The AXF (Annotation Exchange Format) Platform

LTS is founded by Amy Neustein, PhD, Series Editor of **Speech Technology and Text Mining in Medicine and Health Care** *(de Gruyter); and Editor of* **Advances in Ubiquitous Computing: Cyber-Physical Systems, Smart Cities, and Ecological Monitoring** *(Elsevier, forthcoming). These publishers have placed Dr. Neustein's publications on their open access portals linked from* **CORD-19** *(discussed below).*

The **AXF** Platform (hereafter called just **AXF**) is a toolkit for hosting full-text, open access publications, with an emphasis on scientific, academic, and technical documents. At the core of an **AXF** Publication Repository is a collection of files in a machine-readable **AXF** Document Format (**AXFD**), which are paired with human-readable **PDF** documents as well as supplemental multi-media and metadata files. Depending on institutional requirements, an **AXF** repository may be the primary storage resource for the contained publications, or an adjunct resource whose documents are linked to publications hosted elsewhere. In the second scenario, the primary goal of an **AXF** repository is to host manuscripts in **AXFD** format, along with software to aid viewing and text-mining of the associated publications.

AXFD therefore has two distinct purposes: (1) to aid in text- and data-mining (**TDM**) of full publication text (along with research data that may be linked to publications), and (2) to enhance the reader experience, given e-Reader software (canonically, **PDF** viewers) which are programmed to consume **AXF** information. To aid (1) in text mining, **AXFD** documents can be compiled into different structured representations, yielding document versions that can be registered on services such as CrossREF **TDM** and SemanticScholar. Given a Document Object Identifier, text-mining tools can therefore readily obtain a highly structured, machine readable version of the publication, which may then be used as the basis of further text-mining and **NLP** operations. Simultaneously, to (2) improve reader experience, the **AXF** platform generates numeric data linking semantically significant text locations to **PDF** viewport coordinates (such text locations include annotation, quotation, or citation start/end points and paragraph or sentence boundaries — collectively dubbed a Semantic Document Infoset, or **SDI**). This **SDI**+Viewport (**SDIV**) information can then be used by **PDF** applications to provide contexts for word searches, to localize context menus, to activate multi-media features at different points in the text, and in general to make **PDF** files more interactive. Data sets composed in conjunction with **AXF** may include source code for a **PDF** viewer (an extension to **XPDF**) capable of leveraging **AXF** data.

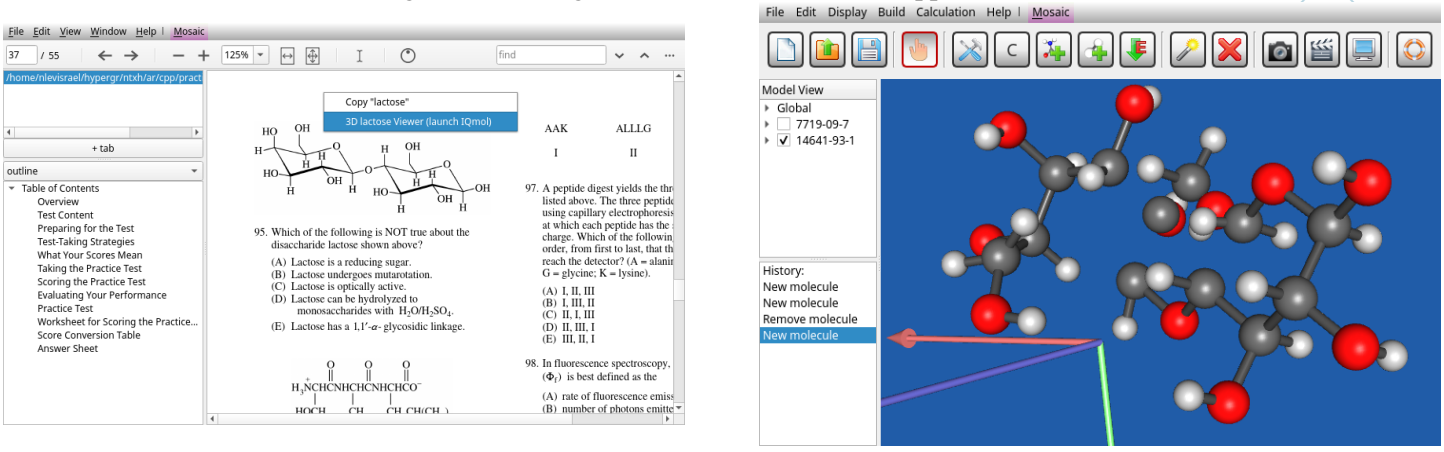
In addition to the **AXFD** document format, the **AXF** platform includes the Annotation Exchange Format itself, a protocol for defining and sharing annotations on full-text publications. Linguistic Technology Systems has developed **AXF** to address shortcomings of existing annotation standards, particularly in the context of annotation environments integrating application, semantic, and multimedia content. The goal of **AXF** is to represent annotations at a novel level of detail. In particular (as will be described below), the *target* of the annotation should be represented in both semantic and interactive contexts (to include, for instance, both sentence-level context and **PDF** viewport coordinates). Meanwhile, the *body* of the annotation (using terms originating with the Linguistic Annotation Framework) can be linked to data "microcitations", multimedia assets, or controlled vocabularies. **AXF** is optimized for **AXFD** manuscripts — i.e., documents encoded according to the native **AXF** protocol — but **AXF** can potentially be used as an exchange format for document-annotation operations in a variety of contexts (named entity recognizers, science-data **API**s, publication metadata sharing, and so forth.

