several revisions and

modifications of ICD exist. ICD-9-CM is a clinical modification of the ninth revision created by the US National Center for Health Statistics which is used in MIMIC III. Synthea uses the Systematized Nomenclature of Medicine SNOMED-CT vocabulary, in which CT stands for clinical term.

Other vocabularies can be used to classify drugs. RxNorm provides normalized names and concept codes (RxCUI) for clinical drugs and drug delivery services. The norm only includes drugs that are approved in the US. Another norm is the national drug code (NDC). An NDC consists of three segments: The first segment contains the labeler, the second one the product, and the last one the package size.

To migrate from one vocabulary to another, mappings can be used. Some vocabularies, however, are more fine-grained than others. Therefore, an exact one-to-one mapping between different vocabularies is not always possible. To map ICD-9-CM codes to SNOMED-CT, a mapping from the national library of medicine exists, mapping 95% ICD-9-CM diagnostic codes to SNOMED-CT, which is divided into one-to-one (2/3) and one-to-many (1/3) mappings.

MIMIC III

Only a few anonymized databases containing EHRs, such as MIMIC III which contains medical data from the Beth Israel Deaconess Medical Center in Boston, are available (A. Johnson, Pollard, and Mark, 2016). The data consists of information on different intensive care units (ICUs). Furthermore, the database contains tests, orders, billings, demographics, notes, and reports. MIMIC III is the only freely accessible critical care database of its kind (A. E. Johnson et al., 2016), and there is a reason for such rarity.

The MIMIC III database is not a simple one-to-one copy of an existing clinical database. Instead, the database is integrated from three sources:

1. archives from critical care information systems,
2. hospital EHR databases and
3. the Social Security Administration Death Master File.

Procedures, diagnostics, and medications are standardized to dictionaries such as LOINC, ICD-9-CM, or RxNorm, including giving the relation between the codes and their definition in what is the result of a labor-intensive process (Abhyankar, Demner-Fushman, and McDonald, 2012). Patients are automatically de-identified in accordance with the Health Insurance Portability and Accountability Act (Dernoncourt et al., 2016). The development of the database is ongoing and furthered, for example, during datathons

organized by MIT Critical Data. The structure of the data is hardly modified but may, at this point, no longer represent a clinic's real database. The whole database is available in a repository after filling out a data use agreement.

Synthesized Relational Patient Data using Synthea

Synthea is a tool used to synthesize EHRs on demand. In Synthea, the ten most frequent reasons for primary care encounters and the ten chronic conditions with the highest morbidity in the United States are modeled (Walonoski et al., 2018). Only publicly available information, such as health statistics, is used by Synthea to synthesize data. Hence, the data produced by Synthea is free of legal, privacy, security, and intellectual property restrictions.

Synthea is divided into modules. Each module contains a transition graph used to model a disease or a treatment. The development and updating of modules heavily depend on the knowledge of experts from a medical domain who are usually unable to write a program in Java. Hence, Synthea provides a web interface for medical experts to develop modules. These modules are used during the data generation process to model the whole lifespan of a patient. Thus, a patient can have several diseases and treatments during his lifetime. In fact, some transition graphs will not terminate until the death of a patient. Synthea uses standardized dictionaries, e.g., procedure and diagnostic codes from SNOMED-CT and medication codes from RxNorm. Further, the output formats Fast Healthcare Interoperability Resources (FHIR), CSV, Consolidated Clinical Document Architecture (C-CDA), and human-readable TXT are supported. However, synthesized data by Synthea lacks of historical data.

Data Integration

Datasets can have different units, structures, schemes, and semantics and the databases containing them can be different due to their implementation and query language. Thus, querying different datasets from various sources is a challenging task, even in the same context such as clinical databases. These differences are caused, e.g., by several different medical devices, ICUs, billing systems, units, and terminologies. One solution is data integration which describes the creation of a unified global view.

Formally, two different databases σ and σ' exist. The goal is to create a third database τ containing a global view. Data can be physically integrated into τ or queries over τ are automatically rewritten and sent to σ and σ'. Therefore, τ makes accessing and querying data easier as τ hides the underlying data distribution and diversity (Smith et al., 1981).

4.2.1. Enriching Synthesized Patient Data

To improve the care of patients through the analysis of medical data, we need a sufficient number of EHRs containing time series with measurements corresponding to the relational data. EHRs synthesized by Synthea are free of legal restrictions but lack time series. MIMIC III cannot be used as freely as data from Synthea but contains time series data. Therefore, we combine synthesized relational data with time series from MIMIC III. To enrich synthesized patients, we search for similar patients from MIMIC III, and only keep the synthesized patients and time series data of MIMIC III patients. Our approach allows for obtaining relational patient data of variable size containing time series, in which patient data roughly fits the time series data. Having the MIMIC III database and the ability to generate EHRs using Synthea, we perform the following steps:

1. extract common features from MIMIC III and Synthea,
2. compute patient similarity based on the features and
3. enrich relational patient data.

Features

We extract the following six patient features from MIMIC III and Synthea, which are not always explicitly given in the data but are useful for comparing patients:

1. procedures and diagnostics (disease),
2. gender,
3. age,
4. whether the patient is dead,
5. ethnicity, and
6. medications.

In MIMIC III, we obtain the patient's age for each hospital stay by calculating the difference between the patient's date of birth and the year of the hospital stay. In Synthea, we obtain the patient's age by calculating the difference between the patient's date of birth and the time the patient was synthesized. Ethnicities are mapped manually as Synthea patients can have 30 and MIMIC III patients can have 41 different ethnicities.

```
MIMIC_META (PATIENT_ID, MIMIC_GENDER, MIMIC_ETHNIC,
MIMIC_DEAD, MIMIC_AGE)
MIMIC (PATIENT_ID, SNOMED_CT, RX_NORM)
SYNTHEA_META (PATIENT_ID, SYNTHEA_GENDER, SYNTHEA_ETHNIC,
SYNTHEA_DEAD, SYNTHEA_AGE)
SYNTHEA (PATIENT_ID, SNOMED_CT, RX_NORM)
MATCH_1 (MIMIC_ID, SYNTHEA_ID, MIMIC_GENDER, SYNTHEA_GENDER,
MIMIC_ETHNIC, SYNTHEA_ETHNIC, MIMIC_DEAD, SYNTHEA_DEAD,
MIMIC_AGE, SYNTHEA_AGE, MIMIC_RX_NORM, SYNTHEA_RX_NORM,
ARRAY_COMMON_SNOMED_CT)
```

Figure 4.5.: Schema of database *match*

Synthea patient procedures and diagnostics are classified using SNOMED-CT and medications using RxNorm. MIMIC III, however, uses ICD-9-CM and NDC dictionaries for classification. As we are not able to manually map 2.032 different procedures, 6.984 diagnostics, and 4.204 medications from MIMIC III to Synthea, we use mappings from the national library of medicine to map ICD-9-CM to SNOMED-CT and NDC to RxNorm.

Using data integration, we create a new database match with the schema from Figure 4.5 to access extracted features. The view `MIMIC_META` and `SYNTHEA_META` contain patients' gender, ethnicity, age, and information on whether the patient is dead. Additionally, `MIMIC` and `SYNTHEA` contain every `SNOMED-CT` and `RxNorm` code associated with a patient.

Similarity

To obtain `MATCH_1`, as depicted in Figure 4.5, we join `MIMIC_META` with `MIMIC` and `SYNTHEA_META` with `SYNTHEA` using `PATIENT_ID` before joining both results through `SNOMED_CT`. We iteratively improve similarity by filtering out pairs that are not similar, except for the age, for which a difference of five years suffices. Every intermediate result is stored in a relation ranging from `MATCH_1`, in which a pair of patients are only similar by `SNOMED_CT`, to `MATCH_6`, in which a pair of patients are similar by all six features. We unify all `MATCH_n` relations and only keep the best match for each pair. Additionally, we count the common intersection of `SNOMED-CT` codes. We rank patients

based on their similarity using a score:

$$score = \frac{\#\text{Common SNOMED-CT Codes}}{\#\text{SNOMED-CT of Synthea Patient}} \cdot \frac{\#\text{Common SNOMED-CT Codes}}{\#\text{SNOMED-CT of MIMIC III Patient}} \cdot \frac{\#\text{similar features}}{6} \tag{4.1}$$

which can be used to balance the number of matched patients and the number of features they have in common, as patients with a high similarity are rare. Thus, the score ensures that the patients have a similar medical history. A pair of patients having a high score are more likely similar than patients with a low score as a higher score shows that patients have more features and SNOMED-CT codes in common. The score ranges from zero (as a pair of patients can have zero similar features) and one as patients cannot have more than six similar features.

Enrichment

We keep pairs of patients who have a score above a given threshold. As we match the medical history of patients, the linked time series approximately fits the enriched synthesized patient. Almost every MIMIC III patient has time series data and a similar synthetic patient, in which time series approximately fit the synthetic patient. Finally, we keep only synthetic patients' data and time series data. Additionally, we can randomly enrich synthetic patients with time series synthesized by FECGSYN (Behar et al., 2014; Andreotti et al., 2016) obtained from MIMIC III, which are not used or matched but slightly modified time series to prevent overfitting.

4.2.2. Evaluation of the Synthesization of Data

We expect different properties from our result of enriched synthesized relational patient data with time series data:

1. the result should contain a large proportion of time series and synthesized relational patient data to have a variety of patients in our result,
2. each pair of patients should be matched by using as many features as possible to obtain synthesized patients with time series matching their medical history, and
3. the processing should only take a reasonable amount of time to enrich a sufficient number of synthetic patients.

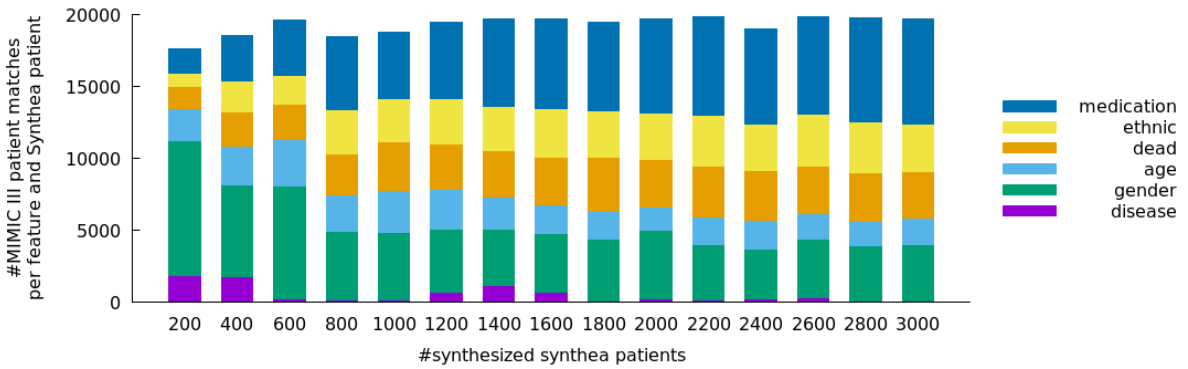


Figure 4.6.: Number of MIMIC III patient matches per feature and Synthea patient

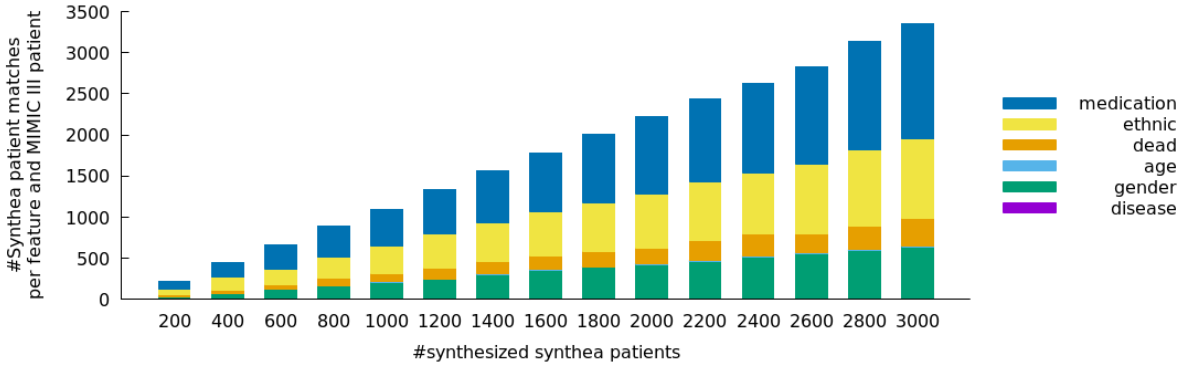


Figure 4.7.: Number of Synthea patient matches per feature and MIMIC III patient

Matching

Unfortunately, we are not able to directly use all MIMIC III patients for the matching. Roughly 7.000 MIMIC III patients do not have any corresponding medication, procedure, or diagnostic codes. In our experience, Synthea only synthesizes about 300 different SNOMED-CTs as they are based on the top ten primary care encounters and chronic conditions with the highest morbidity. Therefore, we can only match about 19.000 of the 46.520 MIMIC III patients. For these 19.000 patients, we find a match for nearly all synthesized patients. Further, we can use the remaining 27.000 patients' time series data to prevent overfitting. Therefore, we can match all-time series data to synthesized relational patient data and also find a proper match for each synthesized patient.

Figure 4.6 shows the number of distinct matched MIMIC III patients per feature for 200 to 3.000 synthesized patients. For each feature, the match also has to hold for the

less restrictive features as we compute the similarity between pairs of patients iteratively. Therefore, pairs of patients who are similar by age are also similar with respect to gender and disease. For example, with 2.000 synthesized patients, we find about 3.300 MIMIC III patients who match a synthesized patient by life/death status, age, gender, and disease, as described in Section Section 4.2.1. On top of this, we can match about 6.500 MIMIC III patients who are similar to at least one synthesized patient in all six features.

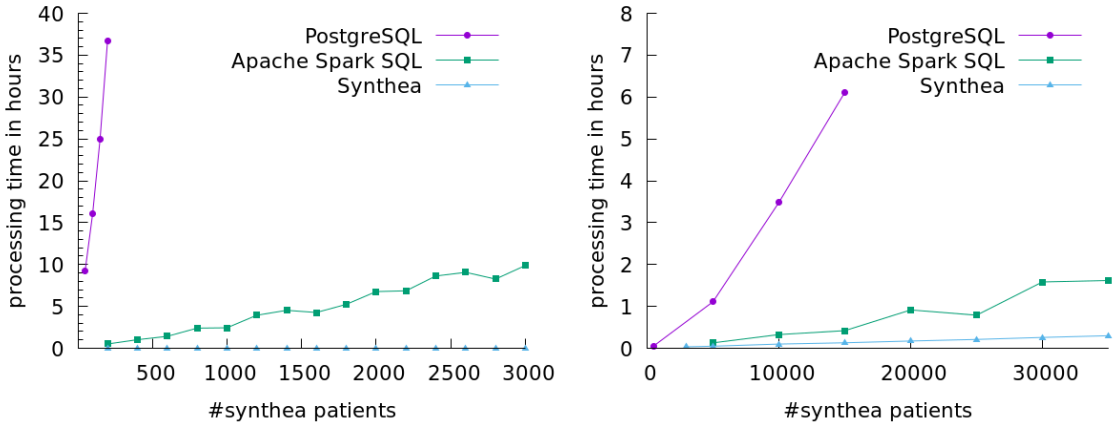
The number of matchable MIMIC III patients is fixed at 19.000, whereas the number of synthetic patients varies. With an increasing number of synthesized patients, the number of MIMIC III patients with a high similarity score (Equation (4.1)) also increases. If we synthesize 200 patients, then around 1.600 MIMIC III patients have a similar synthetic patient in all six features. By synthesizing 3.000 patients that number increases to 7.200 MIMIC III patients.

Figure 4.7 depicts the number of distinctly matched synthetic patients per feature. We obtain more synthetic patients than specified, as some synthetic patients are dead. For example, if we use Synthea to synthesize 3.000 patients, then Synthea synthesizes 3.352 patients (3.000 alive and 352 dead). We find a MIMIC III patient who is similar in all six features for 1.400 of the 3.000 synthetic patients. Each synthetic patient is similar to at least one MIMIC III patient by disease and gender. Regardless of how many patients we synthesize, we can match each synthesized patient with at least one similar MIMIC III patient, and the number of features per synthetic patient is similarly distributed, as the number of MIMIC III patients is fixed. Finally, we select the best MIMIC III patients to enrich the synthesized patients, using our similarity score, Equation (4.1).

We achieve our goals 1. and 2., namely that we can match each patient and thereby use all possible MIMIC III patients and have high similarity scores for these matches. However, we still need to show that we can find the matches in a reasonable timeframe.

Processing

The processing time strongly depends on the number of patients we enrich. Figure 4.8a shows the time needed to enrich up to 3.000 patients without any optimizations. We split the time it takes to generate the patients using Synthea and to enrich them using PostgreSQL or Apache Spark SQL, as described in Section 4.2.1. It takes roughly four minutes to synthesize up to 3.000 patients with Synthea. Enriching even a few patients using PostgreSQL already takes hours and, therefore, is unpractical. With Apache Spark SQL, a distributed execution engine installed on a 4-node cluster, in which each node has 8 Intel Xeon E5-2620 v3 processor cores and 21 GB-Ram, we speed up the enriching process. Using Spark SQL drastically reduces the time required to enrich patients but, unfortunately, still takes around 10 hours for 3.000 patients, which is also not reasonable.



(a) Processing times without optimizations (b) Processing times with optimizations

Figure 4.8.: Processing time to enrich n synthesized patients

Thus, we deploy some optimizations: 1. reducing mapping files from around 8.5 million rows to 131, as most of the contained codes are not in use by MIMIC III or Synthea, 2. preprocessing `MIMIC_META` and `MIMIC`, and 3. computing six different relations containing each pair of patients who are similar by one feature and then using these to iteratively improve the similarity, which avoids large cross-products.. In Figure 4.8b, the time to synthesize and enrich up to 35.000 patients using our optimization ideas are depicted. The processing times are significantly shorter. Using PostgreSQL, we can synthesize and enrich up to 5.000 patients in a reasonable amount of time. Synthea needs around 18 minutes to synthesize 35.000 patients and we are able to enrich them in under two hours using Spark SQL. Using our optimizations and a distributed execution engine such as Spark SQL allows us to efficiently enrich synthesized relational patient data with time series data on demand.

Combining requires data to be related with each other and should only be performed when it makes sense. In some cases it does not make sense such as combining a publication with all of its citations. Instead, one could link related data with each other which we show in the following accompanied by an example from the humanities scholars.

4.3. Linking of Research Data

Not everything that is related to each other could or should be combined. Instead, implicitly available links could be extracted from documents to be analyzed to make

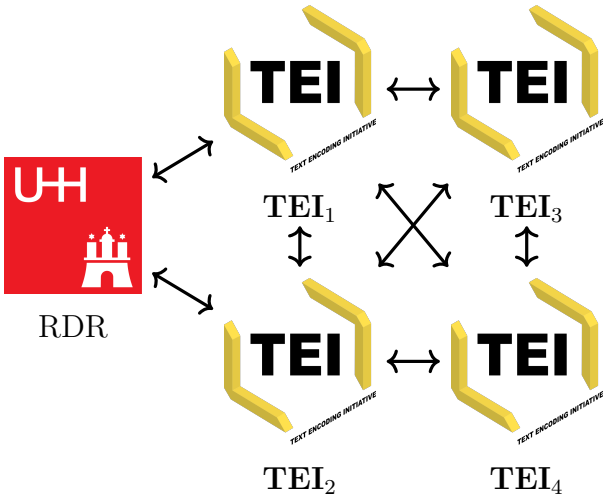


Figure 4.9.: Creation of explicit links between TEI documents

them explicitly available so that machines and humans can follow them. As depicted in Figure 4.9, we load already stored TEI documents at the RDR, make implicit links explicitly available, and then write them back.

4.3.1. Commentary Created by Eva Wilden

As presented in Section 3.1 in Figure 3.3, editions concerning the same core text were created at various times in the past. Some editions depend on only the texts, others on texts and previously created editions, and some editions only on previously created editions. Authors of poems have not only written the texts but additionally commented on them. The commentary and the core texts are of interest to humanities scholars. Hence, editions created in the past contain commentaries next to the core texts, as depicted in Figure 4.10, in which the commentary for the poem "AN 1" is listed. The transcription and transliteration by Eva Wilden are listed at the top of Figure 4.10. An excerpt of a picture of the edition is shown below. Starting by counting from one, each odd line is written in Tamil and each even line is the respective transliteration. Numbers on the right-hand side of the commentary, in this case one to six, are not the line numbers of the commentary itself. These numbers refer to the poem in which one or more words were commented on, as depicted in Figure 4.11. Words stemming from comment are highlighted in bold, while the commentary itself is not. The middle document in Figure 4.11 contains the commentary for the Tamil word "**ததனந்த**" that

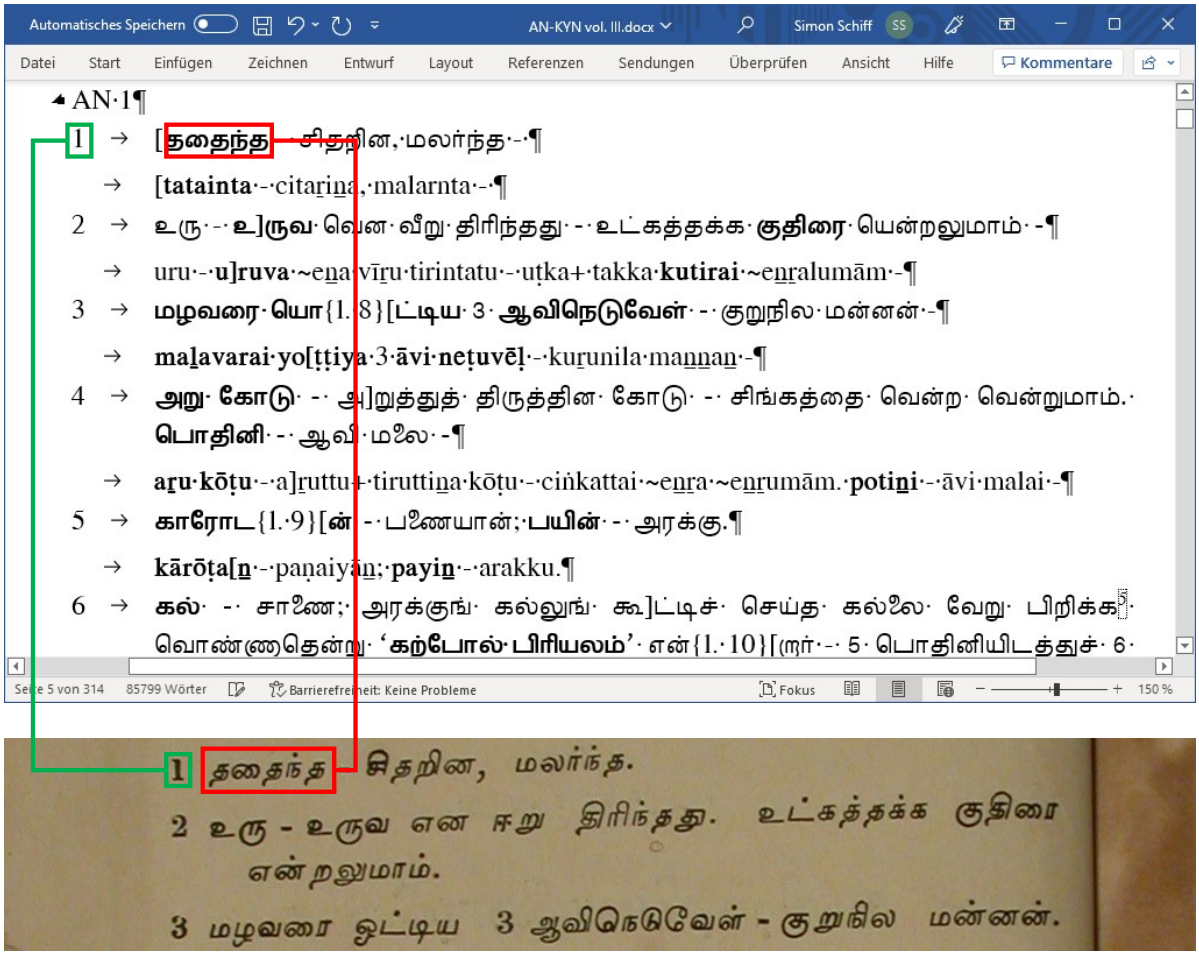


Figure 4.10.: Commentary of an edition

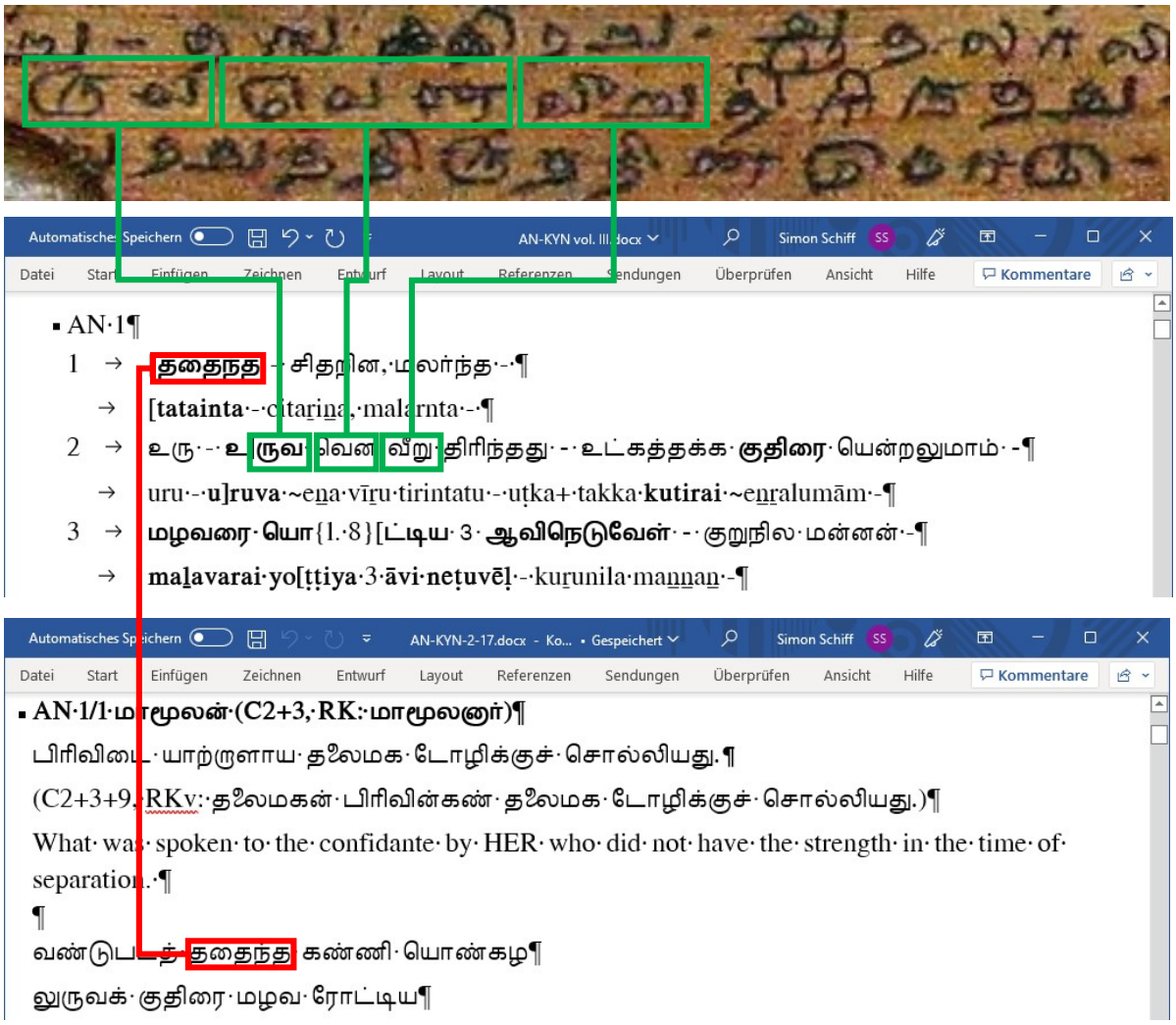


Figure 4.11.: Implicit links between a palm leaf, a commentary, and a poem

id	*chapter*	*poem*	*comment*	*pl*	*references*
1	AN KV	**[தாரன், மாலையன், மலைந்த கண்ணியன் - எ]**	ன்றதூ, தார் விசடேமாக விடூ[வதூ] ; மாலை …	2	[{C4, p. 53}]
⋮	⋮	⋮	⋮	⋮	⋮
19	AN 1	**[ததனைந்த**	சிதறின, மலர்ந்த	1	
20	AN 1	**உ]ரூவ**	வனெ வீறூ திரிந்ததூ - உட்கத்தக்க	2	
21	AN 1	**கூதிரை**	யனென்றலூமாம்	2	
22	AN 1	**மழவரை யி·ா**{l. 8}[ட்டிய ஆவிநடொவளேள்	கூறூநில மன்னன்	3	[{l. 8}]
⋮	⋮	⋮	⋮	⋮	⋮

