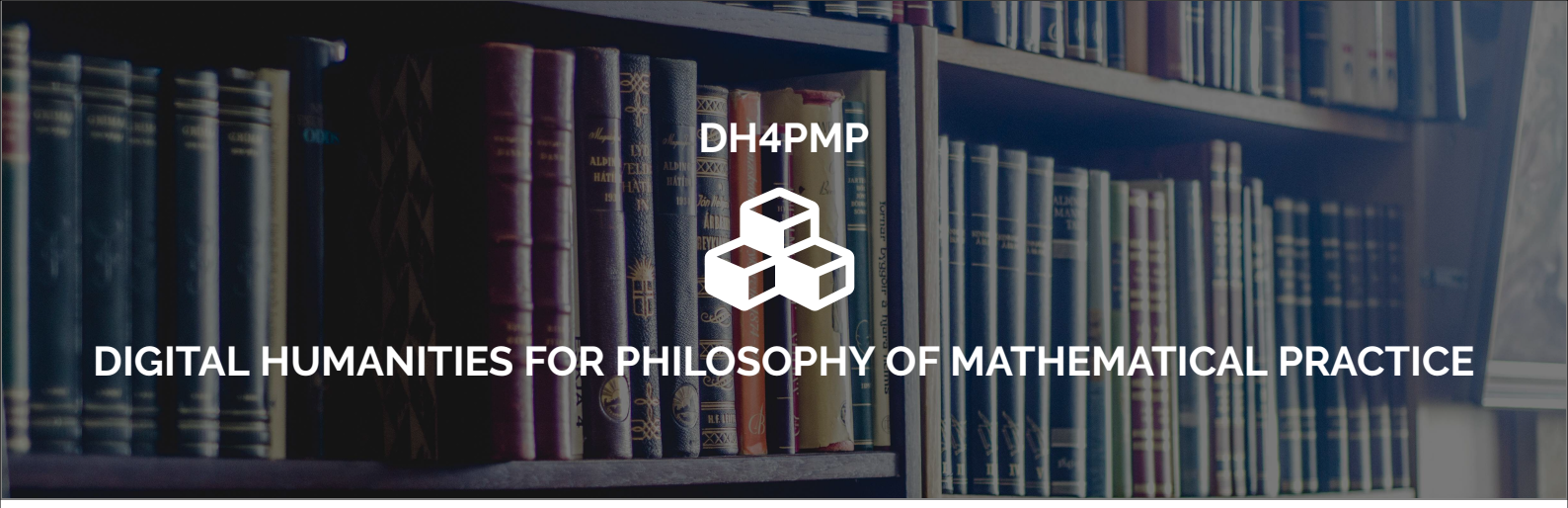


The DH4PMP LaTeX-pipeline for processing arXiv sources

Henrik Kragh Sørensen

Digital Humanities for Philosophy of Mathematical Practice, Section for History and Philosophy of Science, Department of Science Education, University of Copenhagen, Denmark. henrik.kragh@ind.ku.dk.

February 8, 2022

Abstract

The purpose of the DH4PMP LaTeX-pipeline is to provide a robust way of processing LaTeX source into an abstract representation that preserves the features that interest us. This note documents the purpose, construction, and affordances of the pipeline, while Sørensen (2021) documents the present stage of running the pipeline and internal assessments of the quality of data, including descriptive statistics. Additional papers such as SÃ¸rensen (2021b) report on the construction of particular datasets from the pipeline and various external validations of the data.

1 Introduction

The arXiv poses a rich corpus of research prepublications which can be utilized to explore philosophical questions. Some such questions might be answerable just from the posted PDF files, but the fact the vast majority of papers in the arXiv are written in LaTeX (especially in the mathematics and computer science categories) offers additional possibilities (Mejia-Ramos et al., 2019). LaTeX is best understood as a programming language producing typeset text and mathematics. Thus the structure of the LaTeX sources provide additional structural detail to the otherwise hierarchically organized linear text. Thus to understand and exploit the structural information, we have set up a pipeline which inputs LaTeX source and provides a representation of a reduced LaTeX source, split into so-called *contexts*. Using this pipeline, it is possible to compare e.g. word prevalence between the *theorem context* and the *proof context*.

2 LaTeX: structured mathematical typesetting

LaTeX is a further development of TeX which was designed and developed by Donald Knuth and first released in 1978 (Knuth, 1986). Knuth's TeX was designed to typeset complicated

Henrik Kragh SÃ¸rensen (May 24, 2021a). *The DH4PMP LaTeX-pipeline for processing arXiv sources.* DH4PMP Work-in-Progress & Proof-of-Concept Series 12. Version 1. 18 pp. URL: https://www.erda.dk/vgrid/DH4PMP/wippoc/paper12.pdf.

mathematical formulas, and it holds tremendous power but is also (now) considered very difficult to master. Instead, mathematicians, computer scientists and many other academics have adopted LaTeX which was written during the early 1980s by Leslie Lamport as an extension of TeX (Lamport, 1994). LaTeX forms the core of a very complex and evolving software system to which new functionality is constantly added through the publication of *packages* on the open-source repository `ctan.org`. In addition, LaTeX has been compiled and distributed for all the major operating systems, and document editors have been developed to ease the composition of text and online collaboration.[1]

2.1 Commands and environments

The power of LaTeX is that it is actually a *programming language* intended for producing typeset text. The central programming constructs are *commands* and (as extensions thereof) *environments*. A command (or *macro*) has a name, is prefixed by a backslash, can take arguments (either optional or mandatory), and expands to its definition through the LaTeX compilation process. An environment is technically a set of combined commands to *begin* and *end* the environment. These commands can be taylored to specific applications, and are often used to scope together elements that belong together, for instance in a list or a theorem declaration.

A lot more can be said about LaTeX formatting and programming (see Lamport, 1994), but here it suffices to know that LaTeX:

- is a widely used system for typesetting articles and books within certain academic disciplines including mathematics, computer science, and physics,
- is a programming language in which *source files* are written as ordinary text files which are *compiled* into the output (a PDF-file) by running the LaTeX compiler either from a command prompt or from a dedicated editor,
- has the potential to organize the input text in a hierarchy using sectioning commands as well as in a structure using environments,
- is extremely customizable through the use of *packages* and can include input from secondary sources such as code listings or tables,
- is able to handle comments that do not produce output but can serve as notes, messages, outlines etc. between the authors.