The following sections will (1) outline **AXF** and **AXFD** in greater detail and (2) describe how **AXF** repositories can unify publications sharing similar themes, scholarly disciplines, or coding requirements.

I AXF and AXF Documents

The **AXFD** format for describing document content and structure is designed to be a "Pivot Representation" in the sense of **LAF**. In particular, **AXFD** can represent the structure of both **XML** (including several **XML** flavors used in publishing) and LaTeX. Technically, **AXFD** does not prescribe any specific input format; instead, a document is considered an instance of **AXFD** if it can be compiled into a Document Object satisfying interface requirements. A **C++** reference implementation anchors the **AXF** Document Object Model; nodes in this implementation have facets combining **LAF**, **XML**, and LaTeX. In practice, **AXFD** manuscripts are then converted via LaTeX to **PDF**, and simultaneously compiled to **XML** representations so as to generate machine-readable, structured full-text versions of the manuscripts. Authors can choose to compose **AXFD** papers to conform with several common publication **XML** standards, such as **JATS** (Journal Article Tag Suite), **SciXML** [17],

Figure 1: Linking PDF Files with Scientific Applications



and **IEXML** [16] (the latter is an annotation-oriented **XML** language used by the BeCAS project [12]). Authors or editors may further define or model scientific concepts via strategies focused on computer simulation (in the broad sense as defined, for instance, in [19]) or statistics (e.g., Predictive Model Markup Language or Attribute-Relation File Format), and/or conceptual semantics, such as Conceptual Space Markup Language (which was inspired by the linguist Peter Gärdenfors, but developed in a scientific/mathematical framework in e.g. [1], [2], [7], and [4]).

One distinct feature of **AXF** is that LaTeX and **XML** generation are chained in a pipeline: the LaTeX and subsequent **PDF** generation steps yield auxiliary data, which includes **PDF** viewport data. Specifically, **AXFD**-generated LaTeX files include notations for semantic annotations and for sentence boundaries, implemented via LaTeX commands which, as one processing step, write **PDF** coordinates to auxiliary files. The resulting data is then read by a **C++** program which collates annotations and sentence-boundaries into a vector of data structures indexed by **PDF** page numbers, creating a distinct file for each page, and zips those files into an archive which can be distributed alongside (or embedded inside) the **PDF** publications. Simultaneously, sentences, paragraphs, annotations, and other semantically significant content (such as quotations and citations) are assigned unique ids and compiled into their own data structures (from which machine-readable **XML** full-text may be generated). These **XML** files may then be hosted and/or registered on **TDM**-oriented services such as CrossRef. At the same time, unique identifier unify this **XML** data (focused on text mining) with **PDF** viewport data (focused on reader experience). The goal of such integration is to incorporate text-mining results so as to enhance reader experience. For example, Named Entity Recognizers might flag a word-sequence as matching a concept within a controlled vocabulary. Via the relevant paper's **SDI** model, this annotation may be placed in a proper semantic context — for example, obtaining the text of the sentence where the Named Entity occurs. This semantic information may then be used by a **PDF** viewer — e.g., providing a context menu option to select the sentence text, when the context menu is activated within the rectangular coordinates of the annotation itself.

As a representation of annotation data structures, **AXF** ensures that **SDI** and viewport data is included among annotations wherever this data is available. This facilitates the integration between text-mining tools and **PDF** viewer software, which in turn enhances reader experience. As mentioned earlier, every annotation can be placed in a semantic context (e.g., the text of the surrounding sentence), which provides useful reader features such as one-click copying of sentences to the clipboard. Other reader-experience enhancements involve multimedia assets. As a concrete example, suppose a paper includes mention of a chemical; that particular keyword can accordingly be flagged for annotation. As one encoding of the corresponding scientific concept, the annotation can include the chemical's Chemical Abstract Service Reference Number, via which it is possible to obtain Protein Data Bank files to view the molecule in **3D**. Therefore, annotations supply a constellation of data — in this example, concepts may be linked not only to identifiers in cheminformatic ontologies, but also to reference numbers and thereby to **3D** graphics files — which facilitate interactive User Experience at the application level, not only document classification at the corpus level. Once a chemical compound (mentioned in a publication) is linked to a **PDB** file (or any other **3D** format) the **PDF** viewer may include options to for the reader to connect to software or web applications where the corresponding visuals can be rendered. Via **AXF**, the relevant document-to-software connections are asserted not only on the overall document level, but on the granular scale of the precise character and **PDF** viewport coordinates where the relevant annotation is grounded (Figure 1 illustrates such capabilities in the context of a chemistry publication — specifically, test-preparation materials for the Chemistry **GRE** exam).

To support this kind of multimedia functionality, **AXF** standardizes a Plugin Framework, dubbed "**MOSAIC**", allowing programmers to embed code which can parse and respond to **AXF** annotations in different scientific and document-viewer applications. **MOSAIC** allows different applications to inter-operate; in particular, **PDF** viewers can share data with scientific applications that can render files in domain-specific formats such as **PDB**. This application networking protocol is considered part of the **AXF** annotation model, because application-oriented information is computationally relevant for many concepts encountered in scientific and technical environments. For instance, one aspect of cheminformatic data is

that many chemical compounds are modeled by **PDB**, **MOL**, or files, which in turn are associated by software applications that can load files with those file types. This inter-application networking data then becomes relevant to **PDF** viewers when displaying manuscripts with annotations that suggest links to special file types and their applications; the viewers can employ this information to launch and/or communicate with the corresponding software. **AXF** is designed to facilitate implementation of application-networking protocols as an operational continuation of processes related to obtaining and consuming annotation data.