Table 4.7.: Commentary loaded into a PostgreSQL database

has the comment "சிதறின மலர்ந்த". Poem "AN 1" is part of the lower document in Figure 4.11 and contains the word "**ததனைந்த**". However, the commentary "சிதறின மலர்ந்த" cannot be found as the lower document only contains the contents of the poems. Text within curly brackets in the commentary is not visible on the palm leaves, as parts may have broken off or been eaten by tape worms. References such as "{l. 8}" are the line number of the palm leaf the texts in the edition stem from. The palm leaf is depicted at the top of Figure 4.11. One can see that the Tamil word "உ]ரூவ" is part of the transcribed commentary created by Eva Wilden. However, "உ" of "உ]ரூவ" is not visible on the palm leaf. The missing part of "உ]ரூவ" can be found in the edition, depicted at the bottom of Figure 4.10 in line two. Hence, the palm leaf is the primary source of information and the editions are used for reconstructing missing parts, such as "உ".

The commentary, as well as the critical editions and the word indices, is transformed into TEI, as described in Section 3.3. As an intermediate step, the commentary from Figure 4.10 is loaded into a PostgreSQL database, as listed in Table 4.7. Words that stem from comments and are part of a poem are in *poem*, the associated comment in *comment*, and *pl* is the actual line in which the commented words can be found in the

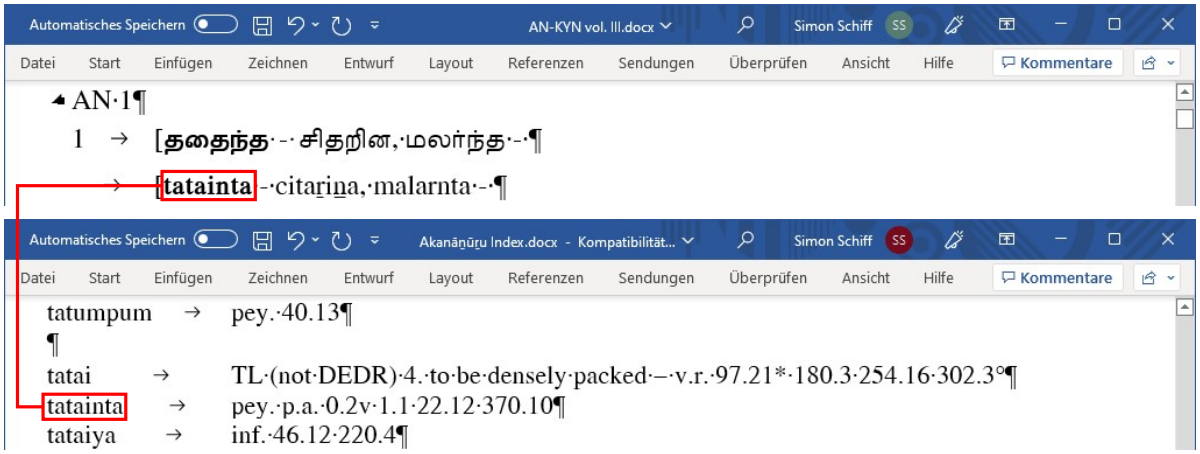


Figure 4.12.: Word index implicitly linked with a commentary

poem. References are either in form of "{C4, p. 53}" or "{l. 8}". The former points to palm leaf 53, part of bundle C4, and the latter to line eight of a palm leaf. If the references are sorted in the order in which commented words can be found in the poems, then for each line reference, the preceding palm leaf reference refers to the palm leaf which contains the referenced line. For instance, "{l. 8}" can be found at "{C4, p. 53}", as listed in Table 4.7.

Word indices are linked implicitly with the commentary, as depicted in Figure 4.12. The word "tatainta", shown in the word index at the bottom of Figure 4.12, is found to reference "1.1" in poem "AN 1" in line 1, as can be seen at the top of Figure 4.12.

4.3.2. Linking of TEI documents

The commentary, critical edition, palm leaves, word indices, and editions created in the past are implicitly linked. Potential readers must follow the links manually by either flipping through published books or through a PDF to find implicitly linked paragraphs or pictures. The latter is only stored at the RDR and access is restricted. For instance, a reader may be interested in a picture of a palm leaf that contains the comment depicted in the middle document in line one of Figure 4.11. A reference "{l. 8}" can be found in line five within the same chapter and denotes that the comment can be found in line eight of a palm leaf. However, the palm leaf itself is not referenced in line five to save space in the document. In the preceding chapter, the reference "{C4, p. 53}" to the palm leaf can be found, as shown on top of Figure 4.13. "C4" is a reference to a palm leaf bundle and "p. 53" to palm leaf number 53, part of bundle "C4." Reference "C4" is further

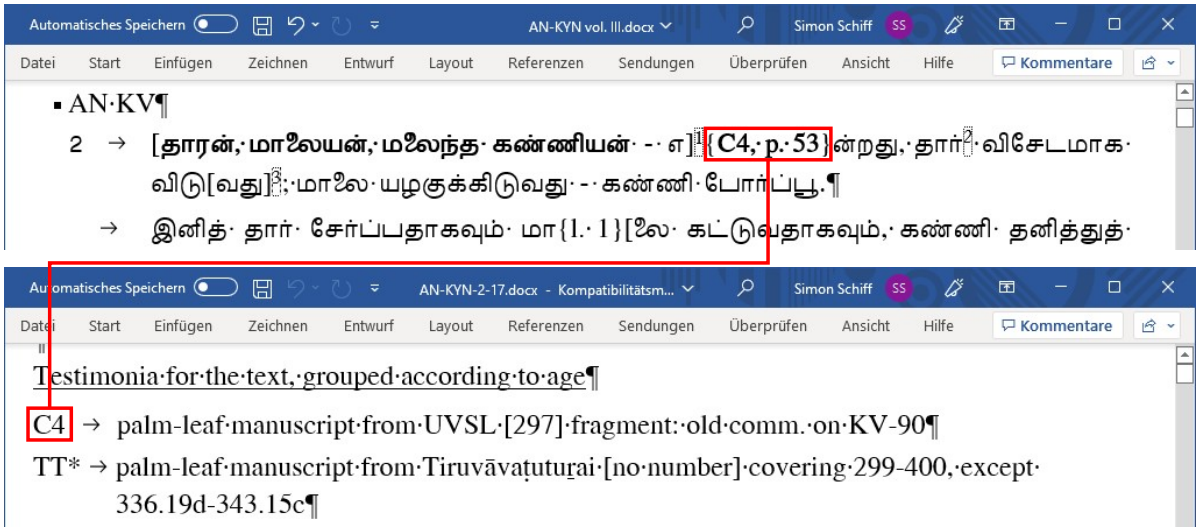


Figure 4.13.: Reference to a palm leaf in the commentary

defined in the testimonia within the same document, as depicted in the lower document of Figure 4.13 and defined as "palm-leaf manuscript from UVSL [297] fragment: old comm. on KV-90." Using the search function provided by the RDR at the Universität Hamburg returns a repository of pictures of palm leaves. Among others, one picture of which an excerpt containing the comment is depicted at the top of Figure 4.11, is named "p.-054.jpg." Hence, it is laborious to follow implicitly available links, and it would be easier if one could click on references embedded in a PDF, for instance, to directly access linked documents and pictures. To make implicitly available links explicit, references must be extracted and analyzed in detail.

All references embedded inside the commentary of a poem, such as "{C4, p. 53}" or "{l. 8}", are extracted during the transformation of the commentary into TEI and loaded into a database, as listed in Table 4.7. The references in each row, sorted by order of the commentary in the poems, are part of a sequence containing either references pointing to the page of a palm leaf within a bundle or a line number. A line number is preceded by its reference to a palm leaf in the sequence. Hence, a new relation is created from Table 4.7, as listed in Table 4.8. All rows are identified via a unique primary key *id*, unambiguously associated with the comments in Table 4.7, and with the image of the palm leaf on which the comment can be found via the name of the image in *image*.

Poems that are part of a critical edition, such as those presented in Section 3.2, stem from palm leaves and editions, as well as the commentary. In contrast to the commentary, the poems do not contain embedded references pointing to a palm leaf

id	*chapter*	*pl*	*bundle*	*page*	*line*	*image*
1	AN KV	2	C4	53	8	p.-053.jpg
⋮	⋮	⋮	⋮	⋮	⋮	⋮
19	AN 1	1	C4	53	8	p.-053.jpg
20	AN 1	2	C4	53	8	p.-053.jpg
21	AN 1	2	C4	53	8	p.-053.jpg
22	AN 1	3	C4	53	8	p.-053.jpg
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Table 4.8.: Mapping between comments and pictures of palm leaves

bundle stored at the RDR. Instead, another document contains a list of references for each poem, as depicted in Figure 4.14 and is a work in progress. Chapter "C1" in Figure 4.14 contains all references to poems for the palm leaf bundle "C1" which is defined as "palm-leaf manuscript from the UVSL [1075] covering 3(kiḷ.)-400.22b, except 46.12c-50 between pp. 165 and 166; 123.14-128.7a between pp. 197 and 198, 336.19d-343.15c" in the testimonia. All references are split by a ";" semicolon character. The first reference "missing: AN 2" denotes that, for the poem "AN 2", the palm leaf is missing and "p. 142, l. 2-7 (half broken): AN 4" that the contents of the poem "AN 4" can be found from line two to line seven on palm leave 142. Odd numbers of poems are left out to save space in the document and can be recomputed by using the last line of the preceding reference and the first line of the reference following if the number of lines of a palm leaf is known. The third reference is "p. 143, l. 5 (half broken) - p. 144, l. 1: AN 6: p. 144, l. 8 - p. 145, l. 4: AN 8" if they are split by a semicolon ";". However, a colon ":" was used instead by accident. References to the pictures of palm leaves, such as "p. 143", do not match the names of the corresponding picture. The picture of "p. 143" is named "p.-143.jpg" and replacing white space characters " " with "-" does not suffice as other pictures contain, for instance, leading zeros that cannot be found in the references. Hence, the document listed in Figure 4.14 is processed semi-automatically by using, among other things, regular expressions for replacing characters, such as ":" with ";" if necessary. The resulting chapter "C1" contains only references that can be specified with regular expressions, as listed in Table 4.9. The first one in Table 4.9 specifies references that link to lines of text ranging from one line of a specific palm leaf to another line on another palm leaf. As each reference matches one of the regular expressions, page and line numbers can be extracted easily, as listed in Table 4.10, in which extracted references are highlighted in bold. References not highlighted in bold are derived from the number of lines a palm leaf has and its preceding and following

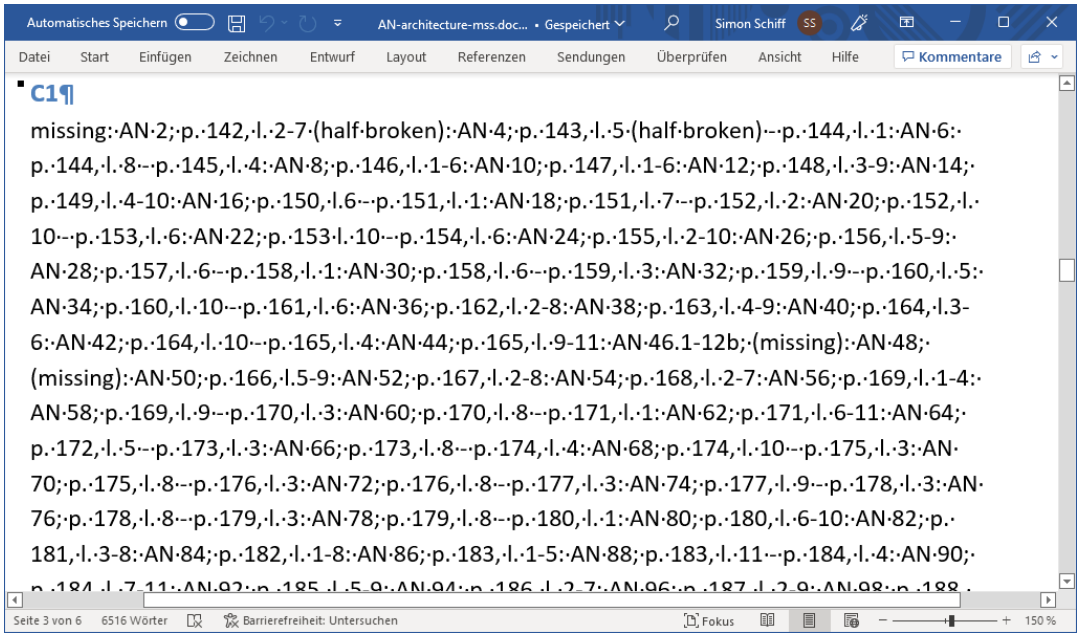


Figure 4.14.: Links between poems and pictures of palm-leaves

```
^p. [0-9]+, l. [0-9]+ - p. [0-9]+, l. [0-9]+: AN [0-9]+$
^p. [0-9]+, l. [0-9]+ - \?: AN [0-9]+$
^p. [0-9]+, l. [0-9]+ - [0-9]+: AN [0-9]+$
^missing: AN [0-9]+$
```

Table 4.9.: Specifying references to palm-leave repository “C1”, using regular expressions

caption	*ps*	*pe*	*ls*	*le*	*poem*	*image*
⋮	⋮	⋮	⋮	⋮	⋮	⋮
p. 204, l. 3 - 9: AN 142	**204**	**204**	**3**	**9**	**142**	**p.-142.jpg**
p. 204, l. 10 - p. 205, l. 1: AN 143	204	205	10	1	143	p.-143.jpg
p. 205, l. 2 - 7: AN 144	**205**	**205**	**2**	**7**	**144**	**p.-144.jpg**
p. 205, l. 8 - 11: AN 145	205	205	8	11	145	p.-205.jpg
p. 206, l. 1 - 5: AN 146	**206**	**206**	**1**	**5**	**146**	**p.-146.jpg**
p. 206, l. 6 - 8: AN 147	206	206	6	8	147	p.-147.jpg
p. 206, l. 9 - p. 207, l. 2: AN 148	**206**	**207**	**9**	**2**	**148**	**p.-148.jpg**
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Table 4.10.: Mapping between poems and palm-leaves

entry. For instance, "**p. 204, l. 3 - 9: AN 142**" refers to the picture "p.-142.jpg" of a palm leaf that has ten lines. Hence, the next entry "p. 204, l. 10 - p. 205, l. 1: AN 143" begins in line ten of "p.-142.jpg" instead of line one of "p.-143.jpg".

Word indices created by Eva Wilden contain links to external sources, such as Dravidian etymological dictionary (DEDR) (Burrow and Emeneau, 1984) and are depicted in Figure 4.2, in which the DEDR's 37th, 39th, and 43rd entries are implicitly linked. The DEDR dictionary is digitized and available to everyone online. It is not possible to create links pointing to an entry in the dictionary by the number of the entry. However, it is possible to create links for queries, such that the website returns results given the query, as listed in Listing 15, in which the query is appended to a URL.

```
https://dsal.uchicago.edu/cgi-bin/app/burrow_query.py?qs=
```

Listing 15: Using the query API of DEDR

Entries are linked with DEDR by appending the word of a word index entry to the URL, as shown in Listing 15.

In summary, implicitly available links in the commentary, critical edition, palm leaves, word indices, and editions created in the past can be extracted and analyzed semi-automatically. Analyzed links, such as those listed in Table 4.8 or in Table 4.10, are then transformed into explicit links, depending on the context. In the next section, a viewer for created TEI documents is presented that is used to visualize TEI documents stored at the RDR. In the case of the viewer, explicitly created links point to other viewers for TEI documents, pictures of palm leaves, or external websites.

4.4. Viewer for TEI Documents

All TEI documents that stem from the transformation of Microsoft Word DOCX are stored at the RDR hosted by the RDM at Universität Hamburg. TEI documents that belong together, as, for example, they stem from the same Microsoft Word DOCX document, are zipped, have associated metadata encoded in METS format, an automatic created and assigned DOI, and have metadata created automatically when the data is stored at the RDR. The latter is standardized by DataCite.org (DataCite Metadata Working Group, 2021) and used to list research data at the RDR so that it can be indexed by a search engine and identified by humans. Metadata encoded in METS is used for listing archives contents and deciding what should be executed if a reader clicks on a TEI document that is part of an archive.

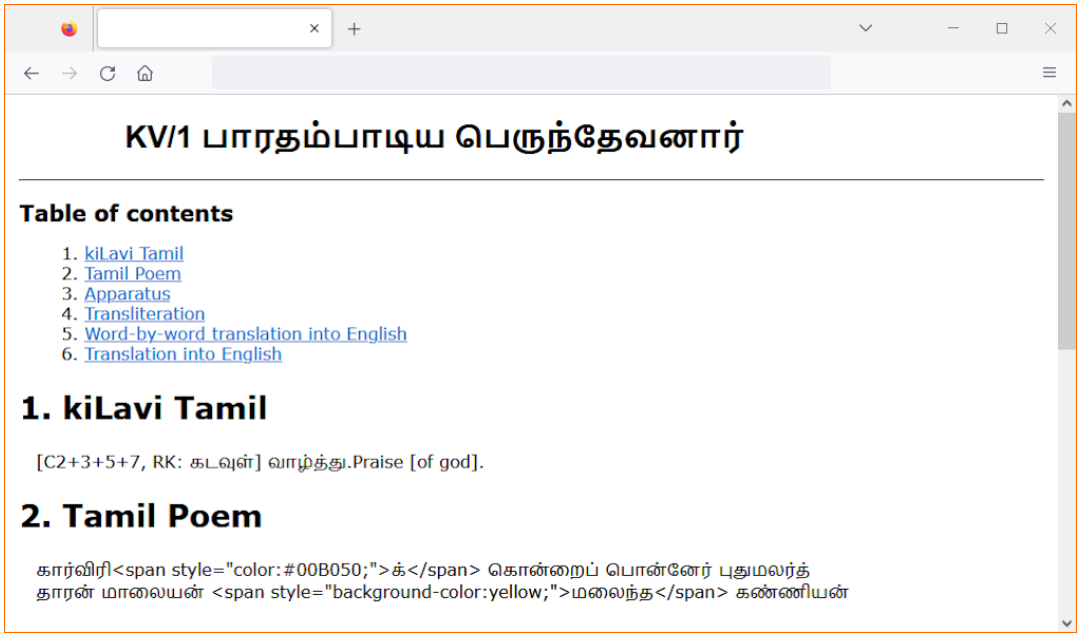


Figure 4.15.: Transformed TEI document into HTML + CSS, using XSLT stylesheets provided by a repository of the TEI initiative, displayed in a web browser

A potential reader must download the archive, extract it and then go through the TEI documents to decide whether they are relevant or not. Depending on the size of the archive in terms of bytes or number of documents and on whether documents are analyzed manually, this can be laborious and time-intensive. Instead, the behaviour section of a METS file associated with the ZIP archive is used to decide whether an executable is suitable for viewing documents stored inside the archive. The TEI guidelines link to a repository of XSLT stylesheets for transforming TEI documents (Rahtz, 2022) that are themselves XML documents with a specific schema into various output formats, such as HTML + CSS. HTML is a markup language and can be combined with CSS to display texts on the web and is thus more appropriate for viewing TEI documents than formatting them. The result of transforming the TEI document, listed in Listing 10, into HTML + CSS is depicted in Figure 4.15. The document in Figure 4.15 is displayed in a web browser. While the content of the TEI document, listed in Listing 10, is easy to read if transformed into HTML + CSS, some content, such as everything between the `<witness>` tags, is entirely lost during the transformation, as the full specification of the TEI guidelines is not yet supported. Another set of XSLT stylesheets, provided by Jeroen Hellingman in Hellingman (2022), also transforms the TEI document, listed in

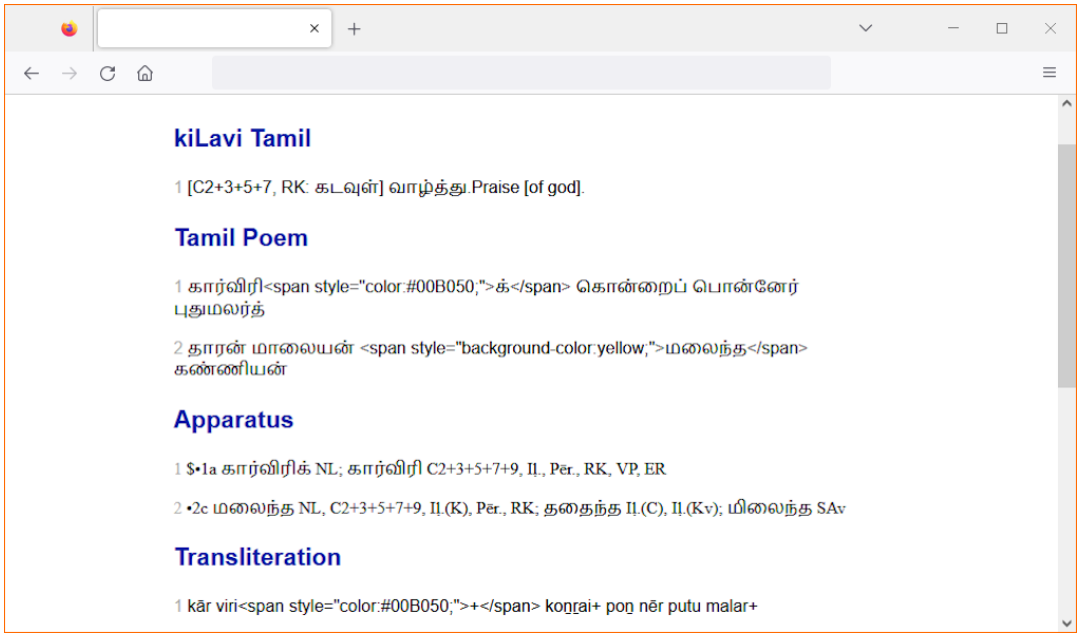


Figure 4.16.: Transformed TEI document into HTML + CSS, using XSLT stylesheets provided by Jeroen Hellingman at GitHub (Hellingman, 2022)

Listing 10, into HTML + CSS. The result is depicted in Figure 4.16; again, everything between the `<witness>` tags is ignored completely.

Available XSLT stylesheets do not completely support the full TEI standard as they are created for specific purposes and mostly by single individuals that do not have the time and resources to further extend the transformation. Even if one implements a transformation from TEI to HTML + CSS that fully supports the TEI standard, the visualization of documents might not satisfy everyone. If all TEI files stored at the RDR are viewed by transforming them with existing XSLT stylesheets into HTML + CSS, then authors of TEI documents potentially misuse XML tags so that all content of the documents is visible in the viewer. For instance, the `<witness>` section is exchanged with generic `<div>` elements that are supported by the XSLT sheets, even if `<witness>` may be more suitable.