2.2 Accessing arXiv data

Our raw data consists of essentially three distinct sets of files:

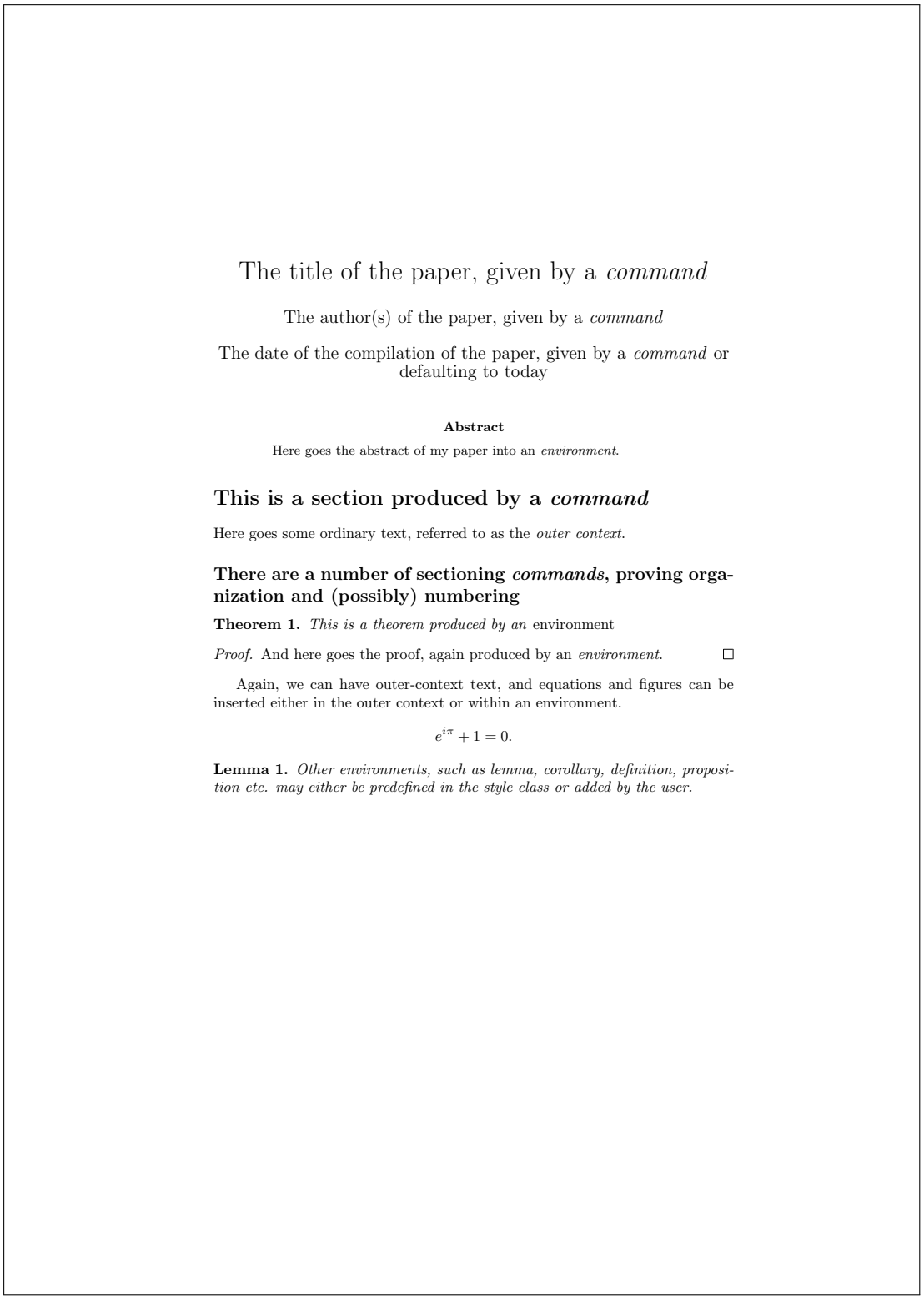
1. A JSON file containing the metadata of all articles published on the arXiv. This file is deposited from the arXiv to Kaggle and scheduled for monthly updates.[2]
2. A collection of PDF files obtained from Kaggle via a Google cloud storage.
3. A collection of TAR files containing the LaTeX source of articles. These are obtained from the AWS[3] for which there is a requester fee.

[1] The LaTeX software system is robust in the sense that its produced output is highly platform independent, whereas small changes in the source may actually lead to substantial differences in the output.

[2] See `https://www.kaggle.com/Cornell-University/arxiv`. To download the metadata from Kaggle, you will need to create a free account.

[3] See `https://arxiv.org/help/bulk_data_s3`.

Henrik Kragh SÃ¸rensen (May 24, 2021a). *The DH4PMP LaTeX-pipeline for processing arXiv sources.* DH4PMP Work-in-Progress & Proof-of-Concept Series 12. Version 1. 18 pp. URL: `https://www.erda.dk/vgrid/DH4PMP/wippoc/paper12.pdf`.



The title of the paper, given by a *command*

The author(s) of the paper, given by a *command*

The date of the compilation of the paper, given by a *command* or defaulting to today

Abstract

Here goes the abstract of my paper into an *environment*.

This is a section produced by a *command*

Here goes some ordinary text, referred to as the *outer context*.

There are a number of sectioning *commands*, proving organization and (possibly) numbering

Theorem 1. *This is a theorem produced by an* environment

Proof. And here goes the proof, again produced by an *environment*. □

Again, we can have outer-context text, and equations and figures can be inserted either in the outer context or within an environment.

$$e^{i\pi} + 1 = 0.$$

Lemma 1. *Other environments, such as lemma, corollary, definition, proposition etc. may either be predefined in the style class or added by the user.*

Figure 1: The output of the a simple LaTeX source, indicating various structural and hierarchical elements.

Henrik Kragh SÃ¸rensen (May 24, 2021a). *The DH4PMP LaTeX-pipeline for processing arXiv sources.* DH4PMP Work-in-Progress & Proof-of-Concept Series 12. Version 1. 18 pp. URL: https://www.erda.dk/vgrid/DH4PMP/wippoc/paper12.pdf.

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage[latin1]{inputenc}

\usepackage{amsthm,amsmath}
\newtheorem{theorem}{Theorem}
\newtheorem{lemma}{Lemma}

\begin{document}

\author{The author(s) of the paper, given by a \emph{command}}
\title{The title of the paper, given by a \emph{command}}
\date{The date of the compilation of the paper, given by a
  \emph{command} or defaulting to today}

\maketitle

\thispagestyle{empty}

\begin{abstract}
  Here goes the abstract of my paper into an \emph{environment}.
\end{abstract}

\section*{This is a section produced by a \emph{command}}

Here goes some ordinary text, referred to as the \emph{outer context}.