The **AXF** document model, at the manuscript-structure level, is paired with a novel "Hypergraph Text Encoding Protocol" (**HTXN**) operating at the character-encoding level. Within the **HTXN** protocol, an annotation target is a character-index interval in the context of an **HTXN** character stream. On that basis, **HTXN** treats documents as graphs whose nodes are ranges in a character stream, where text can be recovered as an operation on one or more nodes (e.g., the text of a sentence is derived from a pair of nodes representing the sentence's start and end). **HTXN** code-points are distinguished in terms of their semantic role, which may be more granular than their visible appearance — for example, a period glyph is assigned different code-points depending on whether it marks a sentence-ending punctuation, an abbreviation, a decimal point, or part of an ellipsis. Procedures are then implemented to represent text in different formats, such as **ASCII**, Unicode, **XML**, or LaTeX. In contrast to a format such as Web Annotations, any particular human-readable text presentation (including **ASCII**) is considered a *derived* property of the annotation, not a foundational representation.

AXFD manuscripts do not need to utilize **HTXN** for character data, but **HTXN** simplifies certain **AXF** operations, such as identifying sentence boundaries. In particular, **HTXN** provides distinct code-points for end-of-sentence punctuation, so that sentence-boundary detection reduces to a trivial search for those particular code-points. Proper **HTXN** encoding requires that authors follow certain simple heuristics — e.g., that end-of-sentence periods should be followed by two spaces and/or a newline, whereas other uses of a period character should precede at most one space. Aside from the goal of preparing documents for text-mining machine-readability, such conventions are appropriate even for basic typesetting, because non-punctuation characters have distinct kerning rules (this is why LaTeX provides a distinct command for non-punctuation glyphs that would otherwise be read as punctuation characters). **HTXN** hides many of these typesetting details within its character-encoding schema, which is useful both for producing professional-caliber LaTeX output and for identifying **SDI** details (such as sentence boundaries) which with less rigorously structured text would need elaborate text-mining or **NLP** algorithms.

Each **AXFD** document is, in sum, associated with an aggregate of character-encoding, annotation, document-structure, and **PDF** viewport information. The **AXF** platform uses code libraries to pull this information together as a runtime object system, so that any application which loads an **AXFD** manuscript can execute queries against the corresponding collection of **AXF** objects (queries such as obtaining the sentence text around an annotation, obtaining the concave-octagonal viewport coordinates for a sentence,[1] obtaining application-networking information for an annotation, etc.) In addition to such runtime data, **AXF** platforms can compile the full suite of information into machine-readable files for text and data mining. These files, collected across a corpus of multiple documents, then form the backbone of an **AXF** publication repository, as will be discussed next.

II AXF Publication Repositories

The **AXF** platform is designed for hosting collections of publications sharing a common academic or technical focus. If **AXF** is used in the context of a general-purpose text and/or data repository, the platform is designed to work with collections that are organized into separate projects or topics, each giving rise to an archive or corpus of publications. Insofar as these corpora internally share a common theme or focus, they can be associated with their own ontologies, code libraries, annotation models, and application-networking protocols, based on the sorts of applications and data structures commonly used in the corresponding scholarly discipline. In some cases, publishers may choose to package an entire archive of research papers (perhaps along with research data) as a single downloadable resource. **AXF** allows publishers to construct such Research Archives following the structure of existing examples such as the **ACL** (Association for Computational Linguistics) Anthology or the recent **CORD-19** corpus. This latter archive is a useful case-study in both the possibilities and limitations of existing publication-repository technology.

CORD-19, curated by the Allen Institute for Artificial Intelligence, was spearheaded by a White House initiative to centralize scientific research related to **COVID-19**. The collection was formulated with the explicit goal of promoting both *text mining* and *data mining* solutions to advance coronavirus research, so that **CORD-19** is intended to be used both as a document

[1] In the general case, sentence coordinates are concave octagons because they incorporate the line height of their start and end lines; in the general case sentences share start and end lines with other sentences, while also including whole lines vertically positioned between these extrema. A sentence octagon roughly corresponds with the screen area where a mouse/pointer action should be understood as occurring in the context of that sentence from the user's point of view — implying that the user would benefit from context menu options pertaining specifically to that sentence, such as copy-to-clipboard.

archive for text mining and as a repository for finding and obtaining coronavirus data for subsequent research. Although novel research is being incorporated into CORD-19, many of the articles reproduced in this corpus are older publications related to coronaviruses and to SARS in general, not just to the current pandemic. As a result, the full-text versions of these publications were retroactively aggregated into a single archive due to the unanticipated emergence of a coronavirus crisis, with the full text often obtained from PDF files rather than from structured representations (such as JATS) explicitly intended for text mining.

This archival methodology results in CORD-19 being limited as a TDM framework. These limitations include the following:

Transcription Errors Transcription errors can easily result from trying to read scientific data and notations based on PDF files — or on full-text representations using relatively unstructured formats such as (the response-encoding format for the ScienceDirect API). Transcription errors cause the machine-readable text archive to misrepresent the structure and content of documents. For instance, there are cases in CORD-19 of scientific notation and terminology being improperly encoded. As a concrete example, "2'-C-ethynyl" is encoded incorrectly in one CORD-19 file as "2 0 -C-ethynyl" (see [6] for the human-readable publication where this error is observed; the corresponding index in the corpus is 9555f44156bc5f2c6ac191dda2fb651501a7bd7b.json). To help address these sorts of errors — which could stymie text searches against the CORD-19 corpus — it is obviously preferable to archive structured, machine-readable versions of publications, using a platform such as AXF.