Instead of modifying existing XSLT stylesheets, which is cumbersome in our view, we use JavaScript (JS) for different kinds of TEI documents. As TEI documents stored at the RDR are unmodifiable and visualized according to the author's specifications, a viewer does not need to be realized by implementing a transformation from TEI to HTML + CSS that supports the full TEI standard. If a revision of a TEI document is

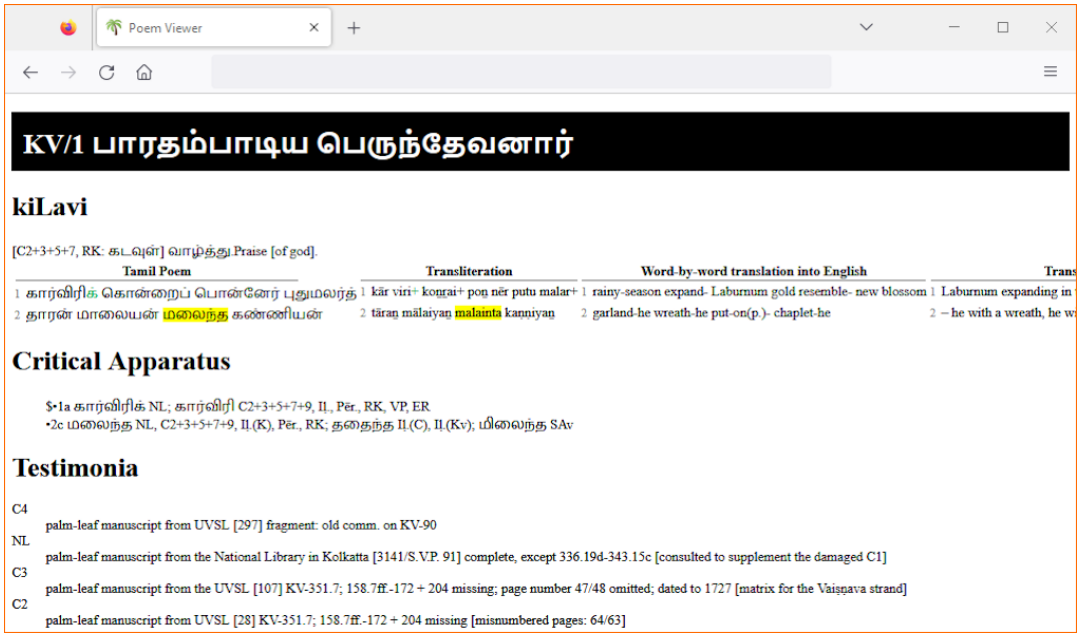


Figure 4.17.: Viewer for TEI documents of type critical

uploaded at the RDR with changes in the structure of the document, the transformation implemented in JS can be modified more easily than if it is implemented in XSLT. Our viewer, as depicted in Figure 4.18, can be used anywhere locally which has a web browser installed supporting JS and DOMParser which is shipped with most modern web browsers. Everything that is between the `<witness>` tags is visible in our viewer under the chapter "Testimonia." Colors are displayed in Figure 4.17 correctly instead of with HTML tags, as can be seen in Figure 4.15 and in Figure 4.16. Embedded HTML tags for colorizing text between TEI tags are not standardized by the TEI guidelines, as well as the colors themselves. While existing implementations ignore the HTML markup, the colors are specifically chosen by the author of the contents and need to be preserved during the transformation from Word DOCX into TEI and from TEI to HTML. HTML tags between TEI tags are not allowed according to the TEI guidelines and thus escaped by replacing "<" with "<", ">" with "&gt", and "&" with "&".

As presented in Section 4.3.2, transcribed texts and pictures of palm leaves and editions are linked to each other. Critical editions containing transcribed texts that are printed as a book do not contain any of the pictures, as the librarians who administrate the library at which the pictures of palm leaves and editions were taken, do not permit their publication. However, some readers may have permission to view the pictures and

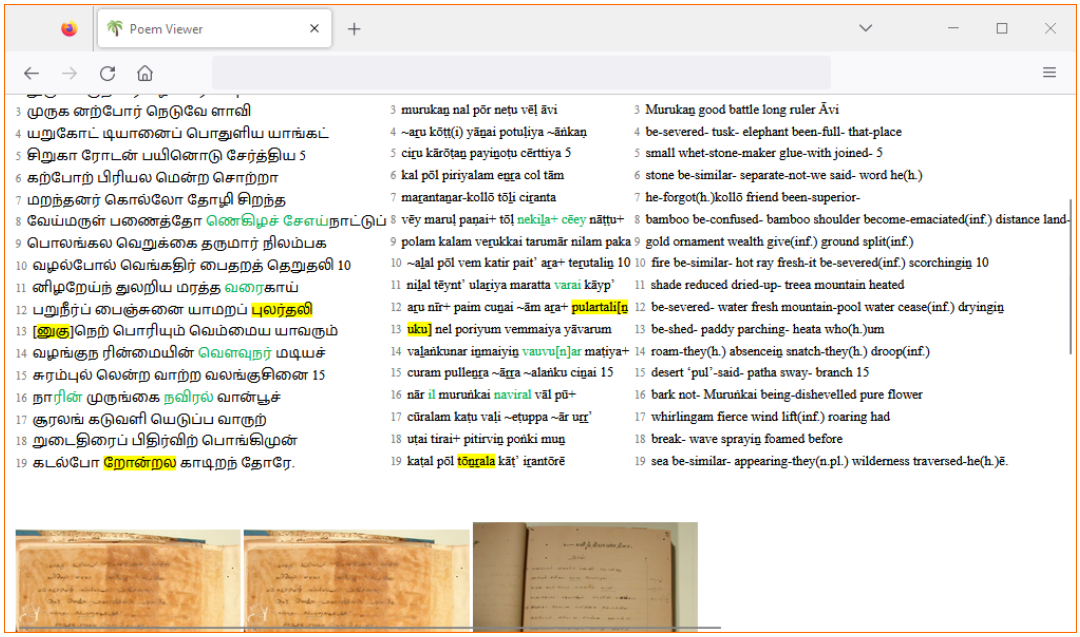


Figure 4.18.: Palm leaves linked with transcribed texts and displayed in our viewer

are interested in these (e.g., to verify a transcription). Our viewer is extended with a login to authenticate readers and check whether they are allowed to view the pictures. If permitted, the pictures are displayed in our viewer, as depicted in Figure 4.18. A snippet of the TEI document displayed by our viewer in Figure 4.18 is listed in Listing 16. Each `<figure>` element contains a `<head>` and `<graphic>` element. The caption of each image is located within the `<head>` element and the path to the image itself is within the `<graphic>` element. Images itself are not part of the ZIP archive containing viewed TEI files, as access is restricted. Instead, images are stored at a linked repository at which the access is restricted.

```
<?xml version="1.0" standalone="yes"?>
<TEI xmlns="http://www.tei-c.org/ns/1.0">
  ...
  <text>
    <body>
      ...
      <figure>
        <head>AN 1</head>
        <graphic url="AN-C7-001.jpg"></graphic>
      </figure>
      <figure>
```

```
12          <head>AN 1</head>
13          <graphic url="AN-C7-001.jpg"></graphic>
14        </figure>
15        <figure>
16          <head>AN 1</head>
17          <graphic url="005-AN-C5-1.jpg">
18          </graphic>
19        </figure>
20      </body>
21    </text>
22  </TEI>
```

Listing 16: Snippet of a TEI document that is linked with images

Word indices contain words in all transliterations that are part of a critical edition, their translation into English, and are linked with them via references, as presented in Section 4.1.1. When given a word in a printed word index, one has to comb through all transliterations in a printed critical edition to find the corresponding occurrences. Even if searching for the occurrences of the words in the transliterations manually is time-consuming, it is beneficial as texts surrounding a word help understand its semantics. Hence, word indices displayed by our viewer in a web browser, as depicted in Figure 4.19, are linked with transliterations by their reference, such as "54.7" referring to poem 54 at line 7, and with DEDR entries such as DEDR's 43rd entry. Links pointing to specific lines of transliterations are embedded in TEI documents with `<ref>` elements, as listed in Listing 17 and transformed into `<a>` anchor elements for displaying them in our viewer, as depicted in Figure Figure 4.19.

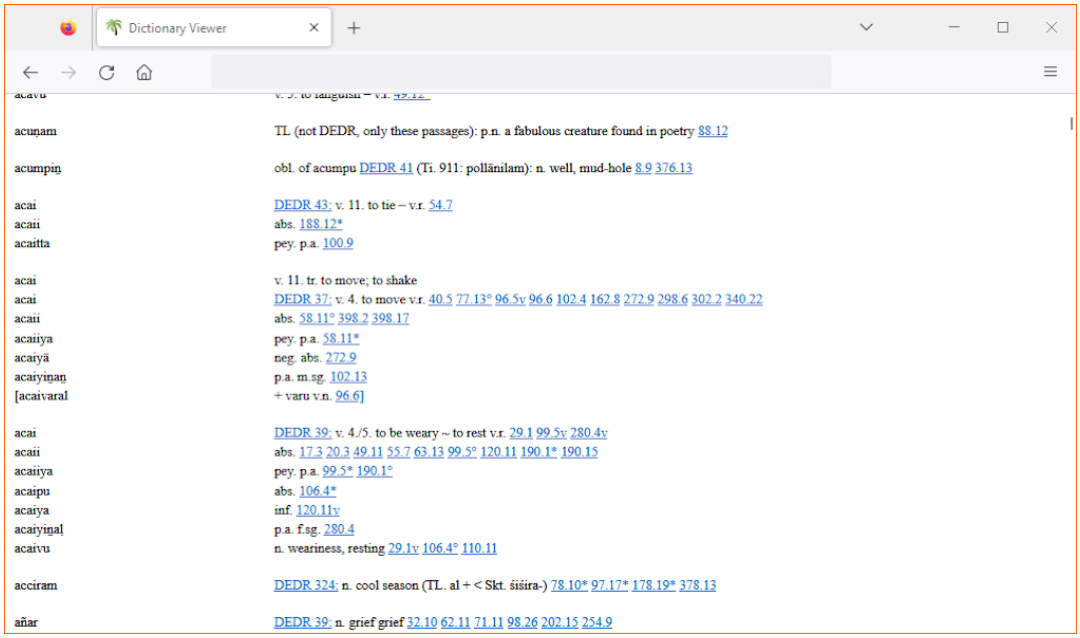


Figure 4.19.: Word index displayed by our viewer containing links to DEDR and transliterations

```xml
<?xml version="1.0" standalone="yes"?>
<TEI xmlns="http://www.tei-c.org/ns/1.0">
  ...
  <text>
    <body>
      <div>
        <head>A</head>
        <superEntry>
          <entry>
            <form>
              <orth>acai</orth>
            </form>
            <def>DEDR 37: v. 4. to move  v.r. 40.5 77.13&#xB0;
            <ref target="links/96_5">96.5v</ref>
            <ref target="links/96_6">96.6</ref>
            <ref target="links/102_4">102.4</ref>
            <ref target="links/162_8_5">162.8</ref>
            <ref target="links/272_9">272.9</ref>
            <ref target="links/298_6">298.6</ref>
            <ref target="links/302_2">302.2</ref>
            <ref target="links/340_22">340.22</ref>
            </def>
          </entry>
          ...
        </superEntry>
        ...
      </div>
    </body>
  </text>
</TEI>
```

Listing 17: Snippet of a TEI document that contains links to other documents

5. Publishing Collections of Research Data

TEI documents stored at the RDR and displayed by our viewer are read by those interested in the documents' contents for their own research and, thus, might want to refer to specific parts of the data. Using the DOI that is created for TEI documents that stem from the transformation of a Word DOCX document and are stored at the RDR is an option; however, it refers to a repository of TEI documents and not specific parts of the documents of interest. We extend our viewer, presented in Section 4.4, by collections that can be created and associated with unique identifiers, such as DOI, and present the extension in Section 5.1. Each collection contains a set of references to specific parts of research data, stored at the RDR at the Universität Hamburg and visualized by our viewer. Referring not only to text snippets, we extend our viewer by an annotation system presented in Section 5.2 that enables humanities scholars to annotate regions of images. As well as snippets of text, annotations can be added to a collection.

5.1. Creation of Collections

Recent trends show that publishing research data is considered good research practice and is increasingly mandated by funding agencies. Publishing research data allows for verification of results, reuse for other purposes, combination with other data, competitions and many other uses. Platforms such as Kaggle (Kaggle, 2022) or Zenodo (Sicilia, García-Barriocanal, and Sánchez-Alonso, 2017; Peters et al., 2017) allow sharing of research data with others. Sharing data first requires creating an account, then setting up a repository, and finally uploading the data. Uploaded data is associated with metadata to index it with an IR system and to allow users to decide whether the data fits their needs. Zenodo assigns each version of a repository a DOI that enables everyone to uniquely refer to it. This is important, as otherwise, other users are unable to verify results based on published research data, as the research data may have changed, leading to a potential change in the findings. A repository can contain research data of arbitrary format, ideally standardized, any number of files, and stem from one or more

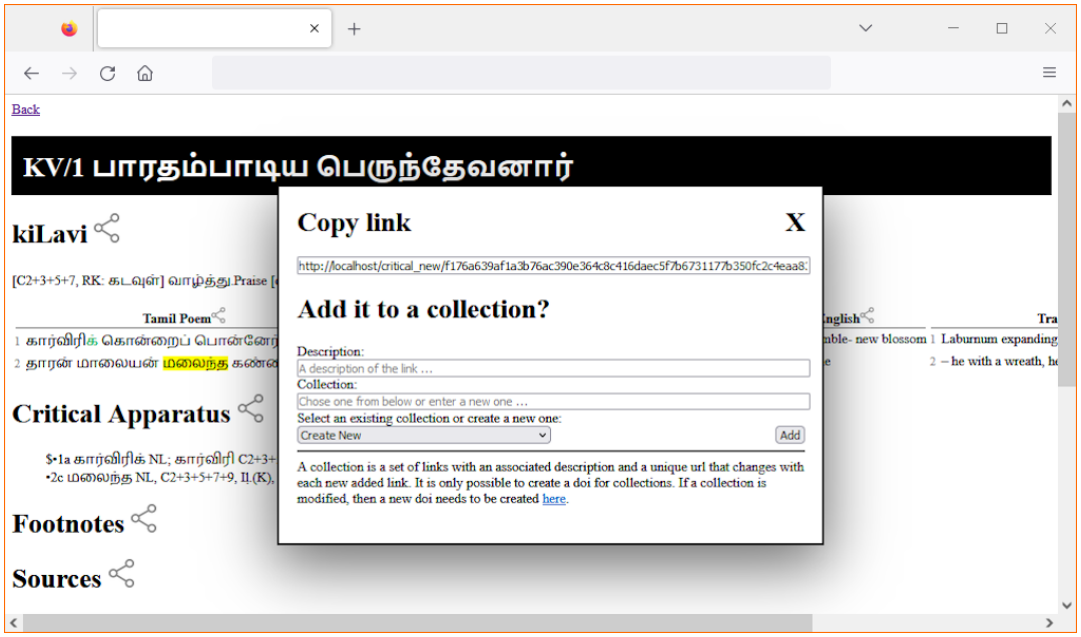


Figure 5.1.: Viewer extended with share buttons for the creation of collections

domains. Citing a repository of research data is only possible with a unique identifier, such as a DOI. In some cases, results depend on only a small subset of research data stored at a repository and citing the complete repository is not precise enough. One solution is to create a new repository that contains a subset of research data on which results depend and which is to be cited. This can lead to a collection of many small repositories containing subsets of research data from larger repositories that can only be used to produce results by the author of the repository. Larger ones would be more difficult to find. Instead, we argue that one should be able to directly refer to subsets of research data within a repository. This allows everyone to comprehend how results were produced from actual data. As repositories mainly contain research data that is in a format not primarily designed to be read by humans, a viewer is required that visualizes the data appropriately and allows referencing to any parts of the data regardless of its format. However, previewers for repositories hosted at Kaggle or Zenodo only show a small excerpt of uploaded data.

Hence, we extend our viewer with share buttons, as depicted in Figure 5.1, that allow adding parts of documents to a collection. If a collection is complete, one can associate it with a DOI that refers uniquely to the collection as long as the collection does not change. Collections not associated with a DOI are visible only to the collection's creator.

5.2. Annotation of Images

As depicted in Figure 4.18, our viewer displays linked pictures of palm leaves along with transcribed texts. Users may not only be interested in referring to pictures of palm leaves but also in referring to specific regions of pictures, annotating these, and adding them to a collection. This could be done by implicitly referring to specific regions of a picture with a description such as "at the top left of image eight" or more formally, as described in Section 4.3.1, by embedding references such as "{l. 8}" in the texts. As shown in Section 4.3.2, the latter are references that can be processed automatically. However, it is more precise for a humanities scholar to be able to create explicit links referring to textual annotations in regions of pictures. Explicit links are easy to process, and readers can follow links by simply clicking on them instead of searching for referred information using an IR system. Textual annotations themselves can be of arbitrary type and include descriptions, classifications, or transcriptions. The latter could be used for training models for optical character recognition (OCR). OCR requires huge amounts of data. This is rare in the case of texts written in languages that were spoken many centuries ago. Even if a model is not very precise, it can help humanities scholars transcribe texts from pictures, as rewriting automatically transcribed texts requires less work than starting from scratch. Hence, we find a loop in which humanities scholars are supported by AI methods, such as OCR, for the creation of data that is used for, among others, improving OCR models.

We use Annotorious, a flexible JS library for implementing a picture annotation system on the web (Annotorious, 2022). First, one needs to log in to determine whether one is allowed to view and annotate the pictures. The username is additionally used to discriminate between authors of annotations. After logging in, annotations can be created by left-clicking in the top left corner of the region to be annotated in the image. While still pressing the mouse button, the mouse is moved to the bottom right of the region to be annotated. Once the mouse button is released, a window pops up, as depicted in Figure 5.2. One can add a transcription or image description, comments on the annotation itself and tags.

Annotations, such as those depicted in Figure 5.2, are stored at Heurist, as shown in Figure 5.3. Storing annotations at Heurist is advantageous compared to other databases as Heurist is designed to be usable by humanities scholars. Hence, annotations are not only accessible by our viewer also by using Heurist.

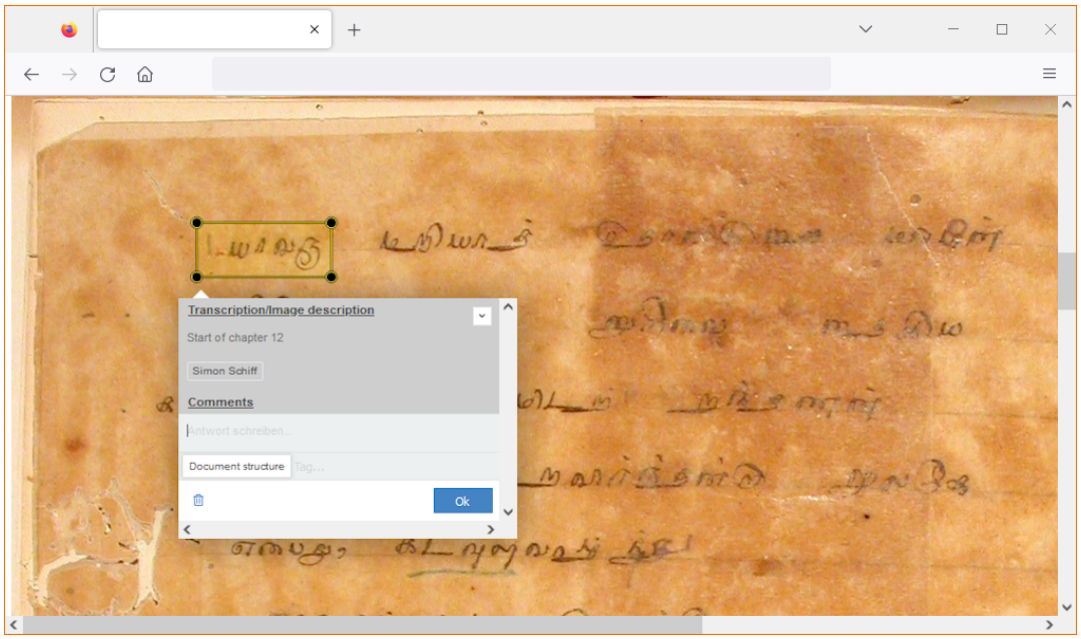


Figure 5.2.: Annotation of a picture in our viewer

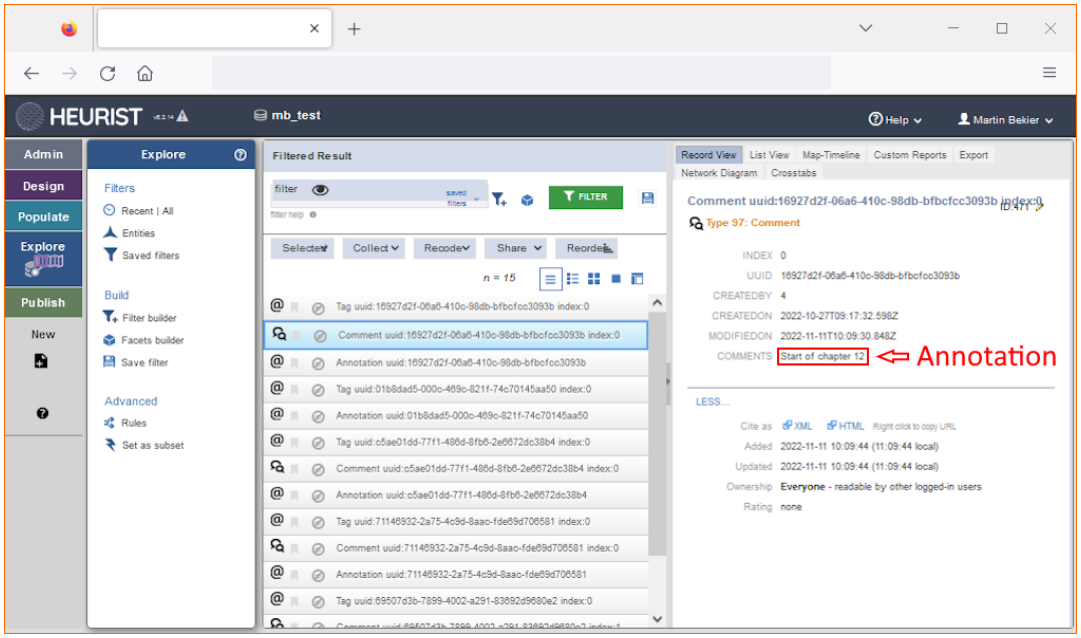


Figure 5.3.: Annotation listed at a Heurist database

6. Human Aware Information Retrieval

Humans have goals in life and reaching them sometimes requires specific information that they do not have in their brain. A human brain could be modeled with the ACT-R model (Anderson, 2007) which proposes that the brain consists of different modules, including the procedural module that can be seen as a central module controlling the behavior of a human by communicating with the other modules:

Visual Processing vision such as a picture of $3 \cdot x - 7 = 5$.

Aural Process sound such as that produced by a human asking for the solution to $3 \cdot x - 7 = 5$.

Goal The possible goals of a human are arbitrary. One of these could be to solve the equation $3 \cdot x - 7 = 5$.

Imaginal Might hold a representation of an intermediate equation such as $3 \cdot x = 12$.

Manual Control movements, such as using a calculator.

Declarative Retrieve information from somewhere in the brain that $3 \cdot 4 = 12$.

Vocal Generation of speech by telling another human that x equals 4.

Regardless of an individual's goals, the declarative module might fail to retrieve information from the brain as the brain might not "contain" the information. Thus, the human in question would use an IR agent to retrieve the required information, as illustrated in Figure 6.1. However, unlike the connected declarative module, the IR agent is not directly connected to the procedural module of a human. Even if the IR agent is directly connected, the information need from the procedural module needs to be translated before being sent to an IR agent. A human brain is inherently different from an IR agent. An IR agent can be written in any language and executed on a computer. From a high-level perspective, such a computer is modeled with the Turing machine and not with the ACT-R model. From a low-level perspective, the human brain consists of billions of neurons, while a computer consists of transistors. An IR agent has

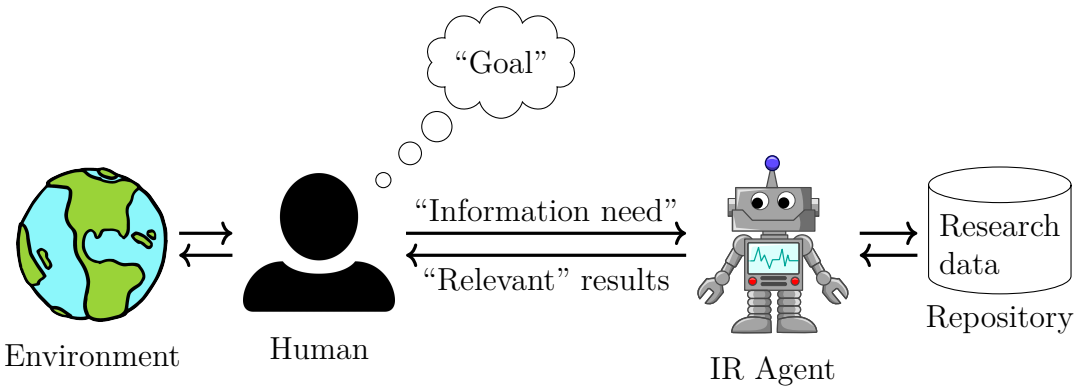


Figure 6.1.: Collaboration between a human and an IR agent

access to a repository of digital objects. Usually, these objects are texts written in any language, audio files, pictures, etc., containing information. If a human's information need could be translated to make it “compatible” with the IR agent, then the IR agent would need to match it with the objects in the repository. Every object could contain a piece of information that would satisfy the information need of the human. Comparing the translated information need of the human with the objects is possible as both the objects and the translation are “compatible” with the IR agent. However, comparing the translated information need of the human with the objects is not at all sufficient. The translated information need of the human is only a “request” of the procedural module to the declarative module and then compared with digital objects. The goal of the human, as well as anything related to the human creator of the digital object, is completely lost/ignored. If the IR agent returns a digital object back to the human, the human updates its brain with new information that the declarative module can access. Finally, the goal of the human might be reached or require access to other digital objects. Thus, while the information need of a human might change over time, it relates to the same goal of the human's previous information needs.