\subsection*{There are a number of sectioning \emph{commands}, proving
  organization and (possibly) numbering}

\begin{theorem}
  This is a theorem produced by an \emph{environment}
\end{theorem}

% Some lines can be commented out and thus produce output while still
% being part of the source

\begin{proof}
  And here goes the proof, again produced by an \emph{environment}.
\end{proof}

Again, we can have outer-context text, and equations and figures can
be inserted either in the outer context or within an environment.
\begin{align*}
  e^{i\pi}+1  = 0.
\end{align*}

\begin{lemma}
  Other environments, such as lemma, corollary, definition,
  proposition etc. may either be predefined in the style class or
  added by the user.
\end{lemma}

\end{document}
```

Figure 2: The LaTeX source responsible for producing the output in Figure 1.

Henrik Kragh SÃ¸rensen (May 24, 2021a). *The DH4PMP LaTeX-pipeline for processing arXiv sources.* DH4PMP Work-in-Progress & Proof-of-Concept Series 12. Version 1. 18 pp. URL: https://www.erda.dk/vgrid/DH4PMP/wippoc/paper12.pdf.

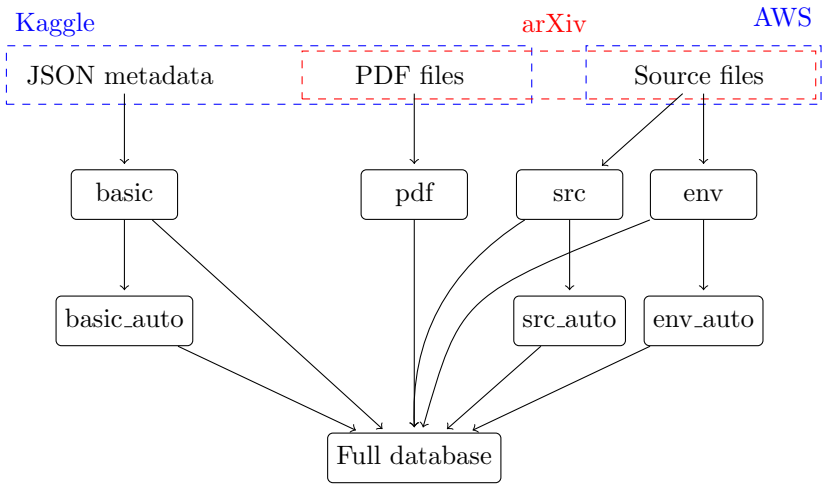


Figure 3: Schematic overview of the data processing.

From Kaggle, we first obtain a JSON file with metadata on all articles in the arXiv. These are then filtered to include only articles assigned to (at least) one mathematical category math.*.

Using `gsutil`, we can download PDF files of the articles from, say, September 2020 by using the command:[4]