Converting Between Data Formats Although the CORD-19 corpus is published as JSON files, many text-mining tools such as those reviewed in [11] recognize inputs or produce outputs in alternative formats, such as XML, BioC, CoNLL (Conference on Natural Language Learning), or JSON trees with different schema than CORD-19. For this reason, rather than providing data with one single representational format, it is better to encode the data along with code libraries that can express the data in different formats as needed for different TDM ecosystems.

Inconsistent Annotations The structure of CORD-19 allows text segments to be defined via a combination of JSON file names, paragraph ids, and character indices. This indexing schema is used for representing certain internal details of individual articles, such as citations, but is not explicitly defined as an annotation target structure for standoff annotations against the archive as a whole. This problem could also be rectified with code libraries that map index targets to file handles and character pointers.

Limited Support for Research Data-Mining Even though many papers in CORD-19 are paired with published data sets, there is currently no tool for locating research *data* through CORD-19. For example, the collection of manuscripts available through the Springer Nature portal linked from CORD-19 includes over 30 CoVID-19 data sets, but researchers can only discover that these data sets exist by looking for a "supplemental materials" or a "data availability" addendum near the end of each article. These Springer Nature data sets encompass a wide array of file types and formats, including FASTA (which stands for Fast-All, a genomics format), SRA (Sequence Read Archive, for DNA sequencing), PDB (Protein Data Bank, representing the 3D geometry of protein molecules), MAP (Electron Microscopy Map), EPS (Embedded Postscript), CSV (comma-separated values), and tables represented in Microsoft Word and Excel formats. To make this data more readily accessible in the context of CORD-19, it would be appropriate to (1) maintain an index of data sets linked to CORD-19 articles and (2) merge these resources into a common representation (such as XML) wherever possible. This research-data curation can then be treated as a supplement to text-mining operations. In particular, queries against the full-text publications could be evaluated *also* as queries against the relevant set collection of research data sets.

Wrappers for Network Requests Scientific use of CORD-19 will often require communicating with remote servers. For example, genomics information in the CoVID-19 data sets (such as those mentioned above that are available through Springer Nature) is generally provided in the form of accession numbers which are used to query online genomics services. Similarly, text mining algorithms often rely on dedicated servers to perform Natural Language Processing; these services might take requests in BioC format and respond with CoNLL data. As another case study epidemiological studies of CoVID-19 may need to access APIs or data sets such as the John Hopkins University "dashboard" (see https://coronavirus.jhu.edu/map.html, which is paired with a GIT archive updated almost daily). To reduce the amount of "biolerplate code" which developers need for these networking requirements, an archive's text-mining code could provide a unified framework with which to construct web-API queries, one that could be used across disparate scientific disciplines (genomics, NLP, epidemiology, and so forth).

Many of these limitations observed in CORD-19 reflect the fact that this corpus was prepared as "raw (text) data" without any supporting code. Recent initiatives, such as the Research Object protocol (see [3]) and FAIR ("Findable, Accessible,

Interoperable, Reusable″; see [18]) encourage authors to publish code and data together, so that the computing environment needed to process published data is provided within the data set itself. This Research Object model is usually defined in the context of a single publication, but the paradigm applies equally well to corpora encompassing many single articles. That is, **AXF** is structured so that Research Archives can be designed as higher-scale Research Objests, wherein the document collection is bundled with supporting code and an overall computing and software-development environment. Such archive-specific **SDK**s would include **AXF**-specific code as well as libraries or applications often utilized in the academic disciplines relevant to the archival subject areas. The **AXF** platform especially promotes the design of domain-specific **SDK** which are *standalone* and *self-contained*, with minimal external dependencies. As much as possible, users should not have to install external software to utilize data provided along with an **AXF** repository; instead, the needed data-management tools should be provided in source-code form within the archive itself.