In Section 6.1, we give an overview of the different environments that are assumed when designing an IR agent followed by Section 6.2 where we present session search, an environment where an IR agent and a human collaborate to satisfy the information need of the human. During session search, the IR agent needs to act explicable which we show in Section 6.3 and to use additional information if available, to improve retrieval performance, which we present in Section 6.4 and Section 6.5. Finally, we show in Section 6.6 of how our proposed IR agent could be evaluated with datasets we present in Section 6.7.

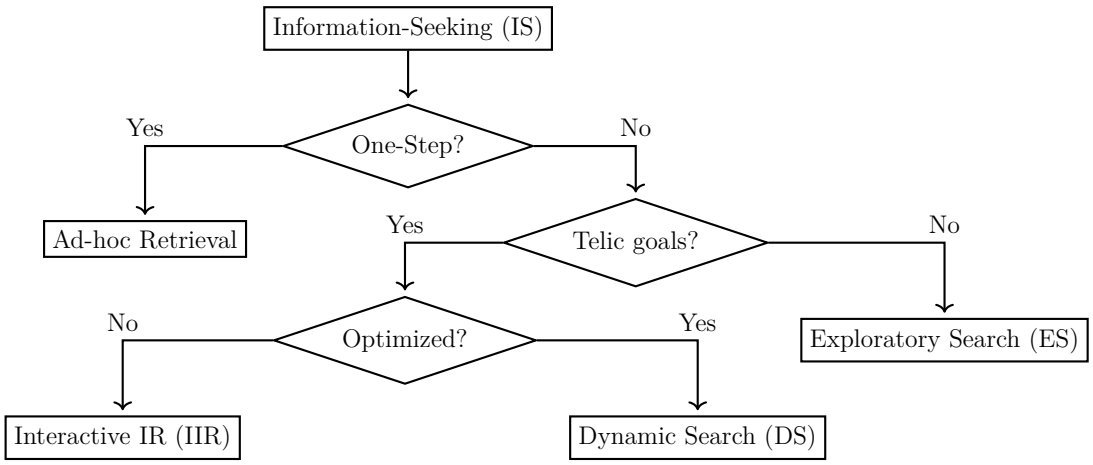


Figure 6.2.: Information-seeking overview (Tang and G. H. Yang, 2022)

6.1. Environment

IR algorithms, interfaces, etc., are designed, in particular, to support humans in seeking for information. Information seeking (IS) processes can be further differentiated, as depicted in Figure 6.2 (Tang and G. H. Yang, 2022). In the case of the "information need" of the human sent as a query to the IR agent has to be satisfied ad-hoc within one step, each query of the human is answered independently from others, which is okay if the IR agent returns mostly relevant information. If not, the IR agent can improve its performance depending on whether the human has a telic goal. If the human does not have a telic goal, the IR agent may return information the human might not have expected to receive. For instance, humans searching for news articles have no particular goal. Humans with telic goals for which the information need could not be satisfied ad-hoc within one step require more sophisticated methods in which the IR agent returns information linked to previous information it sent to the human as well as previous queries sent by the human to the IR agent. An IR agent may be unable to answer queries ad-hoc, either as it has poor performance or because the human was unable to express a precise enough query. The latter might be the case if the human learns how to express the query more precisely with each new piece of information received from the IR agent. Hence, if queries sent by a human to an IR agent depend on each other, the IR agent can answer each query depending on the previous query it has received (interactive information retrieval (IIR)) or aim to approximate the long-term goal of the human with each query (dynamic search (DS)).

6.2. Session Search

Initially, the human has a specific goal in mind that cannot be reached without a specific information, that the human has not in its brain. The missing information is fuzzy, so the human is initially unable to express the information need as a query. Instead, the human aims to express the query as well as possible and sends it to the IR agent, expecting it to retrieve information that will help to formulate the query more precisely. Given the query, the IR agent decides which information from a repository the IR agent has access to is relevant and sends it to the human. The human gives optional feedback based on the information retrieved from the IR agent. This process is repeated until the information need of the human is satisfied, or the IR agent gives up. Not only does the human learn from each new piece of information it receives, but the IR agent could also learn from each new query and optional feedback from the human. Feedback can be either explicit with the human indicating whether the received information is relevant or implicit by observing, among other things, how long a human spends on viewing information. It could also be blind, meaning that the IR agent assumes, for instance, that the information it found is relevant and aims to improve its performance based on the relevant information. Each round in which a human sends a query to the IR agent and the IR agent retrieves "relevant" information for the human based on the query, and the human, based on received information, gives optional feedback to the IR agent, is referred to as an iteration. A sequence of iterations is a session in which an IR agent collaboratively seeks for information with a human.

Many different kinds of problems can occur during the session as the IR agent and the human have to collaborate via queries, feedback, and results without being able to physically observe each other. If a human expresses a query, she or he has an approximate mental model of the IR agent, which influences how the human expresses a query. For instance, the human expects the query to be needed in a specific foreign language, the IR agent may only compare the query with the titles of documents, or the IR agent may weight some terms in the query too high such that the human omits them. However, the human's expectations may be wrong. The IR agent is aware that the human has specific expectations and aims to anticipate them. If the IR agent correctly anticipates them, it might be able to adapt its behavior. Adapting its behavior not only needs to satisfy the human's expectations. In addition, the behavior of the IR agent needs to be explicable from the human's perspective as, otherwise, the human cannot have an accurate approximate mental model of the IR agent and may face difficulties in expressing a query. The IR agent cannot correctly anticipate, adapt, and act explicably in regard to the requirements of every human for whom it collaboratively seeks information and is aware of that. If the IR agent is uncertain of whether it can collaborate with the human

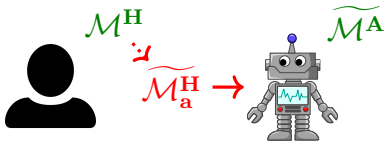


Figure 6.3.: IR agent's approximation of the human mental model $\mathcal{M}^{\mathbf{H}}$ as $\widetilde{\mathcal{M}^{\mathbf{H}}_{\mathbf{a}}}$ (Kambhampati, 2020)

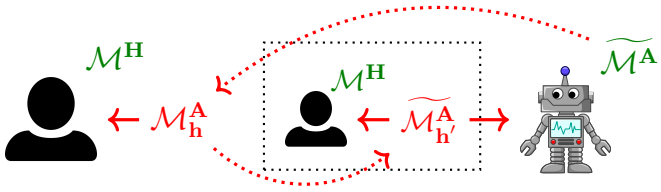


Figure 6.4.: The approximation $\widetilde{\mathcal{M}^{\mathbf{A}}_{\mathbf{h}'}}$ of the IR agent to the mental model $\widetilde{\mathcal{M}^{\mathbf{A}}_{\mathbf{h}}}$ that the human has about the IR agent $\widetilde{\mathcal{M}^{\mathbf{A}}}$ (Kambhampati, 2020)

or the human explicitly requests an explanation, it has to explain its behavior.

More formally, an IR agent and a human interact over a finite number t of time steps T in a session, with $t \in \mathbb{N}_0$. As proposed by Kambhampati et al. (Kambhampati, 2020; Y. Zhang et al., 2017) with a slightly different notation, while the human and the IR agent interact with each other, both are modeled as $\mathcal{M}^{\mathbf{H}}$ and $\mathcal{M}^{\mathbf{A}}$, respectively. Model $\mathcal{M}^{\mathbf{H}}$ contains the information need of the human that can only be approximated by the IR agent as $\widetilde{\mathcal{M}^{\mathbf{H}}_{\mathbf{a}}}$, as depicted in Figure 6.3 and model $\mathcal{M}^{\mathbf{A}}$ is represented as $\widetilde{\mathcal{M}^{\mathbf{A}}}$ and approximated by the human as $\mathcal{M}^{\mathbf{A}}_{\mathbf{h}}$. In addition, the IR agent approximates $\mathcal{M}^{\mathbf{A}}_{\mathbf{h}}$ as $\widetilde{\mathcal{M}^{\mathbf{A}}_{\mathbf{h}}}$, in order to reflect whether it has acted in an explicable manner or not, as depicted in Figure 6.4. As depicted in Figure 6.5, the IR agent incrementally updates its models $\mathbf{A}_t = (\widetilde{\mathcal{M}^{\mathbf{A}}}, \widetilde{\mathcal{M}^{\mathbf{H}}_{\mathbf{a}}}, \widetilde{\mathcal{M}^{\mathbf{A}}_{\mathbf{h}}})$ at each time step t, given a query Q_t and possibly available feedback F_t. The human updates its models $\mathbf{H}_t = (\mathcal{M}^{\mathbf{H}}, \mathcal{M}^{\mathbf{A}}_{\mathbf{h}})$ given a result R_t that the IR agent has selected and may satisfy the information need of the human.

At each time step t during a session, the IR agent's $\mathbf{A}_t$ and the human's mental models $\mathbf{H}_t$ are in a specific state that changes with each time step t. Thus, a session S is a sequence of session states $S = (S_t)_{t=0}^{T}$, in which each session state S_t contains the current query Q_t, feedback F_t expressed by the human, given results R_{t-1} the IR agent has sent to the human in the previous session state, results R_t that might satisfy the information need of the human, and updated mental models $\mathbf{H}_t$ of the human and $\mathbf{A}_t$ of the IR

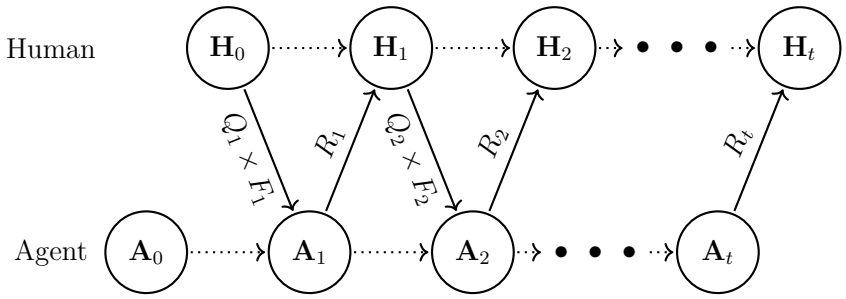


Figure 6.5.: Collaboratively seek for information during a session S_t

agent.

A session S at time step t is successful if R_t in S_t satisfies the information need of the human. Each session starts with initial, possibly non-empty, models $\mathbf{H}_0$ and $\mathbf{A}_0$ at time step $t = 0$. Then, at the next time steps $t \geq 1$, the IR agent first updates its models $\mathbf{A}_{t-1}$, given query Q_t and optionally available feedback F_t:

$$\mathbf{A}_{t-1} \times Q_t \times F_t \longrightarrow \mathbf{A}_t$$

and then selects result R_t, depending on $\mathbf{A}_t$:

$$\mathbf{A}_t \longrightarrow R_t$$

The human receives R_t and updates its models:

$$\mathbf{H}_{t-1} \times R_t \longrightarrow \mathbf{H}_t$$

and then expresses the information need as query Q_{t+1} and feedback F_{t+1}:

$$\mathbf{H}_t \longrightarrow Q_{t+1} \times F_{t+1}$$

Session S helps the agent close the gap between the human models $\mathbf{H}_t$ and its own models $\mathbf{A}_t$ over time and thus anticipate the IR goal of the human.

As noted by Kambhampati et al., collaboration between a human and an agent works only if both have an approximate mental model of each other with a small gap in between (Kambhampati, 2020). Thus, the human and the IR agent need to close the gap over time by sharing their models with each other. However, while the IR agent has a set of models with a concrete representation, the human does not. Both cannot directly interpret the state of the other's model and therefore need to share their models by sending/receiving queries, results, and feedback to each other. The challenging task is to

1. find a language for the queries, results, feedback, and objects stored in a repository the IR agent has access to,
2. find good representations for $\mathbf{H}_t$ and $\mathbf{A}_t$ and incrementally update them at each time step, and
3. evaluate the IR agent.

6.3. Words that Make up the Context

In this section, we present an IR agent that not only ranks uploaded documents, given a query, by relevance in descending order, it additionally returns for each document and score an explanation, in order to act legible. We assume that our IR agent has access to a set of documents D that are part of a corpus C. Each document D is a sequence of words $\langle w_1^D, \ldots, w_n^D \rangle$ of length n. We assume that the surrounding words $\{w_j^D \mid j - r \leq j \leq j + r\}$ of a word w_j^D within a given radius r make up the context of the word w_j^D. As depicted in Figure 6.6, document D is a sequence of words $\langle w_1^D, \ldots, w_{13}^D \rangle$. The context is highlighted with blocks of red crossed lines, each for the words w_5^D, w_6^D, w_7^D, and w_8^D respectively. For instance, the words that make up the context for the word w_7^D are $\{w_3^D, \ldots, w_{11}^D\}$ are depicted in the third row of Figure 6.6. In Algorithm 1, we show how to initially compute the words that make up the context for each word in every document in the corpus. The result is a mapping c that maps all words $w_j^{D_i}$ to a set of words that make up the context: $c : w_j^{D_i} \rightarrow \{w_{j-r}^{D_i}, \ldots, w_{j+r}^{D_i}\}$, with D_i being the current document, j being the position of word $w_j^{D_i}$ in document D_i, and r the radius. All sets of words in one document possibly have similar sets in other documents. We measure the similarity of two sets by the size of their intersection (i.e., the number of words they have in common). If it is above a given threshold t, then we assume that both contexts are, to a certain extend, similar.

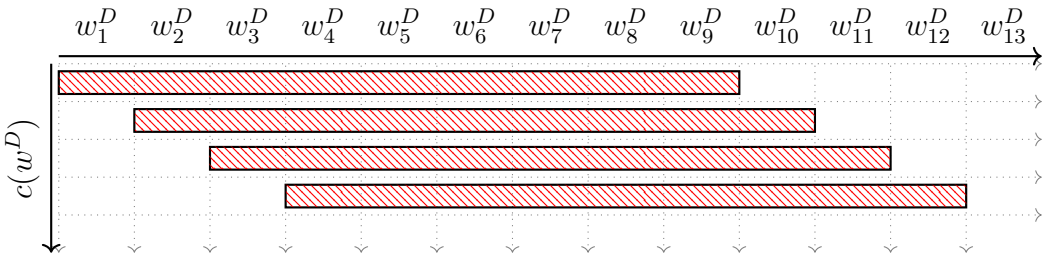


Figure 6.6.: Window function over a sequence of words $\langle w_1^D, \ldots, w_{13}^D \rangle$ with $r = 4$

Algorithm 1 Compute windows

procedure CONTEXTWINDOWS(C, r) ▷ Corpus C and radius r
  $c : w_j^{D_i} \rightarrow \left\{w_{j-r}^{D_i}, \ldots, w_{j+r}^{D_i}\right\}$
  for all $D_i \in C$ **do**
    REMOVESTOPWORDS(D_i) ▷ Remove stop words from document D_i
    for $j \leftarrow r$ to $|D_i| - r$ **do** ▷ Length $|D_i|$ of document D_i
      $c\left(w_j^{D_i}\right) \leftarrow \{\}$
      for $k \leftarrow j - r$ to $j + r$ **do**
        $c\left(w_j^{D_i}\right) \leftarrow c\left(w_j^{D_i}\right) \cup \left\{w_k^{D_i}\right\}$
  return c ▷ Return mapping c

Algorithm 2 returns a mapping r, mapping words in all documents to text snippets in the same context in other documents if the similarity is above a given threshold t. A human interested in text snippets from a similar context sends a word as a query to

Algorithm 2 Compute results

procedure COMPUTERESULTS(c, t) ▷ Contexts c and threshold t
  $r : w_j^{D_i} \rightarrow \left\{c\left(w_1^{D_1}\right), \ldots, c\left(w_l^{D_k}\right)\right\}$
  for all $w_j^{D_i} \in c$ **do**
    for all $w_l^{D_k} \in c$ **do**
      $i \leftarrow \left|c\left(w_j^{D_i}\right) \cap c\left(w_l^{D_k}\right)\right|$
      if $D_i \neq D_k$ and $i \geq t$ **then**
        $r\left(w_j^{D_i}\right) \leftarrow r\left(w_j^{D_i}\right) \cup c\left(w_l^{D_k}\right)$
  return r ▷ Return mapping r

our IR agent. Our IR agent returns all documents of similar context, given a word as a query, that contain text snippets returned by mapping r, with respect to the similarity of the text snippets in descending order.

Given the j-th word $w_j^{D_i}$ in document D_i, the context of the word $c\left(w_j^{D_i}\right)$, a text snippet $c\left(w_l^{D_k}\right) \in r\left(w_j^{D_i}\right)$, with $\left(w_l^{D_k}\right) = \left\{w_{l-r}^{D_k}, \ldots, w_{l+r}^{D_k}\right\}$, from another document $D_k \neq D_i$, and radius r, our IR agent must explain why it has returned the document D_k as a result for the query $w_j^{D_i}$. It generates an explanation for each document D_k in the result set by returning an excerpt from each document that contains the words $c\left(w_l^{D_k}\right) \in r\left(w_j^{D_i}\right)$. Each excerpt is a sequence of words containing $c\left(w_l^{D_k}\right)$, of which

$c\left(w_j^{D_i}\right) \cap c\left(w_l^{D_k}\right)$ are emphasized. The human can decide to send another query to the IR agent and to change the radius r or the threshold t. Even if results do not satisfy the information need of the human, the IR agent can act in a legible manner from the perspective of the human. It is possible for a human to change both r and t to a value that may lead to more sophisticated results, as the IR agent explains how it computes a result.

6.4. Collections of Text Paragraphs and Annotations

As presented in Chapter 5, humans can create collections of links, each linking to paragraphs of text. Hence, each collection represents regions of interest (ROI) pointing to research data that an IR agent could potentially use. Given a query, the IR agent compares it with all paragraphs that are part of an ROI and assigns a score to them. If the highest score of an ROI is above a given threshold, the IR agent assumes that the human has a similar interest in the research data as the creator of the ROI. The IR agent assigns a score to each document containing paragraphs that are part of the identified ROI. Documents with a high score contain many such paragraphs and documents with a low score only contain a few. However, the ROI might differ from the information need of the human sending the query to the IR agent. Therefore, besides already computed scores, the IR agent compares all documents in the repository with the query it has received using TF.IDF (Rajaraman and Ullman, 2011). A parameter between zero and one that needs to be adjusted carefully controls the proportion of both scores. If an IR agent uses an ROI, it can send it to the human in addition to the results it has computed. As each ROI in a collection has a description created by its author, the human can check whether it matches its information need and give feedback. The IR agent can then adjust the parameter accordingly.

More formally, a human sends a query Q to an IR agent with access to a corpus C of documents $D \in C$ and a collection of $ROI = \{(D_i, st, en) \mid D_i \in C \wedge D_i = \langle w_1^{D_i}, \ldots, w_{st}^{D_i}, \ldots, w_{en}^{D_i}, \ldots, w_n^{D_i} \rangle\}$ that is created by humans. The IR agent approximates $\mathcal{M}^{\mathbf{H}}$ as $\widetilde{\mathcal{M}_{\mathbf{a}}^{\mathbf{H}}}$ from query Q and the IR agent's mental model $\widetilde{\mathcal{M}^{\mathbf{A}}}$ is modeled with the collections. A human collaboratively seeking information with an IR agent might then correctly approximate $\widetilde{\mathcal{M}^{\mathbf{A}}}$ as $\mathcal{M}_{\mathbf{h}}^{\mathbf{A}}$ as the IR agent sends next to found documents, a ROI it has used to rank the documents. The IR agent assumes that the human approximates $\widetilde{\mathcal{M}^{\mathbf{A}}}$ as $\mathcal{M}_{\mathbf{h}}^{\mathbf{A}}$ and thus approximates $\mathcal{M}_{\mathbf{h}}^{\mathbf{A}}$ as $\widetilde{\mathcal{M}_{\mathbf{h}}^{\mathbf{A}}}$, from the feedback it receives. Given a query Q, the IR agent computes the most probable $ROI_{\max}$ of the collection, as listed in Algorithm 3. Given $ROI_{\max}$, the IR agent computes for each document D_i

Algorithm 3 Estimate most probable collection $ROI_{\max} \in Collection$, given query Q

procedure ESTIMATEMOSTPROBABLECOLLECTION($Collection, Q$)
  $ROI_{\mathbf{max}} \leftarrow \emptyset, s_{\max} \leftarrow 0$
  for all $ROI_i \in Collection$ **do** ▷ Loop through all collections
    $s \leftarrow 0$
    for all $(D_j, st, en) \in ROI_i$ **do**
      $s \mathrel{+}= \left|\left\langle w_{st}^{D_j}, \ldots, w_{en}^{D_j} \right\rangle \cap Q\right| / \left|\left\langle w_{st}^{D_j}, \ldots, w_{en}^{D_j} \right\rangle \cup Q\right|$
    $s \leftarrow s/\left|ROI_i\right|$
    if $s > s_{\max}$ **then**
      $ROI_{\max} \leftarrow ROI_i$
      $s_{\max} \leftarrow s$
  return $ROI_{\max}$ ▷ Return most probable ROI

a score $SCORE(D_i)$, as listed in Algorithm 4. Finally, the IR agent ranks documents

Algorithm 4 Compute scores for documents $D \in C$ given $I_{\max}$

procedure COMPUTESCOREFORDOCUMENTS($C, ROI_{\max}$)
  $SCORE : D_i \rightarrow 0$ ▷ Function that assigns scores to all $D_i \in C$
  for all $D_i \in C$ **do**
    $SCORE(D_i) \leftarrow |\{D_i \mid (D_i, st, en) \in ROI_{\max}\}|$
  $m \leftarrow \text{MAX}((SCORE(D_i)))$ ▷ Get max score m of an ROI
  for all $D_i \in C$ **do**
    $SCORE(D_i) \leftarrow SCORE(D_i)/m$ ▷ Normalize scores
  return $SCORE$

$D_i \in C$, using $score(D_i)$, as listed in Equation (6.1).

$$score(D_i) = \alpha \cdot SCORE(D_i) + (1 - \alpha) \cdot TF.IDF(Q, D_i) \tag{6.1}$$

If $SCORE(D_i)$ is not above a given threshold τ, then the IR agent sets α to zero.

6.5. Subjective Content Descriptions

As shown in Section 6.4, a collection helps a human-aware IR agent improve its performance. However, the assumption is that collections of ROIs already exist which is

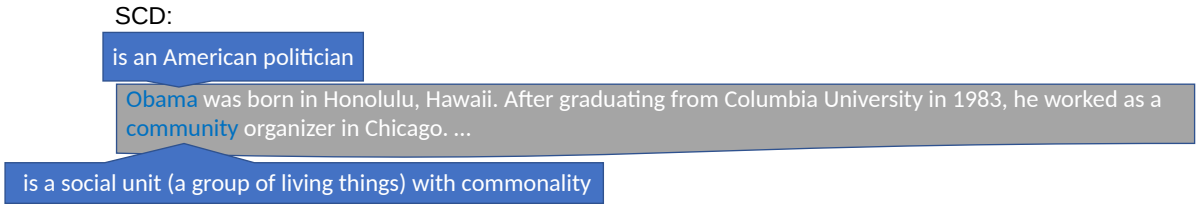


Figure 6.7.: Text with associated SCDs

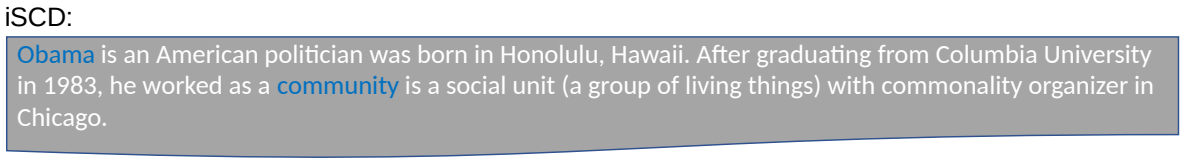


Figure 6.8.: Text containing iSCDs

often not the case. Next to collections of ROI, documents might contain subjective content descriptions (SCDs) (Kuhr et al., 2019), which are textual annotations of texts, as depicted in Figure 6.7. Content descriptions (CDs) are subjective, as different humans create different CDs for the same texts. They are similar to sticky notes attached to a printed document to add additional information to texts. Some texts, such as that depicted in Figure 6.8, contain inline subjective content descriptions (iSCDs) that can be found within a text (Bender et al., 2021). Even though they are located in the texts, one can always distinguish between texts of the documents and texts of iSCDs. However, words part of an iSCD are not always distinguishable from the rest of the texts in the document, as depicted in Figure 6.9. In that case, the IR agent cannot make use of any iSCD to improve its performance, even if they are available.

As described in Section 4.3.1, authors of poems have not only written poems on palm leaves but also commented on them. Comments are only distinguishable from the rest of the text, written on a palm leaf, if one knows the poem itself. This is identical to the text from Figure 6.9 in which it is unknown whether and where it contains iSCDs. From

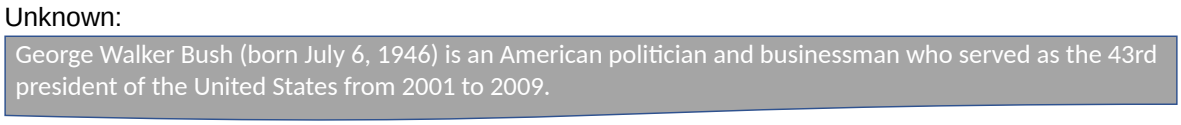


Figure 6.9.: Text in which it is unknown whether and where it contains iSCDs

another perspective, comments are iSCDs, interleaved with poems that make up the actual content of a document. Identifying iSCDs automatically is beneficial as they help improve the performance of a human-aware IR agent and support humanities scholars in identifying comments within a text.