```
gsutil cp -m -r gs://arxiv-dataset/arxiv/arxiv/2009* .
```

Here, we meet the arXiv way of indicating production time: the yymm format: two digits to indicate the year (after 2000) and two digits to indicate the month (0-prefixed).If we are only interested in specific articles (say the the mathematical ones), we filter which TAR-files to download based on the JSON metadata.

At the AWS, article sources are stored in chronological collections, so given the *manifest* we identify those collections that are relevant for us, download those and extract the relevant individual article sources. Each source for an arXiv article is collected in a (gzipped) TAR-file, which will typically contain LaTeXsource as well as optional figures, bibliography files etc.

In our 'backend', we can prioritize downloads from AWS to focus on specific months or specific categories; ultimately it is our intention to keep a full duplicate of the arXiv on AWS.

The result of this download process is (almost) identical to the result of downloading each of these article sources individually from arXiv, but in accordance with arXiv policy, we wish to help curtail the traffic to the site. However, individual article sources can later be supplemented by download from the arXiv. The only difference between the two approaches is that articles may be *updated* which will reflect in the directly downloaded source and PDF-files but may not always be incorporated in the TAR-file.

Each arXiv article is identified by its arxivID, a number of the format yymm.ddddd where yymm is the time indication of initial upload, and the digits are the separator indicates a running index (0-prefixed to either 4 or 5 places).

We process information from the raw data sources into *pandas* dataframes through a series of import routines written in *python* and constituting the DH4PMP LaTeX-pipeline.

2.3 Processing LaTeX sources

Considered as a programming language, the typical way to process LaTeX sources would be to build 1) a generic parser which could parse LaTeX source into an abstract representation and 2) a

[4]See `https://www.kaggle.com/Cornell-University/arxiv/discussion/200472`.

Henrik Kragh SÃ¸rensen (May 24, 2021a). *The DH4PMP LaTeX-pipeline for processing arXiv sources.* DH4PMP Work-in-Progress & Proof-of-Concept Series 12. Version 1. 18 pp. URL: `https://www.erda.dk/vgrid/DH4PMP/wippoc/paper12.pdf`.

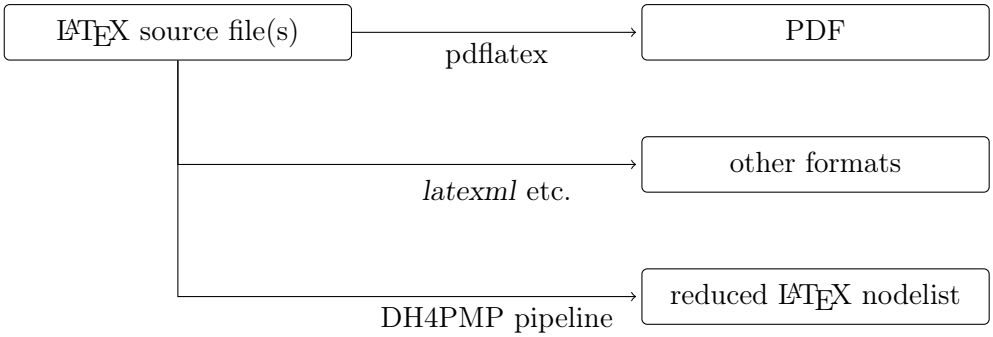


Figure 4: Processing LaTeX sources.

specific backend for producing the desired output (PDF or other formats); see Figure 4. However, because LaTeX is such a complex and powerful language (it is 'Turing complete'), no complete grammar was ever defined (Beebe, 2004, p. 15). Instead, the full description of the LaTeX 'programming language' is the implementation of TeX as well as the adjunction of particular constructs in LaTeX and the full set of packages extending that even further. And since Knuth's original TeX implementation is written in a presently almost obsolete programming language (*WEB*), the typical contemporary distributions are actually made by compiling these into *C* using *Web2c* and then compiling the to the desired platform. Thus, accessing the intermediate abstract representations of the typeset is difficult, and instead we are often left to implement parsers and backends for a (small) subset of the entire expressive power of TeX.

A number of processors for TeX and LaTeX which provide the kind of structured representations that we want have been developed, including *latexML* and *pandoc* (Miller and Ginev, 2020) which are able to convert large subsets of LaTeX into *XML* or *HTML* (or a number of other markup formats) which could, in turn, be parsed with relative ease. However, to have control over the entire pipeline and have it implemented in *python*, we chose to build our own pipeline on top of the module *pylatexenc* (Faist, 2019).

3 The DH4PMP LaTeX-pipeline

The purpose of the LaTeX-pipeline, that we describe in the following, is to provide a robust way of processing LaTeX source into an abstract representation that preserves the features that interest us. The core of the pipeline is our adaptation of the *pylatexenc* parser (walker) and processor (Faist, 2019). However, to overcome some important shortcomings of *pylatexenc*, we have had to add an additional level of preprocessing.

The pipeline consists of the following steps:

1. File access (depending on application area such as arXiv or MathSciNet parsing). For arXiv parsing, we rely on our class ArxivSourceManager.

2. Preprocessor, working on an individual file, doing the following:

   a) Removing comments

   b) Extracting certain information from the source files

   c) Resolving static conditionals

   d) Making adjustments to the source (primarily to balance tokens) before it can be submitted to *pylatexenc*

3. Parser, based on *pylatexenc*, producing a nodelist and implementing persistent serialization for subsequent recall.

Henrik Kragh SÃ¸rensen (May 24, 2021a). *The DH4PMP LaTeX-pipeline for processing arXiv sources.* DH4PMP Work-in-Progress & Proof-of-Concept Series 12. Version 1. 18 pp. URL: https://www.erda.dk/vgrid/DH4PMP/wippoc/paper12.pdf.

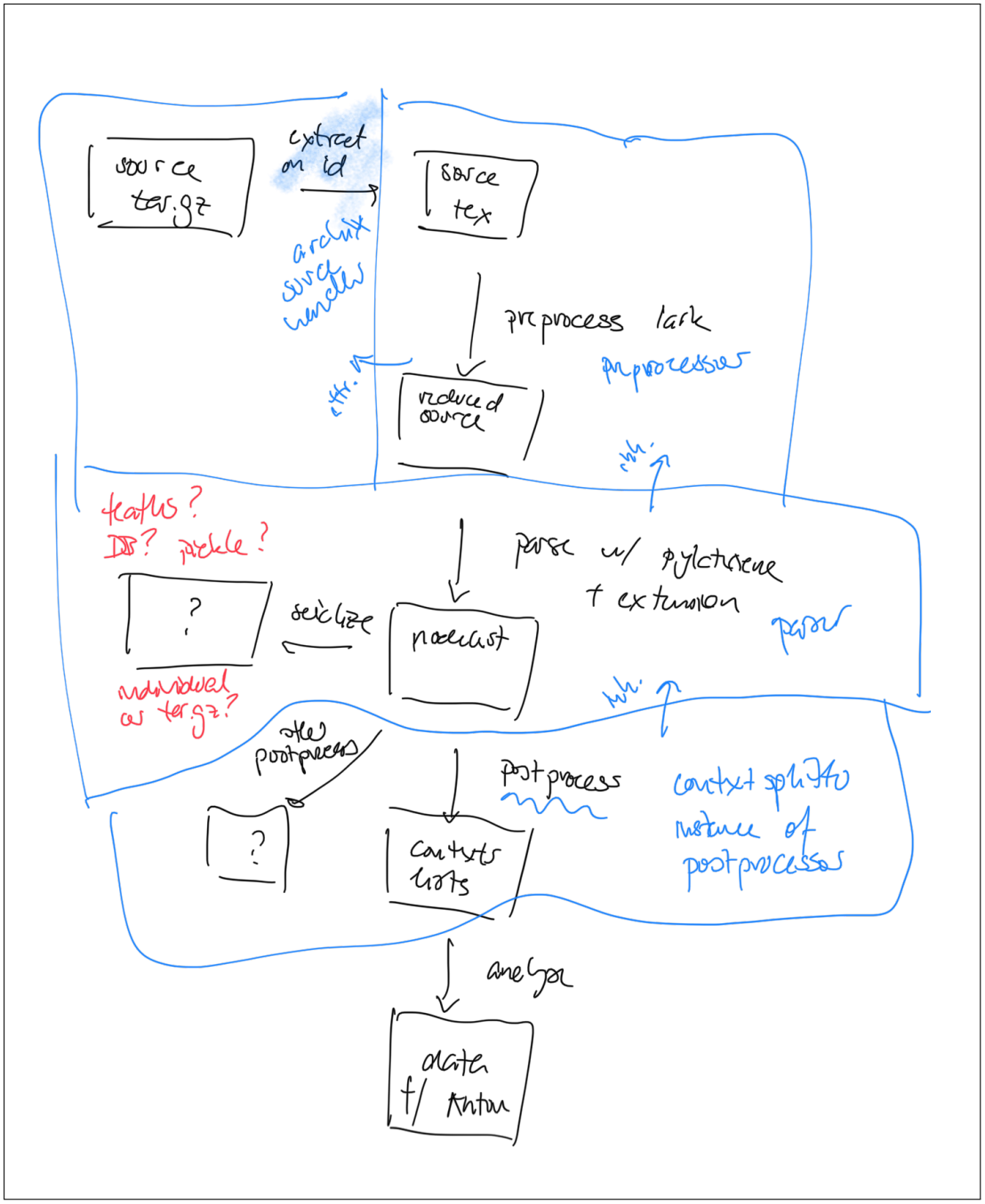


Figure 5: Outline of the pipeline.

4. Postprocessors (depending on application) such as context splitters or linguistic analyses.

The pipeline is implemented as inheritance of classes for steps 3–4, and attribute classes for steps 1 and 2.

4 Preprocessor

The preprocssor uses relatively simple tools such as regular expressions and simple *lark*-based parsers (Shinan, 2021) to prepare the invidual LaTeX-files for processing by *pylatexenc*. In particular, the preprocessor performs the following tasks:

1. Remove comments from the LaTeX source

Henrik Kragh SÃ¸rensen (May 24, 2021a). *The DH4PMP LaTeX-pipeline for processing arXiv sources.* DH4PMP Work-in-Progress & Proof-of-Concept Series 12. Version 1. 18 pp. URL: https://www.erda.dk/vgrid/DH4PMP/wippoc/paper12.pdf.

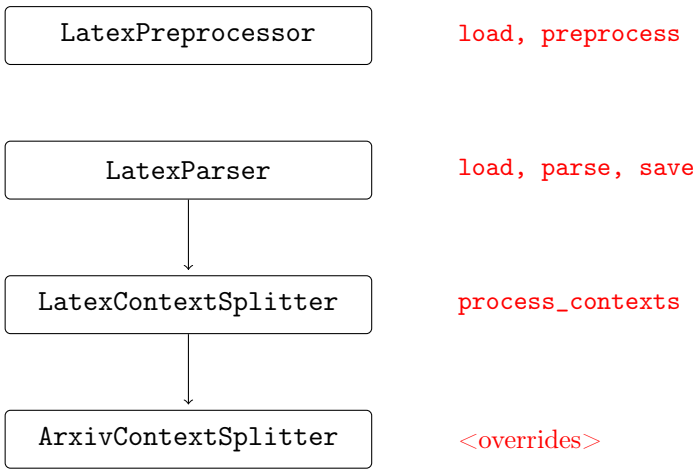


Figure 6: Inheritance structure.

2. Remove conditionals from the LaTeX source
3. Extract simple macro definitions to be expanded in step 4 and process environment constructions in the LaTeX source; see Figure 7
4. Perform simple macro expansion
5. Remove math constructs from the LaTeX source.

A central aspect of the preprocessor is circumventing the nested balancing of braces and math environments. For instance, it is legal (and frequent in the arXiv) to write

$$\$x\backslash in[0,1)\$$$

which contains an unbalanced bracket.[5] Similarly, it is legal to shorthand math environments, writing e.g.

$$\texttt{\textbackslash newcommand\textbackslash ba\{\textbackslash begin\{align*\}\}},$$

which includes an unbalanced opening of a math environment, albeit within a nested (and balanced) set of braces.

Thus, to consistently process such issues, we implemented a layered number of runs of parsers (multiple parse runs) to process LaTeX files.[6]

Some of the grammars, in particular parts of `gobble-conditionals.lark` and of `gobble-math.lark` are fully or partially auto-generated from other configuration files. Then, depending on the run of the preprocessor, *lark* rules can be given different priorities to match the kind of lexing that is currently going on.

4.1 Stages of the preprocessor

The preprocessor works through a number of *stages* (sometimes referred to as *runs* or *jobs*), which constitute the preprocessor pipeline:

[5] Such legal strings are exactly what messes with the *pylatexenc*, giving rise to the need for a preprocessor to handle them before passing the string on to *pylatexenc*.

[6] Obviously, we could also have opted for a different model with increasing handling of special cases. This might still be a preferable option for some specific cases, and can then be implemented as a separate run of the preprocessor.

Henrik Kragh SÃ¸rensen (May 24, 2021a). *The DH4PMP LaTeX-pipeline for processing arXiv sources.* DH4PMP Work-in-Progress & Proof-of-Concept Series 12. Version 1. 18 pp. URL: https://www.erda.dk/vgrid/DH4PMP/wippoc/paper12.pdf.