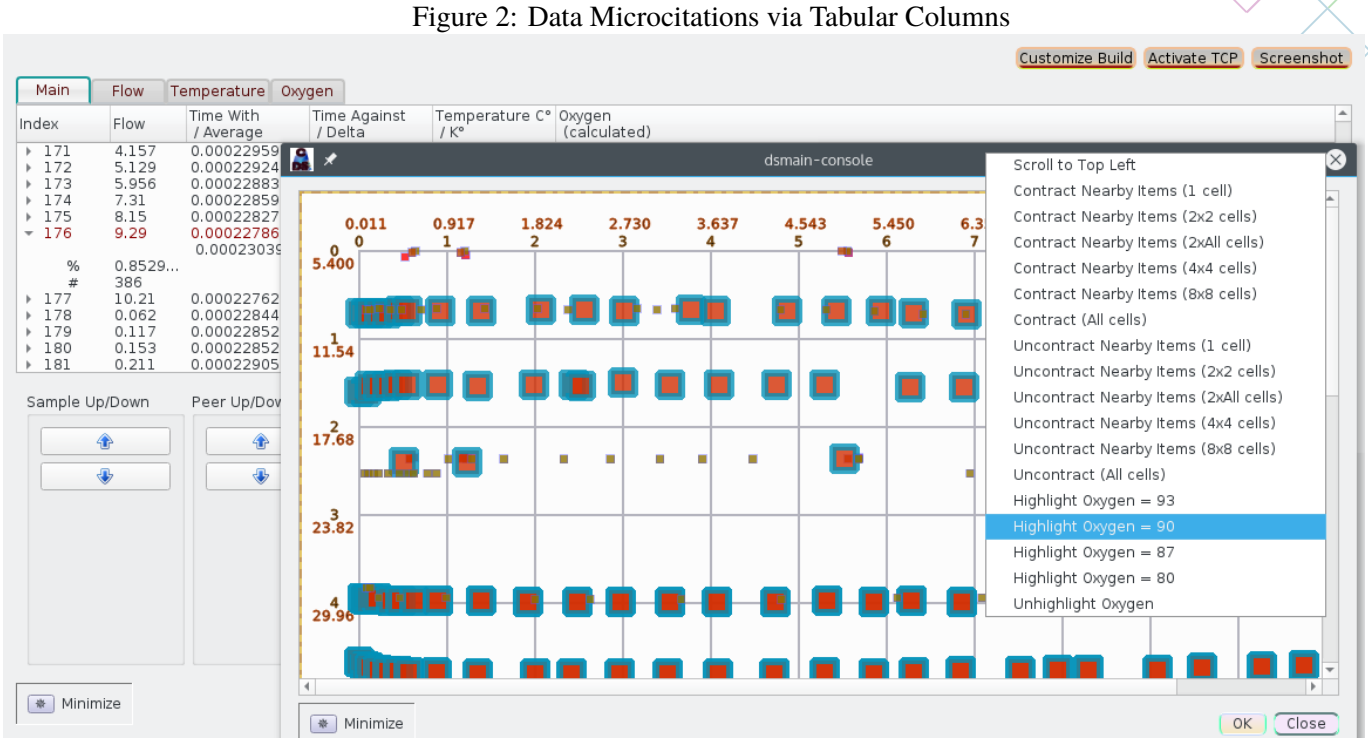
Each **AXF** repository, then, should bundle numerous applications used for database storage, data visualization, and scripting. The goal of this application package would be to provide researchers with a self-contained computing platform optimized for scientific research and findings related to the archived publications. Archival **SDK**s should try to eliminate almost all scenarios where programmers would need to perform a "system install"; for the most part, the entire computing platform (including scripting and database capabilities) should be compiled from source "out-of-the-box". While the actual libraries and applications bundled with an archive would depend on its topical focus, the following is an example of components that would be appropriate in many different **SDK**:

- **XPDF**: A **PDF** viewer for reading full-text articles (augmented with **CORD-19** features, such as integration with biomedical ontologies);
- **Qt**: The **Qt** library is a cross-platform Application-Development framework and **GUI** toolkit commonly used for scientific applications (**XPDF** is one example of a **Qt**-based document viewer). Almost any data set can be accompanied with **Qt** code for data visualization, so that readers would not have to install additional software. For its part, **Qt** can be freely obtained and, once downloaded, resides wholly in its own folder (there is no install step which modifies the user's system); as such, **Qt** along with individual archive **SDK**s function as standalone packages, although optimally the **SDK**s would be updated along with new **Qt** versions.
- AngelScript: An embeddable scripting engine that could be used for analytic processing of data generated by text and data mining operations on **CORD-19** (see [10]);
- WhiteDB: A persistent database engine that supports both relational and **NoSQL**-style architectures (see [14]);
- MeshLab: A general-purpose **3D** graphics viewer;
- LaTeXML: a LaTeX-to-**XML** converter;
- PositLib: a library for use in high-precision computations based on the "Universal Number" format, which is more accurate than traditional floating-point encoding in some scientific contexts (see [8]).

To this list one might add components specific to various scientific fields: **IQmol** for chemistry and molecular biology, for example, or open-source libraries such as EpiFire or Simpact (for Epidemiology), **UDPipe** (for **CoNLL**), and so forth. Here again the priority would be for self-contained components with few external dependencies — particularly libraries programmed in **C** or **C++**, which are the languages best positioned to be a common denominator across diverse research projects (of course, many scientific **C++** libraries have wrappers for languages like **R** or Python that researchers may be more comfortable using). In general, Research Archive code should be (1) *self-contained* (with few or no external dependencies); (2) *transparent* (meaning that all computing operations should be implemented by source code within the bundle that can be examined as code files and within a debugging session); and (3) interactive (meaning that the bundle does not only include raw data but also software to interactively view and manipulate this data). Research Archives which embrace these priorities attempt to provide data visualization, persistence, and analysis through **GUI**, database, and scripting engines that can be embedded as source code in the archive itself.

It is worth noting that a data-mining platform requires *machine-readable* open-access research data (which is a more stringent requirement than simply pairing publications with data that can only be understood by domain-specific software). For example, radiological imaging can be a source of **COVID-19** data insofar as patterns of lung scarring, such as "ground-glass opacity," are a leading indicator of the disease. Consequently, diagnostic images of **COVID-19** patients are a relevant kind of content for inclusion in a **COVID-19** data set (see [15] as a case-study). However, diagnostic images are not in themselves "machine readable." When medical imaging is used in a quantitative context (e.g., applying Machine Learning for diagnostic pathology), it is necessary to perform Image Analysis to convert the raw data — in this case, radiological graphics — into quantitative aggregates. For instance, by using image segmentation to demarcate geometric boundaries one is able to define diagnostically relevant features (such as opacity) represented as a scalar field over the segments. In short, even after research data is openly published, it may be necessary to perform additional analysis on the data for it to be a full-

Figure 2: Data Microcitations via Tabular Columns



fledged component of a machine-readable information space.[2] To deal with this sort of situation, **AXF** equips **SDK**s with a *procedural data-modeling vocabulary* that would both identify the interrelationships between data representations and define the workflows needed to convert research data into machine-readable data sets.