To train a model that discriminates words part of an iSCD, such as comments from words used in a poem, and to evaluate it, we create a corpus C of documents $D_i \in C$, each containing a poem with comments as interleaved iSCDs, as described in Section 3.4. Documents in which words that are part of an iSCD are filtered out are denoted as $\widetilde{D}$. Documents in which words that are part of a poem are filtered out are denoted as $\widehat{D}$. The corpus is split into a train C_{train} and test corpus C_{eval}. The former is used to train a model and the latter to evaluate it. The following steps are performed offline to train a model:

1. Computation of a matrix of word distributions over train corpus C_{train} of documents $\widetilde{D_i} \in C_{train}$, using Algorithm 5.

2. Computation of sequences of similarity values over documents D_i in train corpus C_{train}, using the matrix of word distributions and Algorithm 6, so that each word distribution has an associated similarity value. Similarity values of each distribution are then discretized using the function in Equation (6.2).

3. Training of an HMM, depicted in Figure 6.11, using discretized similarity values associated with each word as evidence and words as either part of a poem or iSCD as state.

Finally, the following step is conducted online using the trained HMM to classify whether a word is part of a poem or an iSCD:

4. Given a new document $D_{new} \in C_{eval}$, Algorithm 7 uses the trained HMM, matrix of word distributions, and the function in Equation (6.2), to classify words as either part of a poem or of an iSCD.

Matrix of Word Distributions

Each document $\widetilde{D} \in C_{train}$ is a sequence of words $\langle w_1^{\widetilde{D}}, \ldots, w_{|\widetilde{D}|}^{\widetilde{D}} \rangle$ of length $|\widetilde{D}|$ without any iSCDs. All sequences of words in each document are separated into subsequences of words by a sliding, i.e. overlapping, window of length wr. Each subsequence is represented as a function *vector* that maintains a word count of words appearing within a subsequence. All vectors are finally normalized so that all counts of words sum up to one, as listed in Algorithm 5. For instance, a document $D = \langle w_1, w_1, w_2, w_2, w_2, w_3, w_4 \rangle$

Algorithm 5 Computation of word distribution *matrix* (Bender et al., 2021)

1: **function** COMPUTEMATRIX(C, wr)
2: $matrix \leftarrow \emptyset$
3: **for all** $\widetilde{D_i} \in C_{train}$ **do**
4: **for** $(start \leftarrow 0 - wr/2; start < \left|\widetilde{D_i}\right| - wr/2; start \mathrel{+}= 1)$ **do**
5: $vector : w^{\widetilde{D_i}} \rightarrow \mathbb{R}$
6: **for** $(index \leftarrow start; index \leq start + wr; index \mathrel{+}= 1)$ **do**
7: $vector\left(w^{\widetilde{D_i}}\right) \mathrel{+}= 1$
8: $m \leftarrow \text{MAX}(vector)$ ▷ Get max count of all words in the vector
9: **for all** $w^{\widetilde{D_i}} \in vector$ **do**
10: $vector\left(w^{\widetilde{D_i}}\right) \leftarrow vector\left(w^{\widetilde{D_i}}\right)/m$
11: $matrix \leftarrow matrix \cup vector$
12: **return** $matrix$

start	Word count *vector*	Normalized vector
0	$vector(w_1) = 1.0$	$vector(w_1) = 1$
1	$vector(w_1) = 2.0$	$vector(w_1) = 1$
2	$vector(w_1) = 2.0, vector(w_2) = 1.0$	$vector(w_1) = 2/3, vector(w_2) = 1/3$
3	$vector(w_1) = 2.0, vector(w_2) = 2.0$	$vector(w_1) = 1/2, vector(w_2) = 1/2$
4	$vector(w_1) = 1.0, vector(w_2) = 3.0$	$vector(w_1) = 1/4, vector(w_2) = 3/4$
⋮	⋮	⋮

Table 6.1.: Example for Algorithm 5

is the input of Algorithm 5 and the size of the window is $wr = 4$. If the for loop in line 4 runs five times, the result is listed in Table 6.1. The column "normalized vector" is the output *matrix* of Algorithm 5, a set containing each row. Each element in the matrix can be compared to a subsequence of text of a document $D_{new} \in C_{eval}$. If even the best match does not have a high similarity with a subsequence of D_{new}, then it is most likely a sequence containing an iSCD. Window size wr is a parameter that needs to be chosen carefully during training.

Sequences of Similarity Values

Given a word distribution matrix $matrix$, we compute sequences of similarity values for each document $D \in C_{train}$ containing poems and interleaved iSCDs, as listed in Algorithm 6. For instance, as depicted in Figure 6.10, a document $D = \langle w_1^D, \ldots, w_{11}^D \rangle$

Algorithm 6 Estimate similarity sequence (Bender et al., 2021)

```
1:  function estimateSimilaritySequence(D, wr, matrix)
2:      sim ← ⟨⟩
3:      for start ← 0 − wr/2; start < |D| − wr/2; start += 1 do
4:          vector_current : w^D → ℝ
5:          for index ← start; index ≤ start + wr; index += 1 do
6:              vector_current (w^D_index) += 1
7:          m ← max(vector_current)
8:          for all w^D ∈ vector_current do
9:              vector_current ← vector_current/m
10:         win_max ← ⟨⟩, s_max ← 0
11:         for vector ∈ matrix do
12:             s ← compare(vector, vector_current)
13:             if s > s_max then
14:                 s_max ← s
15:         sim ← sim ∘ sim_max
16:     return sim
```

contains an iSCD from word w_4^D to w_8^D. Document D is separated into subsequences that are compared with vectors that where computed, using Algorithm 5 with document $\widetilde{D} \in C$ in which all words that are part of an iSCD were filtered out. Hence, subsequences of document D containing many words that are part of an iSCD most likely have low similarity with subsequences that stem from documents $\widetilde{D}$, as depicted on the right-hand side of Figure 6.10, in which the similarities of words w_1^D to w_{11}^D are visualized. The similarity values are:

$$sim = \langle 0.96, 0.96, 0.96, 0.95, 0.84, 0.51, 0.35, 0.18, 0.55, 0.86, 0.98 \rangle$$

All possible similarity values are the observable states of the to-be-trained HMM. However, an infinite number are possible, while an HMM can only have a finite number. Even though the number of states needs only to be finite, only a few should be chosen, as the computation can otherwise become infeasible. We use the discretization function in

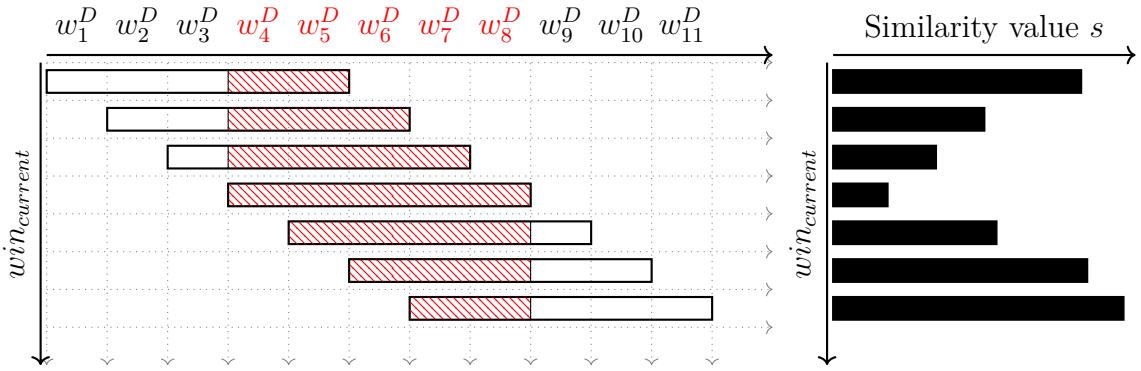


Figure 6.10.: Sliding window with range $wr = 5$ sliding over document D in which words w_4^D to w_8^D represent an iSCD accompanied by corresponding similarity values on the right (Bender et al., 2021)

Equation (6.2) to have n possible observation states with parameters θ_n. Each similarity value x is mapped to observation state y_n if x is maximal with respect to parameter θ_n.

$$f(x) = \begin{cases} y_1 & 0 \leq x < \theta_1 \\ y_2 & \theta_1 \leq x < \theta_2 \\ \dots & \dots \\ y_n & \theta_{n-1} \leq x < 1 \end{cases} \tag{6.2}$$

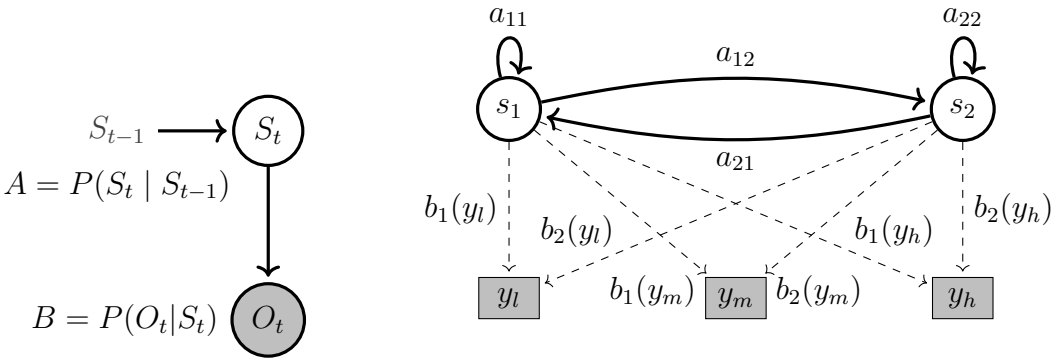
Similarity values *sim* are discretized by the function, listed in Equation (6.3), to have three observable states:

$$O = \langle y_h, y_h, y_h, y_h, y_h, y_m, y_m, y_l, y_m, y_h, y_h \rangle$$

with $\theta_1 = 0.3$ and $\theta_2 = 0.7$ for example. Here, y_l stands for low, y_m for middle, and y_h for high.

$$f(x) = \begin{cases} y_l & 0 \leq x < \theta_1 \\ y_m & \theta_1 \leq x < \theta_2 \\ y_h & \theta_2 \leq x < 1 \end{cases} \tag{6.3}$$

The number of states n, as well as the values for θ_i with $0 \leq i \leq n$ need to be adjusted carefully, as we will show later in Section 6.5.



(a) One-time slice graph (b) State-transition system

Figure 6.11.: HMM with one hidden variable S_t with two possible states $\{s_1, s_2\}$ emitting an observation O_t with three possible values $\{y_l, y_m, y_h\}$ (Bender et al., 2021)

Train a Hidden Markov Model

We use the similarity values computed with Algorithm 6 as observable states and subsequences of words classified as either part of a poem or iSCD as hidden states from training corpus C_{train}, for training an HMM with the Baum-Welch algorithm (Welch, 2003). For instance, the HMM depicted in Figure 6.11 has $S_t = \{s_1, s_2\}$ as state variable and $O_t = \{y_l, y_m, y_h\}$ as evidence variable. Only two states s_1 and s_2 are possible at in which s_1 is the state in which a subsequence of words of length wr belongs to a poem and s_2 in which they belong to an iSCD. Evidence O_t is either y_l, y_m, or y_h and stems from the discretization of similarity values. We assume that the first word of a document is most likely a content word and not part of an iSCD.

Identify Inline Subjective Content Descriptions (iSCDs) with a Hidden Markov Model

For each document D_{new}, we can use similarity values from Algorithm 6, a trained HMM, and Algorithm 7. First, we use Algorithm 6 with D_{new}, wr of the same value as used for training the HMM and $matrix$ from Algorithm 5 to estimate a sequence of similarity values that are discretized with the function listed in Equation (6.2). If subsequences in D_{new} have a high similarity, then words within them are mostly content words, whereas words within sequences of low similarity are more likely part of an iSCD. Discretized similarity values O of document D_{new} are the evidence variables we use together with the trained HMM and the Viterbi algorithm (Forney, 1973) to approximate the most likely

Algorithm 7 Estimate iSCDs using similarity sequences and a trained HMM

function ESTIMATEISCDS(λ, f, D_{new}, wr, $matrix$)
  $O \leftarrow \langle\rangle, sim \leftarrow$ ESTIMATESIMILARITYSEQUENCE$(D_{new}, wr, matrix)$
  for all $s \in sim$ **do**
    $O \leftarrow O \circ f(s)$ ▷ Discretize similarity values
  $S \leftarrow$ VITERBI(λ, O) ▷ Compute the most likely state sequence
  $iscds \leftarrow \emptyset, index \leftarrow 0$
  for all $state \in S$ **do** ▷ Iterate over S
    if $state = s_2$ **then** ▷ s_2 corresponds to "iSCD"
      $iscds \leftarrow iscds \cup \left\langle w_{index}^{D_{new}}, \ldots, w_{index+wr}^{D_{new}} \right\rangle$
      $index$ += 1
  return $iscds$

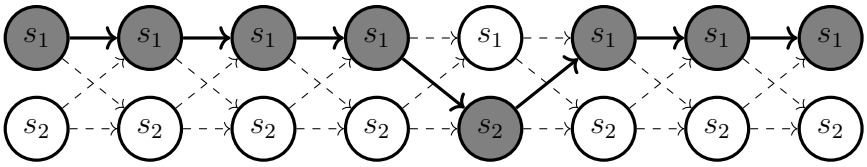


Figure 6.12.: Example trellis of the HMM, where s_2 depicts a commentary, in Figure 6.11 (Bender et al., 2021)

sequence of hidden states s_1 and s_2, namely whether a subsequence contains content words s_1, or words of an iSCD s_2.

For instance, the sequence

$$O = \langle y_h, y_h, y_h, y_h, y_h, y_m, y_m, y_l, y_m, y_h, y_h \rangle$$

is the evidence for the HMM, depicted in Figure 6.11. The output by the HMM is

$$S = \{s_1, s_1, s_1, s_1, s_2, s_1, s_1, s_1\},$$

as depicted in Figure 6.12. Words of one subsequence are most probably part of an iSCD.

Precision and Recall in iSCD Identification

We created a dataset of 91 documents $D \in C$, with $C = C_{train} \cup C_{eval}$ and split it into 80% training data C_{train}, and 20% test data C_{eval} from a critical edition, created by Eva Wilden in Wilden (2018) for the cross-evaluation of our approach, as described in

Section 4.3.1. An HMM is trained with two hidden states $S = \{s_1, s_2\}$, and five evidence states $O = \{y_1, y_2, y_3, y_4, y_5\}$. Discretized similarity values are computed for documents $\widetilde{D} \in C_{train}$ with Algorithm 5 and documents $\widehat{D} \in C_{train}$ with Algorithm 6, respectively. Our evaluation shows that the discretization function in Equation (6.4) leads to good results for training an HMM (Bender et al., 2021).

$$f(x) = \begin{cases} y_1 & 0 \leq x < 0.05 \\ y_2 & 0.05 \leq x < 0.1 \\ y_3 & 0.1 \leq x < 0.15 \\ y_4 & 0.15 \leq x < 0.2 \\ y_5 & 0.2 \leq x < 1 \end{cases} \tag{6.4}$$

We evaluate our approach on test data C_{eval} and compare it with three other approaches:

Word-based Each document D in training data C_{eval} contains poem words $\mathcal{V}_{\widetilde{D}}$ only from $\widetilde{D}$, iSCD words $\mathcal{V}_{\widehat{D}}$ only from $\widehat{D}$, or words $\mathcal{V}_{\widetilde{D}} \cap \mathcal{V}_{\widehat{D}}$ from both $\widetilde{D}$ and $\widehat{D}$. A word appearing only in $\mathcal{V}_{\widetilde{D}}$ is most likely a poem word, and a word appearing only in $\mathcal{V}_{\widehat{D}}$ is most likely a word part of an iSCD, in a document $D_{new} \in C_{eval}$. For words appearing in $\mathcal{V}_{\widetilde{D}} \cap \mathcal{V}_{\widehat{D}}$, it is not obvious whether they are poem words or words of an iSCD in document $D_{new} \in C_{eval}$. We estimate that a word $w_i \in D_{new}$ is a poem word by $P(w_i \in \widetilde{D}_{new}) = \#\mathcal{V}_{\widetilde{D}} / \#(\mathcal{V}_{\widetilde{D}} \cup \mathcal{V}_{\widehat{D}})$ and a word is part of an iSCD with $P(w_i \in \widehat{D}_{new}) = \#\mathcal{V}_{\widehat{D}} / \#(\mathcal{V}_{\widetilde{D}} \cup \mathcal{V}_{\widehat{D}})$.

Threshold-based We use Algorithm 6 to compute a similarity sequence from a document $D_{new} \in C_{eval}$, as illustrated on the right-hand side of Figure 6.10. If a sequence of words has a low similarity value s below a threshold τ, then its words are most likely part of an iSCD; if a sequence has a high similarity value s above a given threshold τ, then its words are most likely part of a poem. Threshold t is chosen by training on C_{train}.

Initial HMM An HMM to be trained has initially chosen parameters. We use the following initial emission probabilities for iSCDs:

$$\{y_1 : 0.15, y_2 : 0.5, y_3 : 0.2, y_4 : 0.1, y_5 : 0.05\},$$

and for text:

$$\{y_1 : 0.05, y_2 : 0.05, y_3 : 0.3, y_4 : 0.35, y_5 : 0.25\}$$

	Tamil	
	Unstemmed	Stemmed
$\|C\|$	91	91
Avg $\|D\|$	73.2	73.2
# iSCDs	908	869
$\|\mathcal{V}_{\widehat{D}} \cup \mathcal{V}_{\widetilde{D}}\|$	8015	5783
$\|\mathcal{V}_{\widehat{\mathcal{D}}}\|$	5521	4304
$\|\mathcal{V}_{\widetilde{D}}\|$	2666	1987
$\|\mathcal{V}_{\widehat{D}} \cap \mathcal{V}_{\widetilde{D}}\|$	172	508

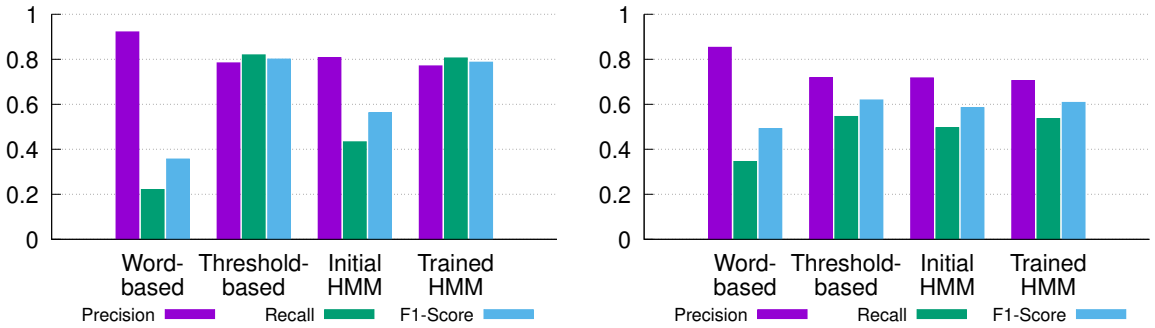
Table 6.2.: Characteristics of data sets divided into two different settings (Bender et al., 2021)

We evaluate our approach on documents at where words where stemmed, as well as on documents at where words where not stemmed, as listed in Table 6.2. There is not as wide a range of stemmer available for Tamil as there is for English, and certainly not for the Tamil spoken approximately 300 years ago. We use a freely available stemmer for modern Tamil created by Rajalingam in Rajalingam (2022), for stemming the words in the poems as well as part of an iSCD. The average length of a poem with interleaved iSCD D is 73.2 words. The number of iSCDs is reduced if words are stemmed, as some words are filtered out. Words in $\mathcal{V}_{\widehat{D}} \cup \mathcal{V}_{\widetilde{D}}$ are all words that occur in C_{train} and are reduced if they are stemmed, as stemming words leads to pairs of words that are equal if they are reduced to their stem. The more words that appear in both $\mathcal{V}_{\widehat{D}}$ and $\mathcal{V}_{\widetilde{D}}$ the more challenging it is to identify whether a word is part of a poem or an iSCD.

The result of our evaluation is depicted in Figure 6.13 with the left-hand side of Figure 6.13a showing results without stemming and the right-hand side of Figure 6.13b showing results with stemming. Scores for each approach are computed in terms of precision, recall, and an F1-score averaged by cross-validation, as listed in Equation (6.5).

$$\text{precision} = \frac{\text{tp}}{\text{tp} + \text{fp}} \quad \text{recall} = \frac{\text{tp}}{\text{tp} + \text{fn}} \quad \text{F1-score} = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{6.5}$$

Here tp stands for true positives, fp for false positives, and fn for false negatives. The word-based approach is superseded by the other three approaches, and the threshold-based one yields the best results. However, finding the right threshold is more complex than training an HMM.



(a) Without stemming

(b) With stemming

Figure 6.13.: Classification performance for HMM-, word-, and threshold-based approaches (Bender et al., 2021)

6.6. Evaluation of a Human-Aware IR agent

The most fitting evaluation method we found in the literature is part of the dynamic domain TREC 2017 conference (G. H. Yang et al., 2017). At the conference, a session-based IR agent was evaluated with a ground truth dataset that contains a wide range of topics handcrafted by experts. Each topic has a description and a set of subtopics. Subtopics contain sets of passages from documents that are part of a corpus the IR agent has access to and are each associated with a relevance score. The IR agent retrieves a description of a topic as an initial query and then has to return documents from the corpus, sorted by relevance in descending order. Documents returned by the agent are sent to a human simulator that has access to the ground truth dataset. The simulator returns feedback by comparing the ground truth dataset and the documents retrieved from the IR agent and evaluates whether a document is relevant. The IR agent decides when to stop the session. The cube test (CT) (Luo, Wing, et al., 2013) returns a score to compare different agents given the same ground truth dataset and corpus. Given a ground truth dataset and the current iteration of a session, the human simulator sends the same feedback to the IR agent for documents the simulator has retrieved. An IR agent could greatly improve its performance if it approximates $\mathcal{M}^{\mathbf{H}}$ as $\widetilde{\mathcal{M}^{\mathbf{H}}_{\mathbf{a}}}$. However, we argue that it is not useful for the agent to approximate $\mathcal{M}^{\mathbf{A}}_{\mathbf{h}}$ as $\widetilde{\mathcal{M}^{\mathbf{A}}_{\mathbf{h}}}$, as the human simulator always gives the same feedback for identical iterations, even if the complete history of two sessions differs.

Our aim is to extend the human simulator that is part of the evaluation at the dynamic domain TREC 2017 conference with a human agent so that the IR agent can improve its

performance by approximating $\mathcal{M}_{\mathbf{h}}^{\mathbf{A}}$ as $\widetilde{\mathcal{M}_{\mathbf{h}}^{\mathbf{A}}}$, in addition to approximating $\mathcal{M}^{\mathbf{H}}$ as $\widetilde{\mathcal{M}_{\mathbf{a}}^{\mathbf{H}}}$. The human simulator still has access to a ground truth dataset; however, the dataset only contains sets of relevant documents, each concerning a session. Expressing queries is now the task of the human agent that does not know which documents are selected as relevant by the human simulator. From a subset of these as relevant selected documents, the human simulator generates a subjective representation in form of weighted words that simulate the human's information need. The representation is subjective as the human simulator generates the weighted words depending on the context of documents selected as relevant. The human agent does not know which documents are deemed relevant by the human simulator and cannot go through all documents in the corpus. However, the human agent can observe the weighted words representing the information need. This is identical to an actual human who cannot compare all documents in a large corpus with a set of terms in his mind. From the set of weighted words, the human agent must express an information need as a query. As is the case for real humans, the human agent tries to anticipate how the IR agent will return documents following a query, as it only wants relevant documents to be retrieved. The human agent is allowed to reweight the weighted words before sending them to the IR agent as a query to retrieve a set of documents it could read. Only sending top-weighted words to the IR agent without adjusting them is not a good strategy, as the IR agent might not always be able to return relevant documents for a subset of words that only subjectively represent the information need of a human. Therefore, the human agent knows that the IR agent might not be able to understand his information need and needs to find a good strategy for expressing a query. The IR agent needs to anticipate how the human agent will reweight words before sending them as a query, while it does not have access to the set of weighted words. However, it can observe the titles and contents of all documents in the corpus and the query sent by the human agent. We argue that the IR agent needs to be aware of the fact that its actions influence the human agent's actions. If the human agent reads a document, the human simulator changes the weighted words, as the human agent has learned something new by reading it. Collaboration between both agents only works if both aim to share and approximate each other's mental models.