```
elems:      (newcmd | newcmdtwo | newenv | docclass | inputfile  | ref
   | cite | includegraphics| letstmt | elem)*
// | ref | includegraphics | cite

?elem.999:        string
            | SEP

//WRONG.999: "\\include" /[a-zA-Z]+/

newcmd.100:     NEWCMD (MACRONAME|arg) [optarg [optarg]] (
   MACRONAME|exparg)
//            |   DEF DEFMACRONAME defopts ["="] [SEP] (
   MACRONAME|exparg)
            | DEF DEFMACRONAME ["="] [SEP] (MACRONAME|exparg)

//DEFOPTS.100:    /#\d/ | /##\d/ | /\[#\d\]/
//DEFSEP.100:     /[\(\):\- ;,<>\/]/ | "\\@nil"
//defopts.100:    ([DEFSEP+] DEFOPTS)* [DEFSEP+]
//DEFMACRONAME.100:   (/\\[^ #\{\[]+/)
DEFMACRONAME.100:   /\\[^\{]+/

newcmdtwo.101:  NEWCMDTWO (MACRONAME|arg) [optarg [optarg [optarg]]] (
   MACRONAME|exparg)

newenv.100:     NEWENV arg [optarg [optarg]] exparg exparg [SEP]

docclass.100:  DOCCLASS [optarg] arg

inputfile.100:          INPUTFILE [SEP] (arg | CHARS+)
ref.100:                REF [SEP] arg | REF SEP | REF MACRONAME
includegraphics.100:    INCLUDEGRAPHICS [SEP] [optarg+] arg
cite.100:               CITE [SEP] [optarg+] arg

CITE.20:               /\\cite[pt]?(?![a-zA-Z~])/ ["*"]
INCLUDEGRAPHICS.20:    "\\includegraphics" ["*"]
REF.20:                (/\\ref(?![a-zA-Z~])/ | /\\eqref(?![a-zA-Z~])/ |
    /\\autoref(?![a-zA-Z~])/) ["*"]
INPUTFILE.20:          (/\\input(?![a-zA-Z~])/ | /\\include(?![a-zA-Z
   ~])/)

//COMMANDTOKENS.20:   REF | CITE | INCLUDEGRAPHICS | INPUTFILE

NEWCMD.100:     ("\\newcommand" ["x"] ["*"] | "\\providecommand" ["x"]
   ["*"] | "\\renewcommand" ["x"] ["*"]) [SEP]
NEWCMDTWO.101:  ("\\newcommandtwoopt") ["*"] [SEP]
NEWENV.100:     ("\\newenvironment" ["*"] | "\\renewenvironment" ["*"])
    [SEP]
DOCCLASS.100:   "\\documentclass" [SEP]

DEF.999:    /\\def(?![a-zA-Z~])/ [SEP]

letstmt.200:    LET (MACRONAME | CITE | REF) ["="] (MACRONAME | RBQ |
   LBQ)
LET.200:     /\\let(?![a-zA-Z~])/ [SEP]

?string.10:    (CHARS|MACRONAME|LBQ|RBQ|LBR|RBR)+
CHARS.0:         /[^\[\]\{\}]/
?sbstring:         (CHARS+|MACRONAME|SEP|INPUTFILE) | arg | optarg
              |REF|CITE
?brstring:         (CHARS+|MACRONAME|SEP|INPUTFILE|LBQ|RBQ) | arg |
   newcmd   |REF|CITE