Another concern in developing an integrated Research Arcive data collection is that of indexing documents and research findings for both text mining *and* data mining. In particular, **AXF** introduces a system of *microcitations* that apply to portions of manuscripts *as well as* data sets. In the publishing context, a microcitation is defined as a reference to a partially isolated fragment of a larger document, such as a table or figure illustration, or a sentence or paragraph defining a technical term, or (in mathematics) the statement/proof of a definition, axiom, or theorem. In data publishing, "data citations" are unique references to data sets in their entirety or to their smaller parts. A data microcitation is then a fine-grained reference into a data set. For example, a data microcitation can consist of one column in a spreadsheet, one statistical parameter in a quantitative analysis, or "the precise data records actually used in a study" (in the words adopted by the Federation of Earth Science Information Partners to define microcitations; see [13]). As a concrete example, a concept such as "expiratory flow" appears in **CORD-19** both as a table column in research data and as a medical concept discussed in research papers; a unified microcitation framework should therefore map *expiratory flow* as a keyphrase to both textual locations and data set parameters. Similarly, a concept such as *2'-C-ethynyl* (mentioned earlier, in the context of transcription errors) should be identified both as a phrase in article texts and as a molecular component present within compounds whose scientific properties are investigated through **CORD-19** research data. In so doing, a search for this concept would then trigger both publication and data-set matches at the same time.

Further discussion on data microcitations depends on how data sets are structured, which is addressed in the next section.

III Research Objects and Data Microcitations

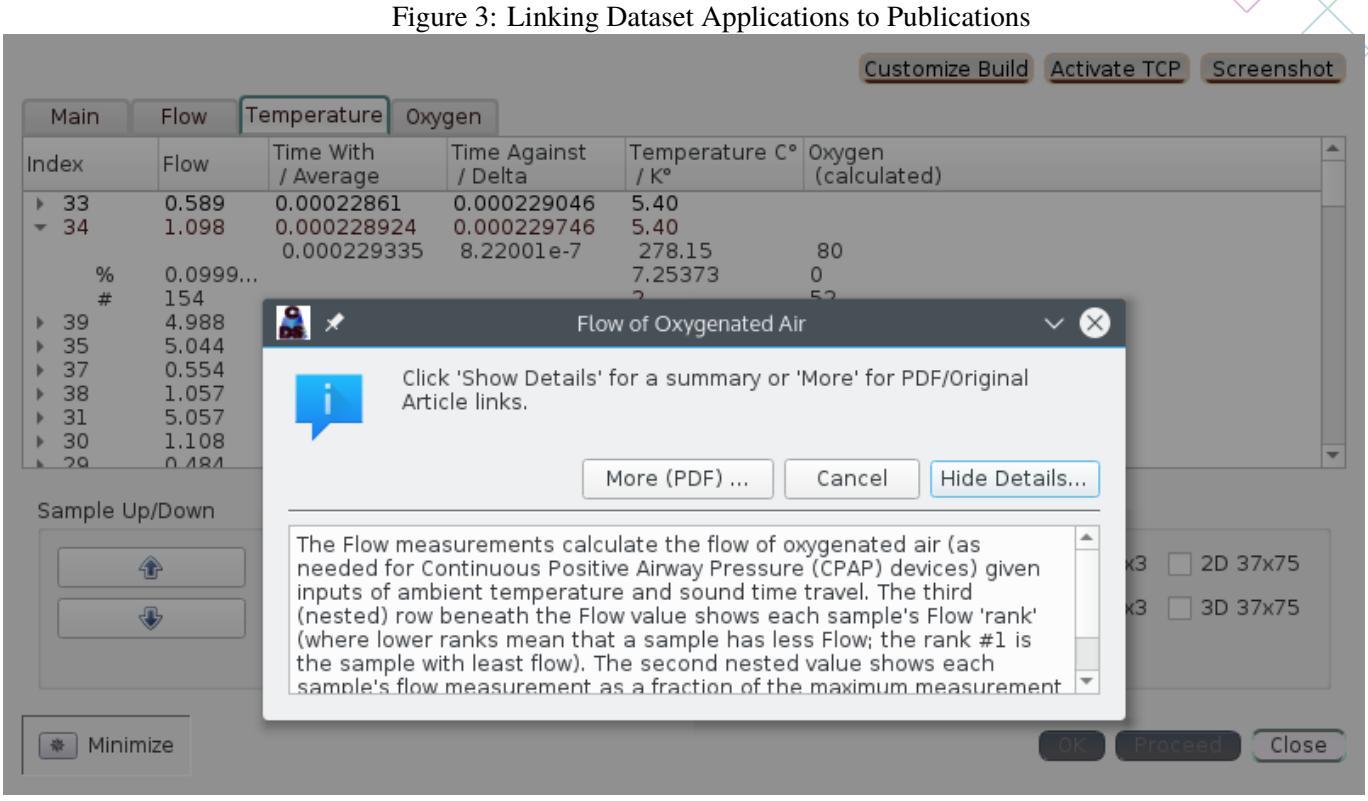
The design of **AXF** assumes that many Research Archives, comprising of multiple publications sharing an academic focus, will also include open-access research data. This means that well-curated archives can prepare data sets for data mining, alongside the preparation of text materials for text mining.

In contrast to text mining, however, it is not feasible, in the general case, to assign one single format (like **AXFD** or **JATS**) for all data sets published within an archive. Precisely how data sets can be annotated depends on the data models, programming languages, and analytic methodologies which they utilize. Because this variability prohibits a single data-annotation protocol from being required, **AXF** adopts a strategy of defining a rigorous protocol for **C++** code bases, which can then be emulated by other languages.