More concretely, the human agent does not compute $Q_{t+1} \times F_{t+1}$ from $\mathbf{H}_t$ directly; the IR agent does not so for R_t from $\mathbf{A}_t$ either. Both $\mathbf{H}_t$ and $\mathbf{A}_t$ contain models, and each model contains a function that maps, depending on the model, queries Q_t and feedback F_t or weighted words to queries Q_{t+1} and feedback F_{t+1} or weighted words, as depicted in Table 6.3. The human agent needs to express its information need as a query and give optional feedback for previously received results: $\mathbf{H}_t \rightarrow Q_{t+1} \times F_{t+1}$, with

Model	Function	From	To
$\mathcal{M}^{\mathbf{H}}$	$f^{\mathbf{H}\to\mathbf{A}}$	$\mathcal{M}^{\mathbf{A}}_{\mathbf{h}} \times W^{\mathbf{H}}$	$Q_{t+1} \times F_{t+1}$
$\mathcal{M}^{\mathbf{A}}_{\mathbf{h}}$	$f^{\mathbf{A}\to\mathbf{H}}_{\mathbf{h}}$	$Q_t \times F_t$	$W^{\mathbf{A}}_{\mathbf{h}}$
$\widetilde{\mathcal{M}^{\mathbf{A}}}$	$\widetilde{f^{\mathbf{A}\to\mathbf{H}}}$	$\widetilde{\mathcal{M}^{\mathbf{H}}_{\mathbf{a}}} \times Q_t \times F_t$	$W^{\mathbf{A}}$
$\widetilde{\mathcal{M}^{\mathbf{H}}_{\mathbf{a}}}$	$\widetilde{f^{\mathbf{H}\to\mathbf{A}}_{\mathbf{a}}}$	$\widetilde{\mathcal{M}^{\mathbf{A}}_{\mathbf{h}}}, \times W^{\mathbf{H}}_{\mathbf{a}}$	$Q_{t+1} \times F_{t+1}$
$\widetilde{\mathcal{M}^{\mathbf{A}}_{\mathbf{h}}}$,	$\widetilde{f^{\mathbf{A}\to\mathbf{H}}_{\mathbf{h}}}$	$Q_t \times F_t$	$W^{\mathbf{A}}_{\mathbf{h}}$,

Table 6.3.: Concrete Mental Models

$\mathbf{H}_t = (\mathcal{M}^{\mathbf{H}}, \mathcal{M}^{\mathbf{A}}_{\mathbf{h}})$. Model $\mathcal{M}^{\mathbf{H}}$ contains a function

$$f^{\mathbf{H}\to\mathbf{A}} : \mathcal{M}^{\mathbf{A}}_{\mathbf{h}} \times W^{\mathbf{H}} \to Q_{t+1} \times F_{t+1}$$

that maps weighted words $W^{\mathbf{H}}$ generated by the human simulator in response to query Q_{t+1} and feedback F_{t+1}, and $\mathcal{M}^{\mathbf{A}}_{\mathbf{h}}$ an approximated function

$$f^{\mathbf{A}\to\mathbf{H}}_{\mathbf{h}} : Q_t \times F_t \to W^{\mathbf{A}}_{\mathbf{h}}$$

of the IR agent that maps Q_t and F_t to a set of weighted words $W^{\mathbf{A}}_{\mathbf{h}}$, possibly part of model $\widetilde{\mathcal{M}^{\mathbf{A}}}$ from the perspective of the human. An approximation of $f^{\mathbf{A}\to\mathbf{H}}_{\mathbf{h}}$ is the human's understanding of how the IR agent interprets a query and feedback. In addition, it sends feedback F_t to the agent that contains which documents returned by the agent in the previous iteration are relevant. Thus, the human agent is aware that the set of weighted words $W^{\mathbf{H}}$ in the mental model $\mathcal{M}^{\mathbf{H}}$ only subjectively represents the relevant documents; the IR agent is not necessarily able to return relevant documents from these words. In addition, it knows that the IR agent is aware of this and aims to reweight the words in Q_t with $f^{\mathbf{A}\to\mathbf{H}}_{\mathbf{h}} : Q_t \times F_t \to W^{\mathbf{A}}_{\mathbf{h}}$ that objectively represents the information need of the human agent from the perspective of the IR agent.

The IR agent needs to assign a score to all documents in the corpus and return top n of them in descending order as R_t: $\mathbf{A}_t \to R_t$ with $\mathbf{A}_t = (\widetilde{\mathcal{M}^{\mathbf{A}}}, \widetilde{\mathcal{M}^{\mathbf{H}}_{\mathbf{a}}}, \widetilde{\mathcal{M}^{\mathbf{A}}_{\mathbf{h}}})$. Model $\widetilde{\mathcal{M}^{\mathbf{A}}}$ contains the function

$$\widetilde{f^{\mathbf{A}\to\mathbf{H}}} : \widetilde{\mathcal{M}^{\mathbf{H}}_{\mathbf{a}}} \times Q_t \times F_t \to W^{\mathbf{A}}$$

where the set of words $W^{\mathbf{A}}$ objectively represents the information need of the human agent, $\widetilde{\mathcal{M}^{\mathbf{H}}_{\mathbf{a}}}$ an approximation of $\mathcal{M}^{\mathbf{H}}$ containing the function

$$\widetilde{f^{\mathbf{H}\to\mathbf{A}}_{\mathbf{a}}} : \widetilde{\mathcal{M}^{\mathbf{A}}_{\mathbf{h}}}, \times W^{\mathbf{H}}_{\mathbf{a}} \to Q_{t+1} \times F_{t+1}$$

and $\widetilde{\mathcal{M}^{\mathbf{A}}_{\mathbf{h},}}$ the function

$$\widetilde{f^{\mathbf{A}\to\mathbf{H}}_{\mathbf{h},}} : Q_t \times F_t \to W^{\mathbf{A}}_{\mathbf{h},}$$

The weighted words $W^{\mathbf{A}}$ objectively represent the information need of the human agent from the perspective of the IR agent and it compares these with the documents in the corpus by using, for instance, latent semantic indexing (LSI). Approximating $f^{\mathbf{A}\to\mathbf{H}}_{\mathbf{h}} : Q_t \times F_t \to W^{\mathbf{A}}_{\mathbf{h}}$ as $\widetilde{f^{\mathbf{A}\to\mathbf{H}}_{\mathbf{h},}} : Q_t \times F_t \to W^{\mathbf{A}}_{\mathbf{h},}$ helps the agent to act according to the expectations of the human by comparing $W^{\mathbf{A}}_{\mathbf{h},}$ with $W^{\mathbf{A}}$. If the gap between $W^{\mathbf{A}}_{\mathbf{h},}$ and $W^{\mathbf{A}}$ is too large, the IR agent could either adapt its behavior to act according to the expectations of the human even if this results in decline in retrieval performance; if the gap is even larger, it should explain its behavior. Note that $W^{\mathbf{A}}_{\mathbf{h},}$ and $W^{\mathbf{A}}$ are not necessarily identical.

Even if the human agent and the human simulator do not simulate an actual human perfectly, we argue that it is a big step towards developing an human-aware IR agent that collaboratively seeks for information together with a real human. The IR agent could be further improved in the future by evaluating it in the real world. If the IR agent performs poorly in the real world, then the human agent and human simulator need to be adapted accordingly. The effort of adapting the agents pays off if the agents are realistic enough for the evaluation of an IR agent. Evaluating an IR agent using a simulator is cheaper and significantly faster than evaluating it in the real world.

6.7. Synthetization of Relevant Document and Query Pairs

As proposed by Marchionini et al. in Marchionini and Shneiderman (1988) and taken up again by Tang et al. in Tang and G. H. Yang (2022), an information-seeking process occurs when an information seeker works with a search system in a certain domain, attempting to accomplish a task in a given setting and the process yields the foraged information as its outcome. We argue that the creation of metadata associated with documents in a repository was the task of a human, and the human accomplished the task with the help of an IR agent. For instance, a corpus of scientific publications consists of linked documents. It was the task of many humans to link them with each other. Each publication linked with a set of documents could be considered a session in which the linked documents are selected as relevant by the human simulator.

Wikipedia contains millions of articles with associated metadata and is publicly avail-

Corpus	Session Type	#Sessions	Average Articles per Session
Simple English Wikipedia	Links	104145	63.79
Simple English Wikipedia	Category	8442	84.89
Simple English Wikipedia	Path	192298	40
Simple English Wikipedia	Random	$\binom{m}{n}$	m

Table 6.4.: Simulated Session Types

able.[1] Associated metadata for each article contains a set of categories and links to other articles. The articles were written by humans who, during the writing process, searched for articles they could link with the articles they were working on. We argue that all documents linked to a randomly selected article in the corpus are relevant with respect to a session.

In addition, Wikipedia contains a hierarchy of categories with the root node "contents" and leave nodes as articles. A human might be interested in all articles that are part of a randomly selected category as they cover the same topic. To reduce the amount of documents that are relevant with respect to a session, only categories that contain no subcategories are selected.

The information need of a human might change over time during a session. This can be simulated using a Wikipedia corpus by first adding a randomly selected document to an empty list. Until a sufficient number of documents is found, a document is appended to the list which is the most similar document to the last one in the current list. Similarity is measured by computing a vector representation for each document, using LSI and then computing the cosine similarity between two document vectors. The topics of the added documents slightly change along the list.

Another session could be simulated by randomly selecting documents from the corpus. Randomly selected documents reflect a human potentially having a high-level goal that the IR agent is unable to anticipate from a session.

An overview of creatable sessions from the Simple English Wikipedia corpus is listed in Table 6.4.

[1] `https://dumps.wikimedia.org/`

7. Conclusion and Outlook

Despite domain independent suggestions as creating a data management plan (DMP) or following guidelines such as FAIR, research data is an ever growing complex mesh of data. Researchers spend a lot of time on deriving results from it. Derived results are by themselves research data. The large amount of time required to derive results is caused by, among others, the need to search for information, to clean and reformat data, and to link results with research data from which they were derived. In most research fields researchers have found a way to deal with such a complex mesh of data, however derived results should be reused which is again not easy. A lot of time is already spend on deriving results and on representing them such that others can follow them. Then, no time is left for preparing the results to be reusable. Producing reusable results is a laborious process, time and cost intensive and mostly impossible without the help of an IT-expert. Additionally, it is often not obvious why the effort is worthwhile.

In this thesis we showed that such complex problems have simple solutions: Results can be structured by a parser generated from a manually written grammar, transformed into almost any format, linked or combined with each other and visualized on the web such that others can precisely refer to it or make annotations. In addition, we propose a human-aware IR-agent that can search for information in a mesh of data by collaborating with a human. In the following, we summarize our contributions in Section 7.1 and finally provide an outlook for further research directions in Section 7.2.

7.1. Summary of Contributions

In this thesis we identified problems and implemented solutions for researchers aiming at publishing results being valid with respect to the FAIR principles. Those who reuse the data have more time to spend on actual research rather than to search, clean, reformat and link data to be reused. We summarize our solutions from this thesis as follows:

(i) Multi-target publishing Published results are more likely to be reused by others if they are available in different formats. However, publishing results requires either to restrict them syntactically to be supported by, e.g., Pandoc or to use custom

implementations to produce and transform results into various formats. We have shown in Chapter 3 that researchers can keep their preferred tools and document formats in use to produce results that can be transformed into various formats. Changes in the transformation process can be easily adjusted. New opportunities, such as linking and combining, viewing, annotating and retrieving research data are offered as transformed data is structured such that machines can interpret them.

(ii) Linking and combining data We have shown in Chapter 4 that research data often contains, e.g., facts, individuals or in general data points which are related to each other. Given a data point, researchers are interested in related data providing more information. Searching for such related information can be laborious and time intensive as related information is either not linked at all or only implicitly by a description written in natural language. A solution is to either make implicit links explicit such that humans and machines can follow them or to combine related data with each other. Combining large word indices as presented in Section 4.1 or combining medical data as presented in Section 4.2 manually can take several weeks or even month, is laborious and time-intensive and is possibly error-prone. This is just as true for linking research data, where explicit links are manually created from implicit links, as shown in Section 4.3. We showed that the high expenditure of time can be reduced to a few minutes or even seconds. Additionally, in case of detected errors, the way of how data is combined and linked can be changed easily and then automatically repeated. If data is combined and linked manually, one might have to start completely over again, in case of errors are detected.

(iii) Viewer for machine readable research data Research data stored at a repository, formatted to be reusable, containing only explicit links such that machines can follow them and associated with metadata such that the data is indexed by an IR system is a huge step towards obtaining reusable data. However, a researcher still needs to decide whether found, possibly very large, data at the repository is relevant. Decision making requires to download the whole dataset and in case of the data is formatted to be machine readable the data needs to be visualized to be human readable. We have shown in Section 4.4 that existing methods for visualizing XML data is not always loss free, even if encoded data is valid with respect to a domain specific XSD schema. Domain independent alternatives such as Pandoc restrict researchers to a specific data format. Resulting data is automatically transformable into machine and human readable formats, however due to the domain independence many desired visualization styles are not supported. We show that a viewer for viewing specific TEI documents, satisfying the needs of the author using JavaScript is easy to implement, as the to be visualized data is

well structured. Humans can use the viewer to follow explicitly available links to related data visualized by our viewer.

(iv) Publishing collections of research data If found data stored at a repository is relevant and used to produce results, then one needs to be able to refer to the data such that others can verify the results. However, data stored at a repository are only unambiguously identifiable as a whole via its unique identifier part of the metadata. We have shown in Chapter 5 of how to point explicitly to a subset of data by adding share buttons at our viewer. Each share button allows to point to a specific region in the metadata that can be added to a collection. Additionally, as it is often the case when humanities scholars transcribe texts from pictures, one would like to refer precisely to specific regions of pictures. Hence, we have shown in Section 5.2 of how to annotate regions of pictures.

(v) Human-aware information retrieval Humans spend a lot of time on deriving results from a complex and large mesh of research data. The high expenditure of time is not only caused by data not respecting to the FAIR guidelines. Additional time is needed to find information in the first place before it is used to derive new results from it. The problem does not get smaller even if data is associated with metadata indexable by an IR system. Humans can reach sometimes only a specific goal if they have access to a specific information. Expressing this information need can be challenging as one needs to express it in a language an IR system can understand and IR systems are usually unaware of the human. Hence, we propose in Chapter 6 a human-aware IR agent that can collaboratively search for information with a human. In addition to the content of a repository, subjective content descriptions (SCDs) are also used as additional information to calculate results. SCDs are either associated with content by using our annotation system or extracted by identifying inline subjective content descriptions (iSCDs). Yet, we have not found an existing evaluation methodology for our human-aware IR agent. Hence we propose a new evaluation methodology that could be used and argue that it should be implemented by using only publicly available datasets such as a Wikipedia corpus.

7.2. Outlook

We have shown that supporting researchers to prepare data to make it reusable helps to partially prevent a mesh of data in the long run. Possible future research directions are:

Information System We presented in multi-target publishing (Chapter 3) and linking & combining research data (Chapter 4) of how to obtain an interactive viewer for

viewing research data. Additionally, our viewer allow to annotate images and to create collections of unique identifiers. Annotations and collections can be used by our human-aware IR agent to improve retrieval results. For future work, all our solutions need to be assembled together such that if one persists her or his data, then one gets an information system almost for free. The goal of an information system is not only to archive research data, but also to support decision making based on that data (Zwass, 2023). We suggest that archived data is formatted to be interpretable by machines and zipped as an archive, associated with a METS metadata file, and executable code is linked with the data it visualizes via the METS file. The creation of a METS file could be supported with an application. Executable code could be reused, such as code for visualizing a generic CSV file. If more sophisticated viewers are demanded, then one needs the help of an IT specialist. However, the IT specialist only needs to implement a viewer for machine interpretable data. Finally, one could modify an existing RDR implementation such as Zenodo by adding an "information system button". If a user clicks on that "information system button", an information system is created on demand with archived data at the RDR as a source. The information system could be equipped with our viewer, annotation system, collections of unique identifiers and a human-aware information retrieval agent.

Adapt data management from other domains One could adapt solutions from platforms whose primary purpose is to share source code with others, like GitHub (GitHub, 2023) or GitLab (GitLab, 2023). Such platforms allow to fork a whole repository to make a pull request. For each request, a forum can be used for discussions and the code can be automatically executed for tests. Research data is not executable code, however for each request, existing AI models based on uploaded data could be retrained and check whether there is an improvement. An improvement is an indication for an improvement of the data quality. Security alerts are optionally available if source code uses a library that is found to be insecure. In case of research data, one could receive an alert if data is linked with sources that are unethical or have a new license that forbids the reuse. Additionally, GitHub and OpenAI launched Copilot, an "AI pair programmer" (Nguyen and Nadi, 2022). Such a Copilot (i.e., agent) could support researchers producing new results based on uploaded research data at a repository by, for instance, searching for information, given the context a researcher is currently working on. Those who allow an agent to access their data to support others are more likely to be cited.

A. Appendix

We present in the appendix in Listing 18 a Python program for transforming a Critical Edition written in Microsoft Word into an intermediate representation at where everything that is semantically different is syntactically differentiable. Followed by an Antlr4 lexer in Listing 19 and Antlr4 grammar in Listing 20 for parsing the intermediate representation of a critical edition. Finally, we present the schema of a PostgreSQL database for critical editions in Listing 21.

A.1. Extracting and Transforming Microsoft Word DOCX Documents

```
1  import os
2  import re
3  import zipfile
4
5  from hashlib import sha256
6
7  import xml.etree.ElementTree as ET
8
9  class Paragraph:
10   def __init__(self):
11     self.chapter = None
12     self.subchapter = None
13     self.subsubchapter = None
14
15 class Row:
16   def __init__(self):
17     self.bold = None
18     self.color = None
19     self.highlight = None
20     self.strike = None
21
22 class DocumentReader:
23   def __init__(self, path):
24     self.path = path
```

```
25
26      word_document = zipfile.ZipFile(self.path, 'r')
27      document_xml = word_document.open('word/document.xml')
28      self.document = self.parse_word(ET.fromstring(document_xml.read()
        ↪  .decode('utf-8')))
29      self.name = os.path.basename(str(self.path))
30      self.id = sha256(self.document.encode('utf-8')).hexdigest()
31
32    def extract_namespace(self, root):
33      if hasattr(root, 'tag'):
34        matcher = re.match(r'\{.*\}', root.tag)
35        namespace = matcher.group(0)
36        tag = root.tag.replace(namespace, '')
37        return namespace, tag
38
39    def parse_word(self, root):
40      for child in list(root):
41        namespace, tag = self.extract_namespace(child)
42        if tag == 'body':
43          return self.parse_body(child)
44
45    def parse_body(self, root):
46      paragraphs = list()
47
48      for child in list(root):
49        namespace, tag = self.extract_namespace(child)
50        if tag == 'p':
51          paragraphs.append(self.parse_p(child))
52        elif tag == 'tbl':
53          paragraphs.append('<!TABLE!>' + self.parse_tbl(child) +
          ↪  '<!TABLE!>')
54        elif tag == 'sectPr':
55          continue
56        elif tag == 'bookmarkEnd':
57          continue
58        else:
59          print('parse_body', tag)
60
61      return '\n'.join(paragraphs)
62
63    def parse_p(self, root):
64      result = list()
65
66      paragraph = None
67      for child in root:
```

```
68        namespace, tag = self.extract_namespace(child)
69        if tag == 'pPr':
70          paragraph = self.parse_pPr(child)
71
72      for child in root:
73        namespace, tag = self.extract_namespace(child)
74        if tag == 'r':
75          result += self.parse_r(child)
76
77      if paragraph is not None and paragraph.chapter:
78        result = ['<!CHAPTER!>'] + result
79      elif paragraph is not None and paragraph.subchapter:
80        result = ['<!SUBCHAPTER!>'] + result
81      elif paragraph is not None and paragraph.subchapter:
82        result = ['<!SUBSUBCHAPTER!>'] + result
83
84      result = ''.join(result)
85      return result if result.strip() else ''
86
87    def parse_tbl(self, root):
88      result = list()
89
90      for child in root:
91        namespace, tag = self.extract_namespace(child)
92        if tag == 'tr':
93          result += ['<!ROW!>'] + [self.parse_tr(child)] + ['<!ROW!>\n']
94        elif tag == 'tblPr':
95          continue
96        elif tag == 'tblGrid':
97          continue
98        else:
99          print('parse_tbl', child)
100
101     result = ''.join(result)
102     return result if result.strip() else ''
103
104   def parse_tr(self, root):
105     result = list()
106
107     for child in root:
108       namespace, tag = self.extract_namespace(child)
109       if tag == 'tc':
110         result += ['<!COLUMN!>'] + [self.parse_tc(child)] +
          ↪  ['<!COLUMN!>']
111       else:
```

```
112             print('parse_tr', child)
113
114         result = ''.join(result)
115         return result if result.strip() else ''
116
117     def parse_tc(self, root):
118         result = list()
119
120         for child in root:
121             namespace, tag = self.extract_namespace(child)
122             if tag == 'tcPr':
123                 continue
124             elif tag == 'p':
125                 result += self.parse_p(child) + '\n'
126             else:
127                 print('parse_tc', tag)
128
129         result = ''.join(result)
130         return result if result.strip() else ''
131
132     def parse_pPr(self, root):
133         paragraph = Paragraph()
134
135         for child in root:
136             namespace, tag = self.extract_namespace(child)
137             if tag == 'pStyle':
138                 for attribute in child.attrib:
139                     if child.attrib[attribute] == 'Titel':
140                         paragraph.chapter = True
141                     elif child.attrib[attribute] == 'Title':
142                         paragraph.chapter = True
143                     elif child.attrib[attribute] == 'berschrift1':
144                         paragraph.chapter = True
145                     elif child.attrib[attribute] == 'berschrift2':
146                         paragraph.subchapter = True
147                     elif child.attrib[attribute] == 'Heading1':
148                         paragraph.chapter = True
149                     elif child.attrib[attribute] == 'Heading2':
150                         paragraph.subchapter = True
151                     elif child.attrib[attribute] == 'Heading3':
152                         paragraph.subsubchapter = True
153                     elif child.attrib[attribute] == 'Textkrper-Zeileneinzug':
154                         continue
155                     elif child.attrib[attribute] == 'NoSpacing':
156                         continue
```

```
157             elif child.attrib[attribute] == 'NurText':
158               continue
159             elif child.attrib[attribute] == 'Textbody':
160               continue
161             elif child.attrib[attribute] == 'KeinLeeraum':
162               continue
163             elif child.attrib[attribute] == 'NormalWeb':
164               continue
165             elif child.attrib[attribute] == 'sdfootnote':
166               continue
167             elif child.attrib[attribute] == 'StandardWeb':
168               continue
169             elif child.attrib[attribute] == 'PlainText':
170               continue
171             elif child.attrib[attribute] == 'BodyTextIndent':
172               continue
173             else:
174               print('parse_pPr', child.attrib[attribute])
175         elif tag == 'outlineLvl':
176           for attribute in child.attrib:
177             if child.attrib[attribute] == '1':
178               paragraph.subchapter = True
179             else:
180               print('parse_pPr', child.attrib[attribute])
181
182       return paragraph
183
184     def parse_r(self, root):
185       rows = list()
186
187       for child in root:
188         namespace, tag = self.extract_namespace(child)
189         if tag == 't':
190           rows.append(child.text)
191         elif tag == 'tab':
192           rows.append('\t')
193         elif tag == 'rPr':
194           continue
195         elif tag == 'lastRenderedPageBreak':
196           continue
197         elif tag == 'br':
198           continue
199         elif tag == 'footnoteReference':
200           for attribute in child.attrib:
201             rows += '<!FOOT!>' + child.attrib[attribute] + '<!FOOT!>'
```

```
202         elif tag == 'softHyphen':
203           continue
204         elif tag == 'sym':
205           continue
206         elif tag == 'bookmarkEnd':
207           continue
208         else:
209           print('parse_r', tag)
210 
211       for child in root:
212         namespace, tag = self.extract_namespace(child)
213         if tag == 'rPr':
214           row = self.parse_rPr(child)
215           if row.bold:
216             rows = ['<b>'] + rows + ['</b>']
217           elif not row.color is None and not row.color == '000000' and
              ↪  not row.color == 'auto':
218             rows = ['<span style=\"color:#'] + [row.color] + [';\">'] +
                ↪  rows + ['</span>']
219           elif not row.highlight is None:
220             rows = ['<span style=\"background-color:'] + [row.highlight]
                ↪  + [';\">'] + rows + ['</span>']
221           elif not row.strike is None:
222             rows = ['<strike>'] + rows + ['</strike>']
223 
224       return rows
225 
226     def parse_rPr(self, root):
227       row = Row()
228 
229       for child in root:
230         namespace, tag = self.extract_namespace(child)
231         if tag == 'color':
232           for attribute in child.attrib:
233             row.color = child.attrib[attribute]
234         elif tag == 'b':
235           row.bold = True
236         elif tag == 'highlight':
237           for attribute in child.attrib:
238             row.highlight = child.attrib[attribute]
239         elif tag == 'strike':
240           for attribute in child.attrib:
241             row.strike = child.attrib[attribute]
242         elif tag == 'rFonts':
243           continue
```

```
244         elif tag == 'lang':
245           continue
246         elif tag == 'sz':
247           continue
248         elif tag == 'szCs':
249           continue
250         elif tag == 'iCs':
251           continue
252         elif tag == 'i':
253           continue
254         elif tag == 'vertAlign':
255           continue
256         elif tag == 'cs':
257           continue
258         elif tag == 'rtl':
259           continue
260         elif tag == 'bCs':
261           continue
262         elif tag == 'rStyle':
263           continue
264         elif tag == 'u':
265           continue
266         elif tag == 'kern':
267           continue
268         elif tag == 'smallCaps':
269           continue
270         elif tag == 'position':
271           continue
272         elif tag == 'spacing':
273           continue
274         elif tag == 'caps':
275           continue
276         else:
277           print('parse_rPr', tag)
278 
279       return row
280 
```

Listing 18: Python3 source code for extracting and transforming Microsoft DOCX documents