LBQ.50: "["
RBQ.50: "]"
```

`gobble-comments` removes comments in LaTeX source, preserving the structure.

`gobble-conditionals` resolves static conditionals (i.e. `\iffalse` and `\iftrue`). To process other static conditional constructs which also can contain `\else` and `\fi` (e.g. such as provided by LaTeX classes and packages), it is (presently) necessary to supply a list of such conditionals to the processor. To aid in this, the processor picks out `\newif` constructs in the file, but these were found to account for only a small portion of the static conditionals.

`macro-extract` extracts macro and environment definitions (such as arising from `\newcommand` or `\newenvironment`). For macros, it stores the literal definition for later. For for environments, it simply creates an empty environment, since we are only interested in the structure of the source, not the actually formatting of environments, including their nesting.

`macro-expansion` performs literal expansion of previoulsy defined macros (picked up through macro-extract), resulting in a one-level literal expansion in each individual file.

`gobble-math` removes all mathematical constructs, potentially replacing them with a customizable placeholder (not yet implemented).

The first of these can be (and is) implemented using regular expressions in *python*, but the remaining stages are all implemented using a (rather) simple LALR(1) parser constructed from EBNF-form through *lark*.[7]

Each *stage* of the preprocessor (after the first one) consists of a standard parser structure:

1. A *parser* which takes the input string and produces an AST. Here, we use the default 'contextual' parser provided in *lark*.

2. A *transformer* which makes transformations to the AST.

3. An *interpreter* which takes the AST and produces the output, in our case a new (valid) LaTeX string with certain (irrelevant) problematic constructs processed (typically removed).

As a matter of convenience, we use the same transformer and interpreter for all stages in the pipeline.

5 Parser

The purpose of the parser is the more detailed processing of the LaTeX source into a *nodelist* (an AST) that picks up on the important structural information in the sources. This is (at the moment) achieved through the customizable *pylatexenc* interface. The results of the parser are stored in a database structure for which nodelists for invidual papers can easily be retrieved.

As part of the parser, the following steps are performed:

1. The list of filenames is retrieved from the source TAR.

2. The *main file* is identified as one that includes a `\documentclass` macro, and this information was picked up in the preprocessor. If more than one file satisfies this, the paper is marked for follow-up by human intervention.

3. Each file in the TAR is extracted and stored based on its relative path. The contents of auxiliary files are then substituted for relevant `\input` commands in the main file, iterated until a stable text is reached.

[7] It is possible that some stages can be combined, but they are split for purposes of debugging the complex process and for the possibility to assign different priorities to the same syntactic constructs at different stages.

Henrik Kragh SÃ¸rensen (May 24, 2021a). *The DH4PMP LaTeX-pipeline for processing arXiv sources.* DH4PMP Work-in-Progress & Proof-of-Concept Series 12. Version 1. 18 pp. URL: https://www.erda.dk/vgrid/DH4PMP/wippoc/paper12.pdf.

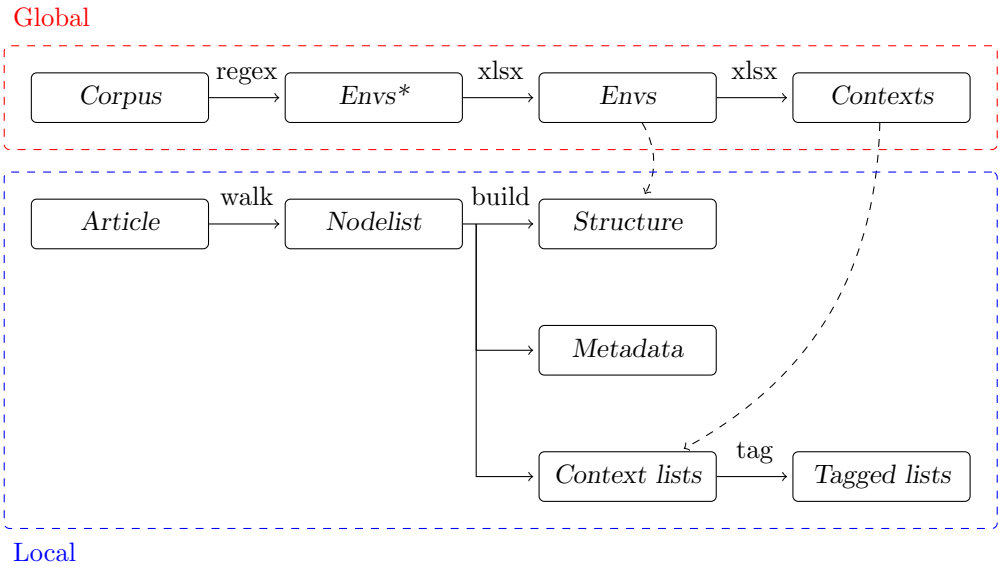


Figure 8: Pipeline of processing LaTeX source for structural and textual information.

4. At this point, the preprocessor is run, preparing a revised version of the expanded main file that can be processed by our customized *pylatexenc* parser. In the preprocessor, numerous structures are extracted, in particular citations to the bibliography, internal references in the document, inclusion of graphics files, and inline and displayed mathematics.

5. In the next step, the customized *pylatexenc* parser is run in the resulting text. As part of this phase, various environments are picked out and stored as *contexts* according to the many-to-one mapping described below.

6 Postprocessors

The purpose of a postprocessor is to turn the nodelist produced in the parser into formats suited for particular research questions. This could involve adding PoS (position-of-speech) tagging to identify e.g. nouns or verbs or associated clusters. The (currently) most important postprocssor splits the LaTeX source text into *contexts* defined by a many-to-one relation between the environments used in the LaTeX source and a set of predefined contexts (section 6.1).

6.1 Context splitting

The pipeline, see Figure 8 consists of a 'global' part which allows for configuration, and a 'local' part in which each article is processed individually based on the global configuration. In order to process the structure of the LaTeX source, we first pick out *all* environments used in the corpus of sources by a regular expression search for the corresponding LaTeX construct, i.e. `\begin{`. This provides us with a list of environments and their respective frequencies. This list is then processed by hand as an Excel sheet where a partial mapping is established which groups variant forms of similar environments together; for instance the construct `\begin{thm} ... \end{thm}` is equated with the `theorem` environment.

A *context* is defined through two sets of many-to-one mappings that combine environments from the LaTeX sources into a small number of seperate structural contexts.

The output of this procedure is, for each LaTeX *mainfile* (i.e. a file containing a \documentclass command and thus signifying a compilable LaTeX source), the following:

Henrik Kragh SÃ¸rensen (May 24, 2021a). *The DH4PMP LaTeX-pipeline for processing arXiv sources.* DH4PMP Work-in-Progress & Proof-of-Concept Series 12. Version 1. 18 pp. URL: https://www.erda.dk/vgrid/DH4PMP/wippoc/paper12.pdf.

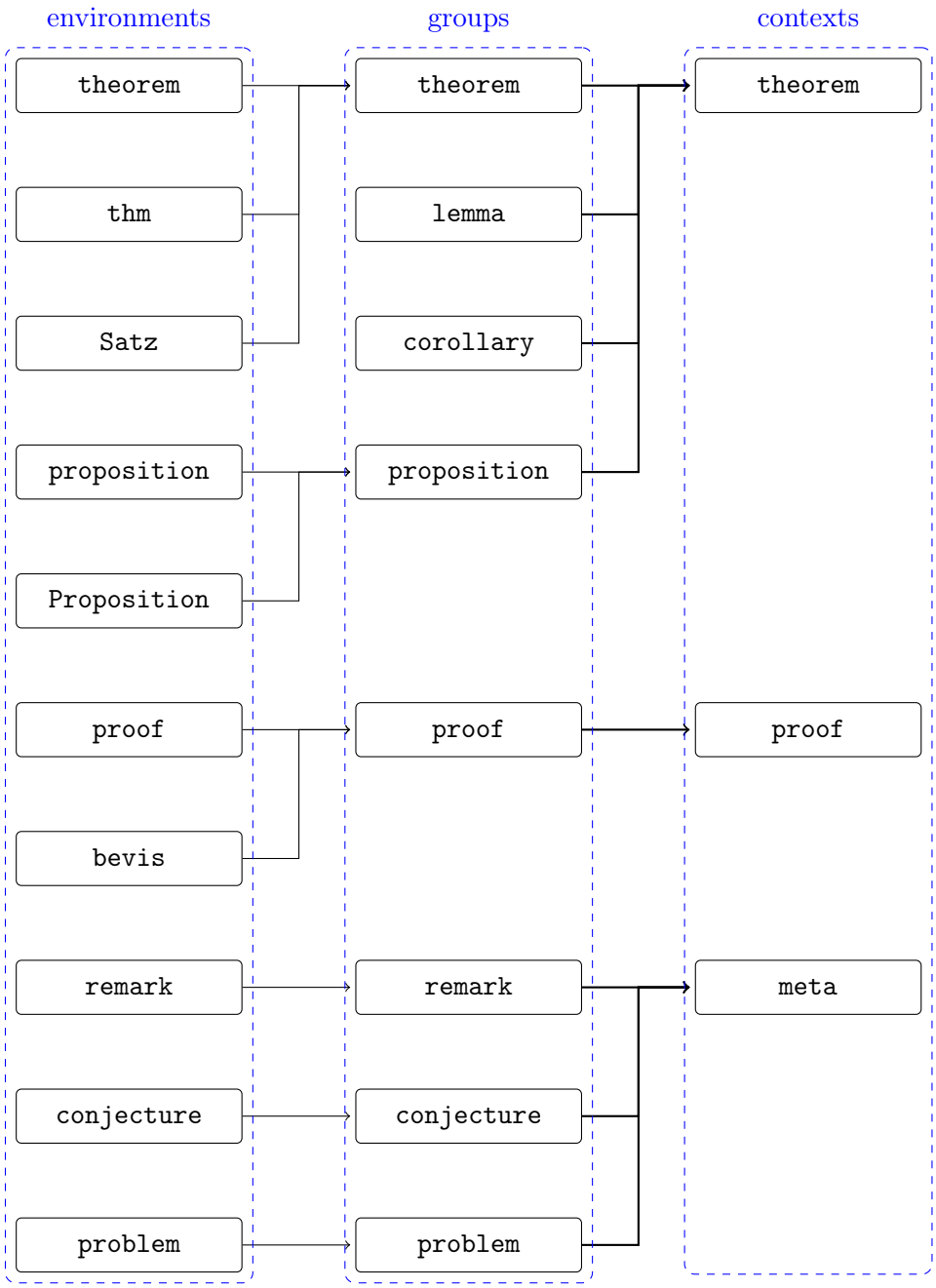


Figure 9: Small extract from the two many-to-one maps which define contexts from environment names. The first one is intended to capture synonyms, abbreviations, and translations. The second one maps these into contexts.

Henrik Kragh SÃ¸rensen (May 24, 2021a). *The DH4PMP LaTeX-pipeline for processing arXiv sources.* DH4PMP Work-in-Progress & Proof-of-Concept Series 12. Version 1. 18 pp. URL: https://www.erda.dk/vgrid/DH4PMP/wippoc/paper12.pdf.

- For each context, a list of lists of paragraphs. This lists are true to the sequence in which the paragraphs occur in the source.
- A list of structural elements combined with the level of nesting for the particular element.
- The default main contexts currently implemented are *definition*, *theorem*, *proof*, and *meta*. Additional contexts are also implemented, primarily to capture one-off environments such as *abstract* and *acknowledgements*. The two sets of mappings are defined in the file environments.xlsx.
- All environments which are not mapped as in Figure 9 are assigned to the *other* context, whereas the *outer* context contains all level-0 text, i.e. text not embedded in an environment. A few groups are not matched to a context; this currently applies to *algorithm*, *figure*, *table*, *diagram*, and *equation* groups.
- For each of the elements `CITE_EXTERNAL`, `REF_INTERNAL`, `GRAPHICS_FILE`, and `MATH_DISPLAY`, the occurance in the main file is substituted with a tag of the form `MATH_DISPLAY(1)`, which points to a list of extracted text from the original occurance. To allow reversing this extraction or studying particular elements such as displayed mathematics, the resulting data also contain lists of these elements. If one is to run linguistic analyses, it may be preferable to remove the indexes, e.g. running