As outlined above, data citations refer to parts within a data set — such as individual data records, but also larger-scale aggregates such as table columns or statistical parameters. The complication when defining data citations is that a concept such as a table column, although it may have an obvious technical status as a discrete conceptual unit from the point of view

[2] This does not mean that diagnostic images (or other graphical data) should not be placed in a data set; only that computational reuse of such data will usually involve certain numeric processing, such as image segmentation. Insofar as this subsequent analysis is performed, the resulting data should wherever possible be added to the underlying image data as a supplement to the data set.

Figure 3: Linking Dataset Applications to Publications



of scientists curating, studying or reusing a data set, does not necessarily correspond to a single coding entity that could be isolated as an annotation target. It is therefore the responsibility of *code base annotations* to provide annotations for computational units — such as data types, procedures, and **GUI** components — that have an annotatable *conceptual* status relative to the data set on which the code operates. Often this will involve mapping one concept to several computational units (for instance, several procedure implementations).

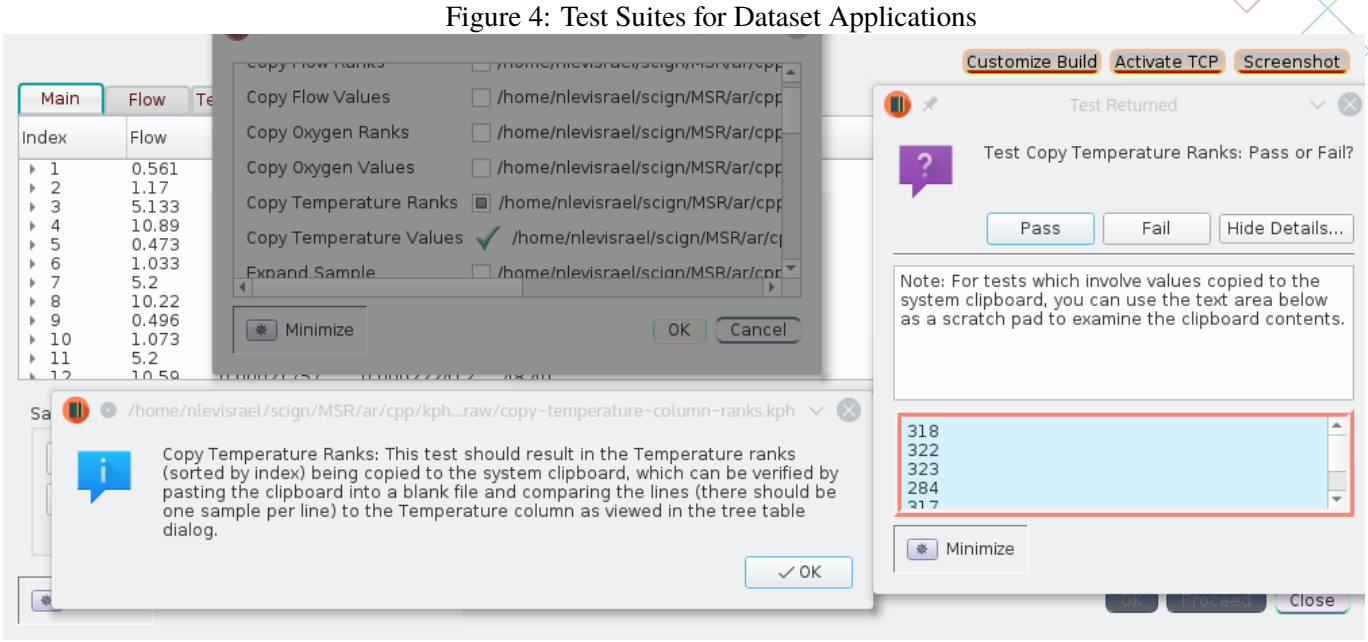
For a concrete example of these points concerning data citations, consider the data set pictured in Figure 2, representing cyber-physical measurements used to calculate oxygenated airflow. The data-set application displays tabular data via a tree widget (which functions as a generalized, multi-scale spreadsheet table), with tabular columns expressing quantities such as air flow and oxygen levels in several formats (raw measures as well as sample rankings and min-max percentages). Conceptually, these columns have distinct methodological roles and therefore can be microcited; indeed, the application links the columns to article text where the corresponding concepts are presented (see Figure 3). However, the implementation does not introduce a distinct **C++** object uniquely designating individual columns. Instead, the individual columns can be annotated in terms of **C++** methods providing column-specific functionality. In the current example, these methods primarily take the form of features linked to context-menu actions (copying column data to the clipboard, sorting data by one column, etc.). In general, rather than a rigid protocol for data-set annotations, **AXF** proposes heuristic guidelines for how best to map programming constructs to scientifically salient data-set concepts.

Defining an annotation schema for data sets can potentially be an organic outgrowth of software-development methodology, such as implementing unit tests. This point is illustrated in Figure 4, which shows a **GUI**-based testing environment for the data set depicted in Figures 3 and 2. For this data set, the context menu actions providing column-specific functionality are also discrete capabilities which can be covered by unit tests, so the set of procedures mapped to the citeable concept correspond with a set of unit-test requirements. In this data set, these procedures are also exposed to scripting engines via the **Qt** meta-object system. In general, there is often a structural correlation between scripting, unit testing, and microcitation, so that an applications' scripting and testing protocol can serve as the basis for annotation schema. For data sets which use in-memory or persistent databases, evaluable queries against these databases provide an additional grounding for annotations. In general, data-annotation should be designed on the basis of a dataset applications' scripting, testing, and/or query-evaluation code. However, this is only a heuristic guideline, and **AXF** does not presuppose any data-annotation scheme *a priori*.