A.2. Antlr4 Lexer Grammar for DOCX Documents of Type "Critical"

```
lexer grammar CriticalLexer;

CHAPTER
  :  '<!CHAPTER!>'
  ;

SUBCHAPTER
    :   '<!SUBCHAPTER!>'
    ;

FOOT
  :  '<!FOOT!>'
  ;

HTML
  :  '</span>'
  |  '<span style="color:#' ('0'..'9' | 'A'..'F')+ ';">'
  |  '<span style="background-color:' ('a'..'z' | 'A'..'Z')+ ';">'
  |  '<strike>'
  |  '</strike>'
  |  '<b>'
  |  '</b>'
  ;

NUMERIC
  :  ('0'..'9')+
  ;

ASCII
  :
  (  'a'..'z'
  |  'A'..'Z'
  )+
  ;

BULLET
  :  '•'
  ;

SPECIAL
  :  ':' | ',' | '-' | '[' | ']' | '.' | '*' | '(' | ')' | ';'
```

```
    |  '/' | '+' | '=' | '?' | '$' | '#' | '{' | '}' | '|' | '>'
    |  '_' | '~' | '’' | '–' | '\u{0190}' | '‡' | '‘' | '“' | '”' | '&'
    |  '!' | '-' | '…' | '"' | '†' | '╪' | '∞' | '\u{200D}'
    ;

NEWLINE
    :  '\n'
    ;

TAB
    :  '\t'
    ;

SPACE
    :  ' '
    ;

TAMIL_WORD
    :  TAMIL_RANGE+
    ;

LATIN_WORD
    :  LATIN_RANGE+
    ;

TAMIL_RANGE
    :  '\u{0B80}'..'\u{0BFF}'
    ;

LATIN_RANGE
    :  '\u{0080}'..'\u{00FF}'
    |  '\u{0100}'..'\u{017F}'
    |  '\u{1E00}'..'\u{1EFF}'
    ;
```

Listing 19: Antlr4 lexer grammar for DOCX documents of type critical

A.3. Antlr4 Parser Grammar for DOCX Documents of Type “Critical”

```
parser grammar CriticalParser;

```

```
options {
  tokenVocab = CriticalLexer;
}

@parser::header {
class Critical:
  def __init__(self):
    self.translation_data = list()
    self.testimonia = None

class TranslationData:
  def __init__(self):
    self.title = None
    self.kilavi_tamil = None
    self.tamil = list()
    self.apparatus = list()
    self.transcription = list()
    self.kilavi_english = list()
    self.first_translation = list()
    self.second_translation = list()

class Testimonia:
  def __init__(self):
    self.key = None
    self.value = None
}

translation returns [item]
  @init {
$item = Critical()
  }
  :  (testimonia_chapter
  {
$item.testimonia = $testimonia_chapter.items
  }
  | translation_chapter
  {
$item.translation_data.append($translation_chapter.item)
  }
  )+
  ;

testimonia_chapter returns [items]
  @init{
$items = list()
```

```
48    }
49    :  CHAPTER? text NEWLINE+ text NEWLINE (testimonia_line
50    {
51  $items.append($testimonia_line.item)
52    }
53    )+ text NEWLINE (testimonia_line
54    {
55  $items.append($testimonia_line.item)
56    }
57    )+
58    ;
59
60  testimonia_line returns [item]
61    @init {
62  $item = Testimonia()
63    }
64    :  k=text
65    {
66  $item.key = $k.text
67    }
68    TAB v=text
69    {
70  $item.value = $v.text
71    }
72    NEWLINE (TAB v=text
73    {
74  $item.value += $v.text
75    }
76    NEWLINE)?
77    ;
78
79  translation_chapter returns [item]
80    @init {
81  $item = TranslationData()
82    }
83    :  (CHAPTER | SUBCHAPTER) title=text
84    {
85  $item.title = $title.item
86    }
87    NEWLINE (kilavi_tamil=text
88    {
89  if $item.kilavi_tamil is None:
90    $item.kilavi_tamil = $kilavi_tamil.item
91  else:
92    $item.kilavi_tamil += $kilavi_tamil.item
```

```
93    }
94    NEWLINE)+ NEWLINE tamil_block
95    {
96  $item.tamil = $tamil_block.items
97    }
98    NEWLINE (apparatus_block
99    {
100 $item.apparatus = $apparatus_block.items
101   }
102   NEWLINE)* transcription_block
103   {
104 $item.transcription = $transcription_block.items
105   }
106   (NEWLINE+ first_translation
107   {
108 $item.first_translation = $first_translation.items
109   }
110   )? (NEWLINE+ second_translation
111   {
112 $item.second_translation = $second_translation.items
113   }
114   )? NEWLINE*
115   ;
116
117 tamil_block returns [items]
118   @init {
119 $items = list()
120   }
121   :  (tamil
122   {
123 $items.append($tamil.item)
124   }
125   NEWLINE | TAB e=text
126   {
127 $items.append($e.item)
128   }
129   NEWLINE+)+
130   ;
131
132 tamil returns [item]
133   @init {
134 $item = ''
135   }
136   :
137   (  NUMERIC
```

```
138     {
139   $item += $NUMERIC.text
140     }
141     |  SPECIAL
142     {
143   $item += $SPECIAL.text
144     }
145     |  TAB
146     {
147   $item += $TAB.text
148     }
149     |  SPACE
150     {
151   $item += $SPACE.text
152     }
153     |  TAMIL_WORD
154     {
155   $item += $TAMIL_WORD.text
156     }
157     |  HTML
158     {
159   $item += $HTML.text
160     }
161     )+
162     ;
163
164   apparatus_block returns [items]
165     @init {
166   $items = list()
167     }
168     :  (left=text
169     {
170   $items.append($left.item)
171     }
172     NEWLINE)* apparatus
173     {
174   $items.append($apparatus.item)
175     }
176     NEWLINE (right=text
177     {
178   $items.append($right.item)
179     }
180     NEWLINE | apparatus
181     {
182   $items.append($apparatus.item)
```

```
183     }
184     NEWLINE)*
185     ;
186
187   apparatus returns [item]
188     @init {
189   $item = ''
190     }
191     :  left=text?
192     {
193   if not $left.text is None:
194     $item += $left.item
195     }
196     BULLET
197     {
198   $item += $BULLET.text
199     }
200     right=text
201     {
202   $item += $right.text
203     }
204     ;
205
206   transcription_block returns [items]
207     @init {
208   $items = list()
209     }
210     :  (transcription
211     {
212   $items.append($transcription.item)
213     }
214     NEWLINE)+
215     ;
216
217   transcription returns [item]
218     @init {
219   $item = '';
220     }
221     :
222     (  NUMERIC
223     {
224   $item += $NUMERIC.text
225     }
226     |  ASCII
227     {
```

```
228 $item += $ASCII.text
229   }
230   |  SPECIAL
231   {
232 $item += $SPECIAL.text
233   }
234   |  TAB
235   {
236 $item += $TAB.text
237   }
238   |  SPACE
239   {
240 $item += $SPACE.text
241   }
242   |  LATIN_WORD
243   {
244 $item += $LATIN_WORD.text
245   }
246   |  FOOT
247   {
248 $item += $FOOT.text
249   }
250   |  HTML
251   {
252 $item += $HTML.text
253   }
254   )+
255   ;
256
257 kilavi_english returns [items]
258   @init {
259 $items = list()
260   }
261   :  (translation_text
262   {
263 $items.append($translation_text.item)
264   }
265   NEWLINE)+
266   ;
267
268 first_translation returns [items]
269   @init {
270 $items = list()
271   }
272   :  (translation_text
```

```
273     {
274   $items.append($translation_text.item)
275     }
276     NEWLINE)+
277     ;
278
279   second_translation returns [items]
280     @init {
281   $items = list()
282     }
283     :  (translation_text
284     {
285   $items.append($translation_text.item)
286     }
287     NEWLINE+)+
288     ;
289
290   translation_text returns [String item]
291     @init {
292   $item = '';
293     }
294     :
295     (  NUMERIC
296     {
297   $item += $NUMERIC.text
298     }
299     |  ASCII
300     {
301   $item += $ASCII.text
302     }
303     |  SPECIAL
304     {
305   $item += $SPECIAL.text
306     }
307     |  TAB
308     {
309   $item += $TAB.text
310     }
311     |  SPACE
312     {
313   $item += $SPACE.text
314     }
315     |  TAMIL_WORD
316     {
317   $item += $TAMIL_WORD.text
```

```
318     }
319     |  LATIN_WORD
320     {
321   $item += $LATIN_WORD.text
322     }
323     |  FOOT
324     {
325   $item += $FOOT.text
326     }
327     |  HTML
328     {
329   $item += $HTML.text
330     }
331     )+
332     ;
333
334   text returns [item]
335     @init {
336   $item = ''
337     }
338     :
339     (  NUMERIC
340     {
341   $item += $NUMERIC.text
342     }
343     |  ASCII
344     {
345   $item += $ASCII.text
346     }
347     |  SPECIAL
348     {
349   $item += $SPECIAL.text
350     }
351     |  SPACE
352     {
353   $item += $SPACE.text
354     }
355     |  TAMIL_WORD
356     {
357   $item += $TAMIL_WORD.text
358     }
359     |  LATIN_WORD
360     {
361   $item += $LATIN_WORD.text
362     }
```

```
363     |  FOOT
364     {
365   $item += $FOOT.text
366     }
367     |  HTML
368     {
369   $item += $HTML.text
370     }
371     )+
372     ;
```

Listing 20: Antlr4 parser grammar for DOCX documents of type critical

A.4. Schema of a PostgreSQL Database Containing a Critical Edition

```
1  DROP TABLE IF EXISTS parser_document CASCADE;
2  CREATE TABLE parser_document (
3    id text primary key,
4    docfile text,
5    docname text,
6    content text,
7    doctype text,
8    docformat text,
9    author text,
10   compiled_by text
11 );
12
13 DROP TABLE IF EXISTS parser_critical CASCADE;
14 CREATE TABLE parser_critical (
15   document_id text references parser_document(id),
16   id serial primary key,
17   title text,
18   kilavi_tamil text,
19   tamil text[],
20   apparatus text[],
21   transcription text[],
22   kilavi_english text[],
23   first_translation text[],
24   second_translation text[]
25 );
26
```

```
27 DROP TABLE IF EXISTS parser_testimonia CASCADE;
28 CREATE TABLE parser_testimonia (
29   document_id text references parser_document(id),
30   id serial primary key,
31   key text,
32   value text
33 );
34
35 CREATE OR REPLACE FUNCTION export_critical_as_tei(id int) RETURNS
   ↪  SETOF xml AS $$
36 BEGIN
37   RETURN QUERY EXECUTE '
38     SELECT xmlroot(
39       xmlelement(NAME "TEI",
40         XMLATTRIBUTES (
41           E''http://www.tei-c.org/ns/1.0'' AS xmlns
42         ),
43         xmlconcat(
44           xmlelement(NAME "teiHeader",
45             xmlelement(NAME "fileDesc",
46               xmlconcat(
47                 xmlelement(NAME "titleStmt",
48                   xmlconcat(
49                     xmlelement(NAME title,
50                       (
51                         SELECT
52                           title
53                         FROM
54                           parser_critical
55                         WHERE
56                           id = ''' || id || '''
57                       )
58                     ),
59                     xmlelement(NAME "author",
60                       (
61                         SELECT
62                           author
63                         FROM
64                           parser_document
65                         WHERE id = (
66                           SELECT
67                             document_id
68                           FROM
69                             parser_critical
70                           WHERE
```

```
                        id = ''' || id || '''
                    )
                  )
                ),
                xmlelement(NAME "respStmt",
                  xmlconcat(
                    xmlelement(NAME "resp",
                      ''compiled by''
                    ),
                    xmlelement(NAME "name",
                      (
                        SELECT
                          compiled_by
                        FROM
                          parser_document
                        WHERE id = (
                          SELECT
                            document_id
                          FROM
                            parser_critical
                          WHERE
                            id = ''' || id || '''
                        )
                      )
                    )
                  )
                )
              ),
              xmlelement(NAME "publicationStmt",
                xmlelement(NAME p, '''')
              ),
              xmlelement(NAME "sourceDesc",
                (
                  SELECT
                    xmlelement(NAME "listWit",
                      xmlagg(
                        xmlelement(NAME witness,
                          XMLATTRIBUTES(
                            replace(key, ''*'', '''') AS "xml:id"
                          ),
                          value
                        )
                      )
                    ) FROM
```

```
116                       parser_testimonia
117                     WHERE document_id = (
118                       SELECT
119                         document_id
120                       FROM
121                         parser_critical
122                       WHERE
123                         id = ''' || id || '''
124                   )
125                 )
126               )
127             )
128           )
129         ),
130         xmlelement(NAME text,
131           xmlelement(NAME body,
132             xmlconcat(
133               (
134                 SELECT
135                   xmlelement(NAME div,
136                     XMLATTRIBUTES(
137                       title AS n,
138                       ''kilavi_tamil'' AS type,
139                       ''ta'' AS "xml:lang"
140                     ),
141                     xmlconcat(
142                       xmlelement(NAME head,
143                         ''kiLavi Tamil''
144                       ),
145                       xmlelement(NAME l,
146                         XMLATTRIBUTES (
147                           ''1'' AS n
148                         ),
149                         trim(
150                           replace(kilavi_tamil, ''<!BOLD!>'', '''')
151                         )
152                       )
153                     )
154                   ) FROM (
155                     SELECT
156                       replace(substring(replace(title, ''<!BOLD!>'',
                            ↪  ''''), ''AN [0-9]+/[0-9]+|KV/[0-9]+''), ''
                            ↪  '', ''_'') AS title,
157                       kilavi_tamil
158                     FROM
```

```
159                         parser_critical
160                       WHERE
161                         id = ' || quote_literal(id) || '
162                     ) sub_khilavi
163                   ),
164                   (
165                     SELECT
166                       xmlelement(NAME div,
167                         XMLATTRIBUTES(
168                           title As n,
169                           ''tamil'' AS type,
170                           ''ta'' AS "xml:lang"
171                         ),
172                         xmlconcat(
173                           xmlelement(NAME head,
174                             ''Tamil Poem''
175                           ),
176                           xmlagg(
177                             xmlelement(NAME l,
178                               XMLATTRIBUTES (
179                                 index AS n
180                               ),
181                               trim(
182                                 replace(line, ''<!BOLD!>'', '''')
183                               )
184                             )
185                           )
186                         )
187                       ) FROM (
188                         SELECT
189                           replace(substring(replace(title, ''<!BOLD!>'',
                               ↪  ''''), ''AN [0-9]+/[0-9]+|KV/[0-9]+''), ''
                               ↪  '', ''_'') AS title,
190                           kilavi_tamil,
191                           trim(line) AS line,
192                           index
193                         FROM
194                           parser_critical,
195                           unnest(tamil) WITH ORDINALITY x(line, index)
196                         WHERE
197                           id = ' || quote_literal(id) || '
198                         ORDER BY
199                           index
200                       ) sub_tamil
201                     GROUP BY
```

```
202                     title,
203                     kilavi_tamil
204                 ),
205                 (
206                   SELECT
207                     xmlelement(NAME div,
208                       XMLATTRIBUTES(
209                         ''apparatus'' AS type
210                       ),
211                       xmlconcat(
212                         xmlelement(NAME head,
213                           ''Apparatus''
214                         ),
215                         xmlagg(
216                           xmlelement(NAME l,
217                             XMLATTRIBUTES (
218                               index AS n
219                             ),
220                             trim(
221                               replace(line, ''<!BOLD!>'', '''')
222                             )
223                           )
224                         )
225                       )
226                     ) FROM (
227                       SELECT
228                         kilavi_tamil,
229                         trim(line) AS line,
230                         index
231                       FROM
232                         parser_critical,
233                         unnest(apparatus) WITH ORDINALITY x(line,
                            ↪  index)
234                       WHERE
235                         id = ' || quote_literal(id) || '
236                       ORDER BY
237                         index
238                     ) sub_apparatus
239                   GROUP BY
240                     title,
241                     kilavi_tamil
242                 ),
243                 (
244                   SELECT
245                     xmlelement(NAME div,
```

```
246                       XMLATTRIBUTES(
247                         ''transliteration'' AS type,
248                         ''ta'' AS "xml:lang"
249                       ),
250                       xmlconcat(
251                         xmlelement(NAME head,
252                           ''Transliteration''
253                         ),
254                         xmlagg(
255                           xmlelement(NAME l,
256                             XMLATTRIBUTES (
257                               index AS n
258                             ),
259                             trim(
260                               replace(line, ''<!BOLD!>'', '''')
261                             )
262                           )
263                         )
264                       )
265                     ) FROM (
266                       SELECT
267                         kilavi_tamil,
268                         trim(line) AS line,
269                         index
270                       FROM
271                         parser_critical,
272                         unnest(transcription) WITH ORDINALITY x(line,
                            ↪  index)
273                       WHERE
274                         id = ' || quote_literal(id) || '
275                       ORDER BY
276                         index
277                     ) sub_transcription
278                   GROUP BY
279                     title,
280                     kilavi_tamil
281                 ),
282                 (
283                   SELECT
284                     xmlelement(NAME div,
285                       XMLATTRIBUTES(
286                         ''kilavi_english'' AS type,
287                         ''en'' AS "xml:lang"
288                       ),
289                       xmlconcat(
```

```
290                       xmlelement(NAME head,
291                         ''kiLavi English''
292                       ),
293                       xmlagg(
294                         xmlelement(NAME l,
295                           XMLATTRIBUTES (
296                             index AS n
297                           ),
298                           trim(
299                             replace(line, ''<!BOLD!>'', '''')
300                           )
301                         )
302                       )
303                     )
304                   ) FROM (
305                     SELECT
306                       kilavi_tamil,
307                       trim(line) AS line,
308                       index
309                     FROM
310                       parser_critical,
311                       unnest(kilavi_english) WITH ORDINALITY x(line,
                            ↪  index)
312                     WHERE
313                       id = ' || quote_literal(id) || '
314                     ORDER BY
315                       index
316                   ) sub_kilavi_english
317                 GROUP BY
318                   title,
319                   kilavi_tamil
320               ),
321               (
322                 SELECT
323                   xmlelement(NAME div,
324                     XMLATTRIBUTES(
325                       ''word_by_word_translation'' AS type,
326                       ''en'' AS "xml:lang"
327                     ),
328                     xmlconcat(
329                       xmlelement(NAME head,
330                         ''Word-by-word translation into English''
331                       ),
332                       xmlagg(
333                         xmlelement(NAME l,
```

```
                                XMLATTRIBUTES (
                                  index AS n
                                ),
                                trim(
                                  replace(line, ''<!BOLD!>'', '''')
                                )
                              )
                            )
                          ) FROM (
                            SELECT
                              kilavi_tamil,
                              trim(line) AS line,
                              index
                            FROM
                              parser_critical,
                              unnest(first_translation) WITH ORDINALITY
                                ↪  x(line, index)
                            WHERE
                              id = ' || quote_literal(id) || '
                            ORDER BY
                              index
                          ) sub_first_translation
                        GROUP BY
                          title,
                          kilavi_tamil
                      ),
                      (
                        SELECT
                          xmlelement(NAME div,
                            XMLATTRIBUTES(
                              ''translation'' AS type,
                              ''en'' AS "xml:lang"
                            ),
                            xmlconcat(
                              xmlelement(NAME head,
                                ''Translation into English''
                              ),
                              xmlagg(
                                xmlelement(NAME l,
                                  XMLATTRIBUTES (
                                    index AS n
                                  ),
                                  trim(
                                    replace(line, ''<!BOLD!>'', '''')
```

```
                                )
                              )
                            )
                          )
                        ) FROM (
                          SELECT
                            kilavi_tamil,
                            trim(line) AS line,
                            index
                          FROM
                            parser_critical,
                            unnest(second_translation) WITH ORDINALITY
                             ↪  x(line, index)
                          WHERE
                            id = ' || quote_literal(id) || '
                          ORDER BY
                            index
                        ) sub_second_translation
                      GROUP BY
                        title,
                        kilavi_tamil
                    )
                  )
                )
              )
            )
        ), version ''1.0'', standalone yes
      ) FROM (
        SELECT
          substring(replace(title, ''<!BOLD!>'', ''''), ''AN
           ↪  [0-9]+/[0-9]+|KV/[0-9]+'') AS title
        FROM
          parser_critical
        WHERE
          id = ' || quote_literal(id) || '
      ) sub_a';
END
$$ LANGUAGE plpgsql;

DROP VIEW IF EXISTS parser_tei CASCADE;
CREATE VIEW parser_tei AS SELECT
  document_id,
  id,
  export_critical_as_tei(id) AS tei
FROM
```

```
421     parser_critical;
```

Listing 21: Schema of a PostgreSQL database containing a critical edition

List of Figures

List of Tables

List of Listings

Acronyms

AI artificial intelligence. 6, 7, 93, 122

API application programming interface. 7, 28, 47, 50, 83, 155

BNF Backus–Naur form. 43

C-CDA Consolidated Clinical Document Architecture. 68

CARE **C**ollective benefit, **A**uthority to Control, **R**esponsibility, and **E**thics. 3, 7

CD content description. 13, 105

CERN European Organization for Nuclear Research. 22

CHAI humanities-centred artificial intelligence. 6

CLEF Conference and Labs of the Evaluation Forum. 11

CSMC Center for the Study of Manuscript Cultures. 6

CSS Cascading Style Sheets. 84, 85, 152

CSV comma-separated values. 9, 23, 24, 27, 68, 122

CT cube test. 114

DC dublin core. 20

DDI data documentation initiative. 20

DEDR Dravidian etymological dictionary. 54, 58, 61, 83, 88, 89, 152, 155

DFA deterministic finite automaton. 44

DM data mining. 1

DMP data management plan. iii, 1, 119

DOI digital object identifier. 2, 8, 13, 22, 23, 83, 91, 92

DS dynamic search. 97

EHR electronic health record. 64–69

EpiDoc Epigraphic Documents in TEI XML. 23

ETL extract transform load. 28–30, 51, 151

FAIR **F**indability, **A**ccessibility, **I**nteroperable, and **R**euse. 2, 3, 5–8, 20, 28, 50, 119, 121

FHIR Fast Healthcare Interoperability Resources. 68

GUI graphical user interface. 32

HMM hidden Markov model. 13, 106, 108, 110–114, 152

HTML HyperText Markup Language. 12, 27, 53, 84–86, 152

IIIF International Image Interoperability Framework. 23, 24

IIR interactive information retrieval. 97

IR information retrieval. iii, 1, 5, 7, 8, 10, 11, 13–15, 47, 53, 91, 93, 95–106, 114–122, 152

IS information seeking. 97

ISBN International Standard Book Number. 2, 22, 23

iSCD inline subjective content description. 13, 105–113, 121, 152

JPEG Joint Photographic Experts Group. 27

JS JavaScript. 85, 86, 93, 120

JSON JavaScript Object Notation. 24, 46, 47

KML Keyhole Markup Language. 24

LSI latent semantic indexing. 117, 118

METS Metadata Encoding and Transmission Standard. 19–22, 28, 29, 55, 83, 84, 122

ML machine learning. 1, 8, 12, 22

NLP natural language processing. 1

OCR optical character recognition. 93

OOXML Office Open XML. 19, 32, 37, 41, 42

PDF portable document format. 7, 28, 32, 38, 79, 80

QA question answering. 1

RDBMS relational database management system. 47

RDM Center for Sustainable Research Data Management. 7, 22, 23, 28, 83

RDR research data repository. 7, 8, 19, 22–24, 27–29, 54–56, 64, 75, 79–81, 83, 85, 86, 91, 122, 151

ROI regions of interest. 13, 103–105

SCD subjective content description. 13, 105, 121, 152

SQL structured query language. 28, 47, 50

TEI Text Encoding Initiative. 19, 21–23, 28, 29, 47, 50–52, 54–57, 64, 75, 78, 80, 83–88, 90, 91, 120, 151, 152, 155

TIFF Tag Image File Format. 27

TREC Text REtrieval Conference. 10

TXT text file. 27, 31, 32, 68

UID unique identifier. 21, 22

URL Uniform Resource Locator. 20, 22, 23, 53, 83

URN Uniform Resource Name. 23

UVSL U. V. Swaminatha Iyer Library. 28–30, 32, 34

UWA Understanding Written Artefacts. 6

W3C World Wide Web Consortium. 18

WWW World Wide Web. 53

WYSIWYG what you see is what you get. 11, 32, 37, 38

XML extensible markup language. 7, 9–11, 18–24, 28, 32, 37, 39, 41–43, 47, 51, 84, 85, 120

XSD xml schema definition. 10, 19, 21, 120

XSLT eXtensible Stylesheet Language. 9, 84–86, 152

Bibliography

Abhyankar, Swapna, Dina Demner-Fushman, and Clement J. McDonald (Aug. 2012). "Standardizing clinical laboratory data for secondary use". en. In: *Journal of Biomedical Informatics* 45.4, pp. 642–650. DOI: 10.1016/j.jbi.2012.04.012.

Anderson, John R. (Sept. 1, 2007). *How Can the Human Mind Occur in the Physical Universe?* en. Oxford University Press. ISBN: 9780195324259. DOI: 10.1093/acprof:oso/9780195324259.001.0001. URL: https://academic.oup.com/book/4367 (visited on 11/11/2022).

Andreotti, Fernando, Joachim Behar, Sebastian Zaunseder, Julien Oster, and Gari D Clifford (Apr. 2016). "An open-source framework for stress-testing non-invasive foetal ECG extraction algorithms". en. In: *Physiological Measurement* 37.5, pp. 627–648. DOI: 10.1088/0967-3334/37/5/627.

Annotorious (2022). *Annotorious*. en. URL: https://annotorious.github.io/ (visited on 07/31/2023).

Bar-Dayan, Yosefa, Halil Saed, Mona Boaz, Yehudith Misch, Talia Shahar, Ilan Husiascky, and Oren Blumenfeld (Mar. 5, 2013). "Using electronic health records to save money". en. In: *Journal of the American Medical Informatics Association* 20.e1 (e1), e17–e20. ISSN: 1067-5027. DOI: 10.1136/amiajnl-2012-001504. eprint: https://academic.oup.com/jamia/article-pdf/20/e1/e17/6052340/20-e1-e17.pdf.

Behar, Joachim, Fernando Andreotti, Sebastian Zaunseder, Qiao Li, Julien Oster, and Gari D Clifford (July 2014). "An ECG simulator for generating maternal-foetal activity mixtures on abdominal ECG recordings". en. In: *Physiological Measurement* 35.8, pp. 1537–1550. DOI: 10.1088/0967-3334/35/8/1537.

Bender, Magnus, Tanya Braun, Marcel Gehrke, Felix Kuhr, Ralf Möller, and Simon Schiff (2021). "Identifying and Translating Subjective Content Descriptions Among Texts". en. In: *International Journal of Semantic Computing* 15.04, pp. 461–485. DOI: 10.1142/S1793351X21400122.