  ```
  re.sub(r'(MATH_DISPLAY)\(\d+\)', \g<1>, x)
  ```

  on the text `x`.

The main contexts can be described as follows:

- Theorem contexts, in which knowledge claims are stated. These include environments such as theorem, proposition etc.
- Proof contexts, which corresponds to the proof environment.
- Meta contexts, which collect reflexive parts of the article and include environments such as example etc.
- Outer context, which captures the outer level of the source with the above-mentioned contexts removed. This can be considered the 'paratext'.

In addition to the text split into contexts, it is also possible to adjoin metadata either from the original metadata provided by arXiv (*arXiv Dataset* 2020), derived from it, or built from other studies. A full list of the variables that can be adjoined can be obtained, but the most applicable ones are summarized in Appendix A.

7 Building a corpus from the DH4PMP LaTeX-pipeline

To extract a corpus of articles split into contexts from the DH4PMP LaTeX-pipeline, one must form a *corpus*. In this situation, a corpus can be defined either

1. by specifying a list of arxivIDs to include, or
2. by specifying a selection criterion from the database, using the SQL commands of SQLite3, using the variables `maincat` or `yymm` (others can be added).

For instance, to build a corpus of all papers in `math.GT` from 2020, one would define the corpus as

Henrik Kragh SÃ¸rensen (May 24, 2021a). *The DH4PMP LaTeX-pipeline for processing arXiv sources.* DH4PMP Work-in-Progress & Proof-of-Concept Series 12. Version 1. 18 pp. URL: https://www.erda.dk/vgrid/DH4PMP/wippoc/paper12.pdf.

```
maincat LIKE "math.GT" AND yymm LIKE "20__"
```

where the underscores represent any character. Similarly, to define a corpus of all texts with a main mathematical category, one would specify the criterion

```
maincat LIKE "math.__"
```

The implementation of the arXiv corpus is the class `arXivCorpus` derived from the abstract class `Corpus`. Its function `export_dataframe` returns a dataframe of the corpus along with a dictionary of information about the corpus in a format that can be processed directly or included in the LaTeX include file generated by the `dh4pmp-paper` class. The function `export_dataframe` takes a number of parameters, determining which information is included in the resulting dataframe:

- `contexts`: A list of which contexts to include (or "all" to include them all).
- `data`: A list of which elements in the extracted data dictionary to include in the mappings (or "all"). These elements include `REF_INTERNAL`, `MATH_DISPLAY`, `CITE_EXTERNAL`, and `GRAPHICS_FILE`.
- `metadata`: A list of which metadata to include (or "all"). Presently, only `yymm`, `maincat`, `mathcats` can be included directly; other metadata can be added by a left-join with the JSON-metadata on `arxivid`.
- `params`: A list of which other parameters to compute (or "all"). Presently, includes `lenghts`, which counts the number of instances of each context, `timestamp` which includes the processing timestamp, and `subversion` which includes the version of the processing pipeline.
- `minimum_version`: A string to indicate the minimum processing pipeline version to allow in the dataframe. Currently not implemented.

Each context column in the dataframe is (typically) returned in a JSON-encoded form, thus should be converted before it becomes a list (of lists):