IV Conclusion

AXF uses a two-tier node structure similar to **LAF**; at one level is an extensible text-encoding methodology (disussed in the next paragraph), while a higher level defines annotations in terms of directed hypergraphs. Programmatically, **AXF** aims in the canonical case for a level of detail that is intermediate between linked-data-oriented projects like Web Annotations (which tend to focus mostly on isolatable semantic resources such as citations and named entities) and **NLP**-oriented paradigms such as **LAF**. That is, **AXF** does not natively serialize fine-grained **NLP** data at the level of individual words (the kind of data asserting semantic and morphosyntactic details: lemmatization, Part of Speech, dependency relations, and

Figure 4: Test Suites for Dataset Applications



so forth), although it does support queries which return sentences as (unparsed) word-sequences. On the other hand, **AXF** offers some granular information about small-scale linguistic units, such as the role of non-alphanumeric characters, or **PDF** coordinates of sentence start and end points. In short, **AXF** occupies a unique space in the landscape of annotation tools at the intersection of application-development, **NLP**, and document-preparation requirements.

References

[1] Benjamin Adams and Martin Raubal, "A Metric Conceptual Space Algebra". https://pdfs.semanticscholar.org/521a/cbab9658df27acd9f40bba2b9445f75d681c.pdf

[2] Benjamin Adams and Martin Raubal, "Conceptual Space Markup Language (CSML): Towards the Cognitive Semantic Web". http://idwebhost-202-147.ethz.ch/Publications/RefConferences/ICSC_2009_AdamsRaubal_Camera-FINAL.pdf

[3] Khalid Belhajjame, *et. al.*, "Workflow-centric research objects: First class citizens in scholarly discourse". https://pages.semanticscholar.org/coronavirus-research

[4] Joe Bolt, ıet. al., Interacting Conceptual Spaces I: Grammatical Composition of Concepts. https://arxiv.org/pdf/1703.08314.pdf

[5] "COVID-19 Open Research Dataset (CORD-19)". 2020. Version 2020-03-13. Retrieved from https://pages.semanticscholar.org/coronavirus-research. Accessed 2020-03-20. doi:10.5281/zenodo.3715506 https://pages.semanticscholar.org/coronavirus-research

[6] Ludëk Eyer, *et. al.*, "Nucleoside analogs as a rich source of antiviral agents active against arthropod-borne flaviviruses". https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5890575/

[7] Peter Gärdenfors and Frank Zenker, Theory Change as Dimensional Change: Conceptual Spaces Applied to the Dynamics of Empirical Theories. Synthese 190(6), pp. 1039-1058, 2013. http://lup.lub.lu.se/record/1775234

[8] John Gustafson, "Beating Floating Point at its Own Game: Posit Arithmetic", http://www.johngustafson.net/pdfs/BeatingFloatingPoint.pdf

[9] Nancy Ide and Keith Suderman, "GrAF: A Graph-based Format for Linguistic Annotations", https://www.cs.vassar.edu/~ide/papers/LAW.pdf

[10] Andreas Jönsson, "AngelCode Scripting Library", www.AngelCode.com/AngelScript/

[11] Amy Neustein, *et. al.*, "Application of Text Mining to Biomedical Knowledge Extraction: Analyzing Clinical Narratives and Medical Literature", https://www.researchgate.net/publication/262372604_Application_of_Text_Mining_to_Biomedical_Knowledge_Extraction_Analyzing_Clinical_Narratives_and_Medical_Literature

[12] Tiago Nunes, *et. al.*, "BeCAS: biomedical concept recognition services and visualization". https://www.ncbi.nlm.nih.gov/pubmed/23736528

[13] Mark A. Parsons and Ruth Duerr, "Data Identifiers, Versioning, and Micro-citation", https://www.thelancet.com/action/showPdf?pii=S1473-3099%2820%2930086-4

[14] Enar Reilent, "Whiteboard Architecture for the Multi-agent Sensor Systems", https://www.thelancet.com/action/showPdf?pii=S1473-3099%2820%2930086-4

[15] Heshui Shi, *et. al.*, "Radiological findings from 81 patients with COVID-19 pneumonia in Wuhan, China: a descriptive study". https://www.thelancet.com/action/showPdf?pii=S1473-3099%2820%2930086-4

[16] Dietrich Rebholz-Schuhman, *et. al.*, "IeXML: towards an annotation framework for biomedical semantic types enabling inter-operability of text processing modules". https://www.semanticscholar.org/paper/IeXML%3A-towards-an-annotation-framework-for-semantic-Rebholz-Schuhmann-Kirsch/1d72a56b6576117c62f388a5f2193965e4c7e293

[17] C. J. Rupp, *et. al.*, "Flexible Interfaces in the Application of Language Technology to an eScience Corpus". https://www.cl.cam.ac.uk/~sht25/papers/Rupp_et_al.pdf

[18] Alina Trifan and José Luís Oliveira, "FAIRness in Biomedical Data Discovery". https://www.researchgate.net/publication/331775411_FAIRness_in_Biomedical_Data_Discovery

[19] Eric Winsberg, *Science in the Age of Computer Simulation*. https://www.press.uchicago.edu/ucp/books/book/chicago/S/bo9003670.html