Burrow, Thomas and Murray Barnson Emeneau (1984). *A Dravidian Etymological Dictionary*. en. Oxford [Oxfordshire]: Clarendon Press. URL: https://dsal.uchicago.edu/dictionaries/burrow/ (visited on 07/27/2023).

Carroll, Stephanie Russo, Ibrahim Garba, Oscar L. Figueroa-Rodríguez, Jarita Holbrook, Raymond Lovett, Simeon Materechera, Mark Parsons, Kay Raseroka, Desi Rodriguez-Lonebear, Robyn Rowe, Rodrigo Sara, Jennifer D. Walker, Jane Anderson, and Maui Hudson (2020). "The CARE Principles for Indigenous Data Governance". en. In: *Data Science Journal* 19.43. DOI: 10.5334/dsj-2020-043.

CERN and OpenAIRE (2013). *Zenodo*. en. DOI: 10.25495/7GXK-RD71. URL: https://www.zenodo.org/ (visited on 10/10/2022).

Cunningham, James A and John D Ainsworth (2015). "Simulating Realistic Enough Patient Records". en. In: *European Federation for Medical Informatics*. Vol. 210. IOS Press Ebooks, pp. 35–39. DOI: 10.3233/978-1-61499-512-8-35.

DataCite Metadata Working Group (Mar. 30, 2021). "DataCite Metadata Schema Documentation for the Publication and Citation of Research Data and Other Research Outputs". en. Version 4.4. In: DOI: 10.14454/3W3Z-SA82.

Dernoncourt, Franck, Ji Young Lee, Ozlem Uzuner, and Peter Szolovits (Dec. 1, 2016). "De-identification of patient notes with recurrent neural networks". en. In: *Journal of the American Medical Informatics Association* 24.3, pp. 596–606. ISSN: 1067-5027. DOI: 10.1093/jamia/ocw156.

Dick, Daniel (2020). *Office Open XML*. en. URL: http://officeopenxml.com/anatomyofOOXML.php (visited on 11/04/2022).

Dominici, Massimiliano (2014). "An overview of Pandoc". en. In: *TUGboat* 35.1, pp. 44–50. URL: https://www.tug.org/TUGboat/tb35-1/tb109dominici.pdf (visited on 07/27/2023).

Emam, Khaled El, Elizabeth Jonker, Luk Arbuckle, and Bradley Malin (Dec. 2, 2011). "A Systematic Review of Re-Identification Attacks on Health Data". en. In: *PLOS ONE* 6.12, pp. 1–12. DOI: 10.1371/journal.pone.0126772.

Forney, George David (1973). "The Viterbi Algorithm". en. In: *Proceedings of the IEEE* 61.3, pp. 268–278. DOI: 10.1109/proc.1973.9030.

Gehrke, Marcel, Simon Schiff, Tanya Braun, and Ralf Moller (Nov. 2019). "Which Patient to Treat Next? Probabilistic Stream-Based Reasoning for Decision Support and Monitoring". en. In: *2019 IEEE International Conference on Big Knowledge (ICBK)*. IEEE, pp. 73–80. DOI: 10.1109/icbk.2019.00018.

GitHub (2023). *The AI-Powered Developer Platform to Build, Scale, and Deliver Secure Software*. en. URL: https://github.com/ (visited on 08/02/2023).

GitLab (2023). *About GitLab*. en. URL: https://about.gitlab.com/ (visited on 08/02/2023).

GO FAIR (2022). *FAIR*. en. GO FAIR International Support and Coordination Office (GFISCO). URL: https://www.go-fair.org/ (visited on 07/31/2023).

Goldberger, Ary L., Luis A. N. Amaral, Leon Glass, Jeffrey M. Hausdorff, Plamen Ch. Ivanov, Roger G. Mark, Joseph E. Mietus, George B. Moody, Chung-Kang Peng, and H. Eugene Stanley (June 2000). "PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals". en. In: *Circulation* 101.23. DOI: 10.1161/01.cir.101.23.e215.

Hameed, Mazhar and Felix Naumann (Dec. 2020). "Data Preparation: A Survey of Commercial Tools". en. In: *ACM SIGMOD Record* 49.3, pp. 18–29. ISSN: 0163-5808. DOI: 10.1145/3444831.3444835.

Hellingman, Jeroen (Nov. 6, 2022). *tei2html*. en. URL: https://github.com/jhellingman/tei2html (visited on 07/31/2023).

HEURIST (2022). *A unique solution to the data management needs of Humanities researchers*. en. The University of Sydney. URL: https://heuristnetwork.org/ (visited on 11/09/2022).

Information Technology Laboratory and Information Access Division (2014). *TREC Session Track*. en. National Institute of Standards and Technology. URL: https://trec.nist.gov/data/session2014.html (visited on 11/30/2022).

Jahnen, Burkhard and Katja Hartig (2022). *Specification of Requirements Relating to the Handling of Research Data in Funding Proposals*. en. DFG. URL: https://www.dfg.de/en/research_funding/announcements_proposals/2022/info_wissenschaft_22_25/index.html (visited on 07/31/2023).

Johnson, Alistair, Tom Pollard, and Roger Mark (2016). *MIMIC-III Clinical Database*. Version 1.4. DOI: https://doi.org/10.13026/C2XW26.

Johnson, Alistair E.W., Tom J. Pollard, Lu Shen, Li-wei H. Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G. Mark (May 2016). "MIMIC-III, a freely accessible critical care database". en. In: *Scientific Data* 3.1. DOI: 10.1038/sdata.2016.35.

Johnson, Michael L., Lenore Pipes, Paula P. Veldhuis, Leon S. Farhy, David G. Boyd, and William S. Evans (Oct. 2008). "AutoDecon, a deconvolution algorithm for identification and characterization of luteinizing hormone secretory bursts: Description and validation using synthetic data". en. In: *Analytical Biochemistry* 381.1, pp. 8–17. DOI: 10.1016/j.ab.2008.07.001.

Kaggle (2022). *Find Open Datasets and Machine Learning Projects*. en. Kaggle. URL: https://www.kaggle.com/datasets (visited on 11/09/2022).

Kambhampati, Subbarao (Sept. 2020). "Challenges of Human-Aware AI Systems". en. In: *AI Magazine* 41.3, pp. 3–17. DOI: 10.1609/aimag.v41i3.5257.

Kanoulas, Evangelos, Leif Azzopardi, and Grace Hui Yang (2018). "Overview of the CLEF Dynamic Search Evaluation Lab 2018". en. In: *Lecture Notes in Computer Science*. Ed. by Patrice Bellot, Chiraz Trabelsi, Josiane Mothe, Fionn Murtagh, Jian

Yun Nie, Laure Soulier, Eric SanJuan, and Nicola Cappellato Lindaand Ferro. Springer International Publishing, pp. 362–371. ISBN: 978-3-319-98932-7. DOI: 10.1007/978-3-319-98932-7_31.

Khalilia, Mohammed, Myung Choi, Amelia Henderson, Sneha Iyengar, Mark Braunstein, and Jimeng Sun (Nov. 5, 2015). "Clinical Predictive Modeling Development and Deployment through FHIR Web Services". en. In: *AMIA Annual Symposium Proceedings*. Vol. 2015. American Medical Informatics Association, p. 717. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4765683/pdf/2248709.pdf.

Khan, Saima, Shazia Khan, and Mohsina Aftab (2015). "Digitization and its Impact on Economy". en. In: *International Journal of Digital Library Services* 5.2, pp. 128–149. ISSN: 2250-1142. URL: http://www.ijodls.in/uploads/3/6/0/3/3603729/vol-5,_issue-2.138-149.pdf (visited on 07/31/2023).

Kielhorn, Axel (Jan. 1, 2011). "Multi-target publishing". en. In: *TUGboat-TeX Users Group* 32.3, pp. 272–277. URL: https://www.tug.org/TUGboat/tb32-3/tb102kielhorn.pdf (visited on 10/11/2022).

Krijnen, Jacco, Doaitse Swierstra, and Marcos O. Viera (2014). "Expand: Towards an extensible Pandoc system". en. In: *Practical Aspects of Declarative Languages*. Springer International Publishing, pp. 200–215. ISBN: 978-3-319-04132-2. DOI: 10.1007/978-3-319-04132-2_14.

Kuhr, Felix, Tanya Braun, Magnus Bender, and Ralf Möller (Nov. 25, 2019). "To Extend or Not to Extend? Context-Specific Corpus Enrichment". en. In: *AI 2019: Advances in Artificial Intelligence*. Ed. by James Liu Jixueand Bailey. Springer International Publishing, pp. 357–368. ISBN: 978-3-030-35288-2. DOI: 10.1007/978-3-030-35288-2_29.

Kumar, D. Udaya, G.V. Sreekumar, and U. A. Athvankar (2009). "Traditional writing system in Southern India - Palm leaf manuscripts". en. In: *IDC School of Design* 7, pp. 2–7. URL: http://www.idc.iitb.ac.in/resources/dt-july-2009/Palm.pdf (visited on 07/31/2023).

Longwood Research Data Management (2022). *Biomedical Data Lifecycle*. en. LMA Research Data Management Working Group. URL: https://datamanagement.hms.harvard.edu/plan-design/biomedical-data-lifecycle (visited on 07/27/2023).

Luo, Jiyun, Xuchu Dong, and Hui Yang (2015). "Session Search by Direct Policy Learning". en. In: *Proceedings of the 2015 International Conference on The Theory of Information Retrieval*. ICTIR '15. Association for Computing Machinery, pp. 261–270. ISBN: 9781450338332. DOI: 10.1145/2808194.2809461.

Luo, Jiyun, Christopher Wing, Hui Yang, and Marti Hearst (2013). "The Water Filling Model and the Cube Test: Multi-Dimensional Evaluation for Professional Search".

en. In: *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management - CIKM '13*. ACM Press, pp. 709–714. ISBN: 9781450322638. DOI: 10.1145/2505515.2523648.

Luo, Jiyun, Sicong Zhang, and Hui Yang (July 2014). "Win-Win Search: Dual-Agent Stochastic Game in Session Search". en. In: *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*. SIGIR '14. Association for Computing Machinery, pp. 587–596. ISBN: 9781450322577. DOI: 10.1145/2600428.2609629.

MacFarlane, John (2022). *Pandoc User's Guide*. en. URL: https://pandoc.org/MANUAL.html (visited on 07/27/2023).

Marchionini, G. and B. Shneiderman (Jan. 1988). "Finding facts vs. browsing knowledge in hypertext systems". en. In: *Computer* 21.1, pp. 70–80. DOI: 10.1109/2.222119.

Marlin, Benjamin M., David C. Kale, Robinder G. Khemani, and Randall C. Wetzel (Jan. 28, 2012). "Unsupervised Pattern Discovery in Electronic Health Care Data Using Probabilistic Clustering Models". en. In: *Proceedings of the 2nd ACM SIGHIT International Health Informatics Symposium*. ACM. ACM, pp. 389–398. DOI: 10.1145/2110363.2110408.

Martinez Cobo, José R. (1987). "Study of the Problem of Discrimination Against Indigenous Populations". en. In: *Conclusions, Proposals and Recommendations*. Vol. 5. United Nations. URL: https://documents-dds-ny.un.org/doc/UNDOC/GEN/N87/121/00/PDF/N8712100.pdf?OpenElement (visited on 08/01/2023).

Melzer, Sylvia, Simon Schiff, and Ralf Möller (2021). *Complementary Document Representations for Information Retrieval*. en. DOI: 10.25592/UHHFDM.9569.

– (2022). "Databasing on Demand and Federated Search in Manuscript Databases". en. In: *School of Languages and Cultures*.

Melzer, Sylvia, Simon Schiff, Franziska Weise, Kaja Harter-Uibopuu, and Ralf Möller (2022). "Databasing on demand for research data repositories explained with a large epidoc dataset". en. In: *CENTERIS 2022 - Conference on ENTERprise Information Systems*. URL: https://www.fis.uni-hamburg.de/en/publikationen/detail.html?id=0d19af7f-b6c7-4b05-98f7-1082b77d5ad0 (visited on 07/31/2023).

Melzer, Sylvia, Stefan Thiemann, Simon Schiff, and Ralf Möller (Aug. 22, 2023). "Implementation of a Federated Information System by means of Reuse of Research Data archived in Research Data Repositories". In: *Data Science Journal*.

Möller, Ralf (2021). "Humanities-Centered AI: From Machine Learning to Machine Training". en. In: *CHAI@KI*, pp. 40–44. URL: https://ceur-ws.org/Vol-3093/paper5.pdf (visited on 07/31/2023).

Murray, Richard E, Patrick B Ryan, and Stephanie J Reisinger (Oct. 22, 2011). "Design and Validation of a Data Simulation Model for Longitudinal Healthcare Data". en. In: *AMIA Annual Symposium Proceedings*. Vol. 2011. American Medical Informatics Association.

Nguyen, Nhan and Sarah Nadi (Oct. 17, 2022). "An Empirical Evaluation of GitHub Copilot's Code Suggestions". en. In: *Proceedings of the 19th International Conference on Mining Software Repositories*. MSR '22. Association for Computing Machinery, pp. 1–5. ISBN: 9781450393034. DOI: 10.1145/3524842.3528470.

Padmakumar, P.K., V.B. Sreekumar, V.V. Rangan, and C. Renuka (2003). "Palm Leaves as Writing Material: History and Method of Processing in Kerala". en. In: *The International Palms Society* 47, pp. 125–129. URL: https://palms.org/wp-content/uploads/2016/05/vol47n3p125-129-1.pdf (visited on 07/31/2023).

Parr, Terence (2013). *The Definitive Antlr 4 Reference*. en. Pragmatic Bookshelf, pp. 1–326. ISBN: 9781934356999. URL: https://dl.acm.org/doi/10.5555/2501720 (visited on 07/31/2023).

Parr, Terence, Sam Harwell, and Kathleen Fisher (Oct. 2014). "Adaptive LL(∗) parsing". en. In: *ACM SIGPLAN Notices* 49.10, pp. 579–598. DOI: 10.1145/2714064.2660202.

Paskin, N. (July 1999). "Toward Unique Identifiers". en. In: *Proceedings of the IEEE* 87.7, pp. 1208–1227. DOI: 10.1109/5.771073.

Perera, Sujan, Amit Sheth, Krishnaprasad Thirunarayan, Suhas Nair, and Neil Shah (Nov. 1, 2013). "Challenges in Understanding Clinical Notes: Why NLP Engines Fall Short and Where Background Knowledge Can Help". en. In: *Proceedings of the 2013 international workshop on Data management & analytics for healthcare - DARE '13*. ACM. ACM Press, pp. 21–26. DOI: 10.1145/2512410.2512427.

Peters, Isabella, Peter Kraker, Elisabeth Lex, Christian Gumpenberger, and Juan Ignacio Gorraiz (Dec. 22, 2017). "Zenodo in the Spotlight of Traditional and New Metrics". en. In: *Frontiers in Research Metrics and Analytics* 2. ISSN: 2504-0537. DOI: 10.3389/frma.2017.00013.

Python Software Foundation (Oct. 11, 2022). *The ElementTree XML API*. en. URL: https://docs.python.org/3/library/xml.etree.elementtree.html# (visited on 07/31/2023).

Rahtz, Sebastian (2022). *TEI Stylesheets*. en. Text Encoding Initiative (TEI). URL: https://github.com/TEIC/Stylesheets (visited on 07/31/2023).

Rajalingam, Damodharan (2022). *An Affix Stripping Iterative Stemming Algorithm for Tamil*. en. URL: https://github.com/rdamodharan/tamil-stemmer (visited on 07/31/2023).

Rajaraman, Anand and Jeffrey David Ullman (Oct. 2011). *Mining of Massive Datasets*. en. Cambridge University Press. DOI: 10.1017/cbo9781139058452. URL: https://www.cambridge.org/core/books/mining-of-massive-datasets/A06D57FC616AE3FD10007D89E73F8B92 (visited on 11/18/2022).

Schiff, Simon, Magnus Bender, and Ralf Möller (Sept. 19, 2022). "Embodiment of an Agent by a Pepper Robot for Explaining Retrieval Results". en. In: *Proceedings of the Workshop on Humanities-Centred Artificial Intelligence (CHAI 2022)*. CEUR Workshop Proceedings, pp. 29–37. URL: https://ceur-ws.org/Vol-3301/paper4.pdf (visited on 08/01/2023).

Schiff, Simon, Marcel Gehrke, and Ralf Möller (2018). "Efficient Enriching of Synthesized Relational Patient Data with Time Series Data". en. In: *Procedia Computer Science* 141, pp. 531–538. DOI: 10.1016/j.procs.2018.10.130.

Schiff, Simon, Felix Kuhr, Sylvia Melzer, and Ralf Möller (2020). "AI-based Companion Services for Humanities". en. In: *KI2020*. URL: https://www.ifis.uni-luebeck.de/uploads/tx_wapublications/SchiffKuhrSylviaMoeller-Extended-Abstract_01.pdf (visited on 07/31/2023).

Schiff, Simon, Mena Leemhuis, Özgür Özcep, and Ralf Möller (May 15, 2023). "Query Transformation for Processing Streams in Decision-making Agents". en. In: *The International FLAIRS Conference Proceedings* 36. DOI: 10.32473/flairs.36.133104.

Schiff, Simon, Sylvia Melzer, Eva Wilden, and Ralf Möller (May 22, 2022). "TEI-Based Interactive Critical Editions". en. In: *Document Analysis Systems*. Ed. by Seiichi- and Barney Uchida and Véronique Elisaand Eglin. Springer International Publishing, pp. 230–244. ISBN: 978-3-031-06555-2. DOI: 10.1007/978-3-031-06555-2_16.

Schiff, Simon and Ralf Möller (Sept. 28, 2021). "On Human-Aware Information Seeking". en. In: *Proceedings of the Workshop on Humanities-Centred Artificial Intelligence (CHAI 2021)*. CEUR Workshop Proceedings, pp. 31–39. URL: https://ceur-ws.org/Vol-3093/paper4.pdf (visited on 07/31/2023).

– (2023). "Persistend Data, Sustainable Information". en. In: *Proceedings of the Workshop on Humanities-Centred Artificial Intelligence (CHAI 2023)*. to appear. CEUR Workshop Proceedings.

Schiff, Simon, Ralf Möller, and Özgür L. Özçep (2019). "Ontology-Based Data Access to Big Data". en. In: *Open Journal of Databases* 6, pp. 21–32. ISSN: 2199-3459. URL: http://www.ronpub.com/ojdb/OJDB_2019v6i1n03_Schiff.html.

Schiff, Simon and Özcep Özgür (2020). "Bounded-Memory Criteria for Streams with Application Time". en. In: *The International FLAIRS Conference Proceedings*. URL: https://cdn.aaai.org/ocs/18421/18421-79322-1-PB.pdf (visited on 07/31/2023).

Al-Shboul, Mohammad Khaled and Abdullah Abrizah (Sept. 1, 2014). "Information Needs: Developing Personas of Humanities Scholars". en. In: *The Journal of Academic Librarianship* 40.5, pp. 500–509. DOI: 10.1016/j.acalib.2014.05.016.

Shi, Zhixin, Srirangaraj Setlur, and Venu Govindaraju (2004). "Digital Enhancement of Palm Leaf Manuscript Images using Normalization Techniques". en. In: *5th International Conference On Knowledge Based Computer Systems*. URL: https://cedar.buffalo.edu/~zshi/Papers/kbcs04_261.pdf (visited on 08/01/2023).

Sicilia, Miguel-Angel, Elena García-Barriocanal, and Salvador Sánchez-Alonso (2017). "Community curation in open dataset repositories: insights from Zenodo". en. In: *Procedia Computer Science* 106, pp. 54–60. ISSN: 1877-0509. DOI: 10.1016/j.procs.2017.03.009.

Smith, John Miles, Philip A. Bernstein, Umeshwar Dayal, Nathan Goodman, Terry Landers, Ken W. T. Lin, and Eugene Wong (1981). "Multibase: integrating heterogeneous distributed database systems". en. In: *Proceedings of the May 4-7, 1981, national computer conference on - AFIPS '81*. AFIPS '81. ACM Press, pp. 487–499. ISBN: 9781450379212. DOI: 10.1145/1500412.1500483.

Society of American Archivists (2023). *Dictionary of Archives Terminology*. en. Society of American Archivists. URL: https://dictionary.archivists.org/entry/born-digital.html (visited on 07/27/2023).

Subramanian, T. S. (July 5, 2013). *Abode of legacy*. en. The Hindu. URL: https://www.thehindu.com/features/friday-review/history-and-culture/abode-of-legacy/article4880651.ece (visited on 07/31/2023).

Tang, Zhiwen and Grace Hui Yang (Mar. 2022). "A Re-classification of Information Seeking Tasks and Their Computational Solutions". en. In: *ACM Transactions on Information Systems* 40.4, pp. 1–32. ISSN: 1046-8188. DOI: 10.1145/3497875. URL: https://doi.org/10.1145/3497875.

Text Encoding Initiative (Oct. 25, 2022a). *Dictionaries*. en. Text Encoding Initiative. URL: https://tei-c.org/release/doc/tei-p5-doc/en/html/DI.html (visited on 10/19/2022).

– (Oct. 25, 2022b). *Text Encoding Initiative Guidelines*. en. URL: https://tei-c.org/release/doc/tei-p5-doc/en/html/index.html (visited on 07/31/2023).

The Library of Congress (2022). *Metadata Encoding and Transmission Standard: Prime and Reference Manual*. en. URL: https://www.loc.gov/standards/mets/METSPrimer.pdf (visited on 10/06/2022).

The PostgreSQL Global Development Group (2023). *PostgreSQL 15.3 Documentation*. en. The PostgreSQL Global Development Group. URL: https://www.postgresql.org/docs/15/index.html (visited on 07/31/2023).

The TExt Retrieval Conference (2002). *TREC Interactive Track*. en. National Institute of Standards and Technology. URL: https://trec.nist.gov/data/t11_interactive/t11i.html (visited on 08/01/2023).

Thiemann, Kathrin (Nov. 7, 2019). *Broschüre Nachhaltiges Forschungsdatenmanagement*. de. Version 2. Universität Hamburg. DOI: 10.25592/UHHFDM.699. URL: https://www.fdm.uni-hamburg.de/fdm/fdm-broschuere-online.pdf (visited on 12/09/2022).

Universität Konstanz (2022). *Forschungsdaten und Forschungsdatenmanagement*. de. URL: https://forschungsdaten.info/themen/informieren-und-planen/datenmanagementplan/ (visited on 07/31/2023).

University of Leeds (2022). *Research data management explained*. en. University of Leeds. URL: https://library.leeds.ac.uk/info/14062/research_data_management/61/research_data_management_explained (visited on 07/04/2023).

Walonoski, Jason, Mark Kramer, Joseph Nichols, Andre Quina, Chris Moesel, Dylan Hall, Carlton Duffett, Kudakwashe Dube, Thomas Gallagher, and Scott McLachlan (2018). "Synthea: An approach, method, and software mechanism for generating synthetic patients and the synthetic electronic health care record". en. In: *Journal of the American Medical Informatics Association* 25.3, pp. 230–238. DOI: 10.1093/jamia/ocx079.

Watkins, Rochelle E, Serryn Eagleson, Bert Veenendaal, Graeme Wright, and Aileen J Plant (Aug. 10, 2009). "Disease surveillance using a hidden Markov model". en. In: *BMC Medical Informatics and Decision Making* 9.1, p. 39. ISSN: 1472-6947. DOI: 10.1186/1472-6947-9-39.

Welch, Lloyd R (2003). "Hidden Markov Models and the Baum-Welch Algorithm". en. In: *IEEE Information Theory Society Newsletter* 53.4. Ed. by Lance C. Pérez, pp. 10–13. ISSN: 1059-2362. URL: http://yanfenglu.net/documents/Baum-Welch_Algorithm.pdf (visited on 07/31/2023).

Wilden, Eva (2018). *A critical edition and an annotated translation of the Akanānūru*. en. Ecole Francaise D'extreme-Orient. URL: https://publications.efeo.fr/en/livres/908_a-critical-edition-and-an-annotated-translation-of-the-akan-u-part-1-ka-i-iy-ainirai (visited on 07/31/2023).

World Wide Web Consortium (2022a). *Extensible Markup Language (XML)*. en. URL: https://www.w3.org/XML/ (visited on 07/31/2023).

– (2022b). *XSLT Cover Page*. en. URL: https://www.w3.org/TR/xslt/ (visited on 07/31/2023).

Wu, Jionglin, Jason Roy, and Walter F. Stewart (June 2010). "Prediction Modeling Using EHR Data: Challenges, Strategies, and a Comparison of Machine Learning Approaches". In: *Medical Care*, S106–S113. DOI: 10.1097/mlr.0b013e3181de9e17.

Yang, Angela and Grace Hui Yang (Oct. 2017). "A Contextual Bandit Approach to Dynamic Search". en. In: *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval*. ICTIR '17. Association for Computing Machinery, pp. 301–304. ISBN: 9781450344906. DOI: 10.1145/3121050.3121101.

Yang, Grace Hui, Ian Soboroff, Jiyun Luo, and Zhiwen Tang (2017). *TREC Dynamic Domain Track*. en. URL: https://infosense.cs.georgetown.edu/trec_dd/index.html (visited on 08/01/2023).

Zhang, Yu, Sarath Sreedharan, Anagha Kulkarni, Tathagata Chakraborti, Hankz Hankui Zhuo, and Subbarao Kambhampati (May 2017). "Plan Explicability and Predictability for Robot Task Planning". en. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 1313–1320. DOI: 10.1109/icra.2017.7989155.

Zwass, Vladimir (June 22, 2023). *Information System*. en. Ed. by The Editors of Encyclopedia Britannica. Britannica. URL: https://www.britannica.com/topic/information-system (visited on 08/01/2023).