```
df['proof'] = df['proof'].apply(json.loads)
```

The `info` dictionary contains information such as

- `metadata:size` indicating the size of the corpus as defined.
- `database:size` indicating the number of papers downloaded and processed which satisfy the corpus definition.
- `corpus:size` providing the resultant size of the corpus.
- `metadata:definition` containing the corpus definition (SQL).

References

arXiv Dataset (Nov. 22, 2020). *arXiv dataset and metadata of 1.7M+ scholarly papers across STEM*. Cornell University. URL: `https://www.kaggle.com/Cornell-University/arxiv/metadata` (visited on 01/30/2021).

Beebe, Nelson H. F. (2004). "25 Years of TeX and METAFONT. Looking Back and Looking Forward". TUG 2003 Keynote Address. *TUGboat*, vol. 25, no. 1, pp. 7–30.

Faist, Philippe (2019). *pylatexenc*. Version 2.8. URL: `https://pylatexenc.readthedocs.io/`.

Henrik Kragh SÃ¸rensen (May 24, 2021a). *The DH4PMP LaTeX-pipeline for processing arXiv sources*. DH4PMP Work-in-Progress & Proof-of-Concept Series 12. Version 1. 18 pp. URL: `https://www.erda.dk/vgrid/DH4PMP/wippoc/paper12.pdf`.

Knuth, Donald E. (1986). *The TEXbook*. Reading et al.: Addison-Wesley. ISBN: 978-0-201-13448-3.

Lamport, Leslie (1994). *LaTeX: A Document Preparation System. User's Guide and Reference Manual*. 2nd ed. Reading (Mass.): Addison-Wesley Publishing Company.

Mejia-Ramos, Juan Pablo et al. (2019). "Using Corpus Linguistics to Investigate Mathematical Explanation". In: *Methodological Advances in Experimental Philosophy*. Ed. by Eugen Fischer and Mark Curtis. London et al.: Bloomsbury Academic. Chap. 8, pp. 239–264.

Miller, Bruce and Deyan Ginev (Nov. 17, 2020). *LaTeXML. A LaTeX to XML/HTML/MathML Converter*. URL: `https://dlmf.nist.gov/LaTeXML/` (visited on 05/20/2021).

SÃ¸rensen, Henrik Kragh (May 23, 2021b). *Validating the DH4PMP pipeline and corpus for studying explanations in the arXiv*. DH4PMP Work-in-Progress & Proof-of-Concept Series 16.

Shinan, Erez (2021). *Lark*. URL: `https://lark-parser.readthedocs.io/`.

Sørensen, Henrik Kragh (Nov. 14, 2021). *Status of the DH4PMP LaTeX-pipeline*. DH4PMP Work-in-Progress & Proof-of-Concept Series 13.

Henrik Kragh SÃ¸rensen (May 24, 2021a). *The DH4PMP LaTeX-pipeline for processing arXiv sources*. DH4PMP Work-in-Progress & Proof-of-Concept Series 12. Version 1. 18 pp. URL: `https://www.erda.dk/vgrid/DH4PMP/wippoc/paper12.pdf`.

A Metadata

abstract : Abstract

Catalog(s): basic

authors : Authors

Catalog(s): basic

authors_parsed : Authors processed into list

Catalog(s): basic

categories : Categories separated by spaces

Catalog(s): basic

categories_parsed : List of categories

Catalog(s): basic_auto
Depends on: categories

comments : Comments on publication etc.

Catalog(s): basic

doi : DOI of eventual publication

Catalog(s): basic

e_Figs_pp : Number of figure (etc.) environments per page

Symbol: $e(\text{Figs})_{pp}$
Catalog(s): env_auto
Depends on: e_figure • e_tikzpicture • e_wrapfigure • numpages

e_figure : Number of figure environments in source

Symbol: $e(\text{figure})$
Catalog(s): env

e_tikzpicture : Number of tikzpicture environments in source

Symbol: $e(\text{tikzpicture})$
Catalog(s): env

e_wrapfigure : Number of wrapfigure environments in source

Symbol: $e(\text{wrapfigure})$
Catalog(s): env

f_gif : Number of gif files in tar archive

Catalog(s): src

Henrik Kragh SÃ¸rensen (May 24, 2021a). *The DH4PMP LaTeX-pipeline for processing arXiv sources.* DH4PMP Work-in-Progress & Proof-of-Concept Series 12. Version 1. 18 pp. URL: https://www.erda.dk/vgrid/DH4PMP/wippoc/paper12.pdf.

f_images : Number of image files in tar archive

Catalog(s): src_auto

Depends on: f_gif • f_jpg • f_pdf

f_jpg : Number of jpg files in tar archive

Catalog(s): src

f_pdf : Number of pdf files in tar archive

Catalog(s): src

f_tex : Number of tex files in tar archive

Catalog(s): src

f_total : Total number of files in tar archive

Catalog(s): src

id : arXiv identification

Catalog(s): basic • basic_auto • env • env_auto • pdf • src • src_auto

Details: This is used as the key for combining the different metadata catalogs. Often renamed arxivid.

maincat : Main (first) category listed

Catalog(s): basic_auto

Depends on: categories_parsed

mainmathcat : Main (first) math.* category listed

Catalog(s): basic_auto

Depends on: categories_parsed

Details: Notice that this variable *need not* refer to the *first* category assigned to the article, but only to the first math.* category listed.

missing_files : List of files not found while parsing the source tar

Catalog(s): src

num_missing_files : Number of missing files in source tar

Catalog(s): src_auto

Depends on: missing_files

numauthors : Number of authors

Symbol: #authors

Catalog(s): basic_auto

Depends on: authors_parsed

numcats : Number of categories

Henrik Kragh SÃ¸rensen (May 24, 2021a). *The DH4PMP LaTeX-pipeline for processing arXiv sources.* DH4PMP Work-in-Progress & Proof-of-Concept Series 12. Version 1. 18 pp. URL: https://www.erda.dk/vgrid/DH4PMP/wippoc/paper12.pdf.

Catalog(s): basic_auto

Depends on: categories_parsed

nummathcats : Number of math.* categories

Catalog(s): basic_auto

Depends on: categories_parsed

numpages : Number of pages in pdf file

Symbol: #pages

Catalog(s): pdf

pdf_filename :

Catalog(s): pdf

pdfnumwords : Number of words in pdftext

Catalog(s):

Depends on: pdftext

pdftext : Extracted text from pdf file

Catalog(s): pdf

t_meta : Extracted meta-context text from src files

Catalog(s): env

t_outer : Extracted paratext (outer-context text) from src files

Catalog(s): env

t_proof : Extracted proof-context text from src files

Catalog(s): env

t_theorem : Extracted theorem-context text from src files

Catalog(s): env

texstructure : Structure of the tex document from src files

Catalog(s): env

title : Title

Catalog(s): basic

update_date :

Catalog(s): basic

version : Maximal version number

Catalog(s): basic_auto

Henrik Kragh SÃ¸rensen (May 24, 2021a). *The DH4PMP LaTeX-pipeline for processing arXiv sources.* DH4PMP Work-in-Progress & Proof-of-Concept Series 12. Version 1. 18 pp. URL: https://www.erda.dk/vgrid/DH4PMP/wippoc/paper12.pdf.

basic	abstract • authors • authors_parsed • categories • comments • doi • id • title • update_date • versions
basic_auto	categories_parsed • id • maincat • mainmathcat • numauthors • numcats • nummathcats • version
env	e_figure • e_tikzpicture • e_wrapfigure • id • t_meta • t_outer • t_proof • t_theorem • texstructure
env_auto	e_Figs_pp • id
pdf	id • numpages • pdf_filename • pdftext
src	f_gif • f_jpg • f_pdf • f_tex • f_total • id • missing_files
src_auto	f_images • id • num_missing_files

Table 1: List of catalogs in the arXiv data. Catalogs with suffix auto are derived from the fundamental catalogs.

Depends on: versions

versions :

Catalog(s): basic

yymm : Year and month extracted from arXiv ID

Catalog(s):

Depends on: id

Henrik Kragh SÃ¸rensen (May 24, 2021a). *The DH4PMP LaTeX-pipeline for processing arXiv sources.* DH4PMP Work-in-Progress & Proof-of-Concept Series 12. Version 1. 18 pp. URL: https://www.erda.dk/vgrid/DH4PMP/wippoc/paper12.pdf.