International Journal on Digital Libraries (2024) 25:517–535
https://doi.org/10.1007/s00799-023-00370-5

RDFtex in-depth: knowledge exchange between LATEX-based research publications and Scientific Knowledge Graphs

Leon Martin[1] · **Andreas Henrich**[1]

Received: 9 November 2022 / Revised: 26 April 2023 / Accepted: 7 May 2023 / Published online: 31 July 2023
© The Author(s) 2023, corrected publication 2023

Abstract
For populating Scientific Knowledge Graphs (SciKGs), research publications pose a central information source. However, typical forms of research publications like traditional papers do not provide means of integrating contributions into SciKGs. Furthermore, they do not support making direct use of the rich information SciKGs provide. To tackle this, the present paper proposes *RDFtex*, a framework enabling (1) the import of contributions represented in SciKGs to facilitate the preparation of LATEX-based research publications and (2) the export of original contributions from papers to facilitate their integration into SciKGs. The framework's functionality is demonstrated using the present paper itself since it was prepared with our proof-of-concept implementation of RDFtex. The runtime of the implementation's preprocessor was evaluated based on three LATEX projects with different numbers of imports and exports. A small user study ($N = 10$) was conducted to obtain initial user feedback. The concept and the process of preparing a LATEX-based research publication using RDFtex are discussed thoroughly. RDFtex's import functionality takes considerably more time than its export functionality. Nevertheless, the entire preprocessing takes only a fraction of the time required to compile the PDF. The users were able to solve all predefined tasks but preferred the import functionality over the export functionality because of its general simplicity. RDFtex is a promising approach to facilitate the move toward knowledge graph augmented research since it only introduces minor differences compared to the preparation of traditional LATEX-based publications while narrowing the gap between papers and SciKGs.

Keywords Research data management · Data and research infrastructure · LATEX-based research publications · Scientific Knowledge Graphs

1 Introduction

Between 2008 and 2018, i.e., in one mere decade, the number of research publications published each year grew from about 1.8 million to about 2.6 million [1]. The resulting flood of publications and scientific data poses different challenges for researchers. For instance, keeping track of relevant related work for a certain topic and state-of-the-art experimental results has become increasingly difficult. Hence, the need for new means of organizing publications and scientific data has risen, eventually leading to the proposal of Scientific Knowledge Graphs (SciKGs) [2–4] that aim to integrate scientific information into a knowledge base. SciKGs have the potential to fundamentally transform the way researchers acquire information for preparing their publications and share their contributions. However, the envisaged shift toward what is herein called *knowledge graph augmented research* raises questions about the form of research publications, i.e., their suitability for this new research paradigm. In a previous paper [5], we discussed three publication forms (including the predominant *document-based publications*) regarding their utility with respect to knowledge graph augmented research. The investigation showed that all of them are flawed in this context. Afterward, a set of five requirements (cf. Sect. 2) for a publication form that is specifically designed for knowledge graph augmented research was compiled with respect to the identified advantages and disadvantages.

Building upon this, the present paper proposes RDFtex as an attempt to narrow the gap between LATEX-based[1] research

✉ Leon Martin
leon.martin@uni-bamberg.de

Andreas Henrich
andreas.henrich@uni-bamberg.de

1 Media Informatics, University of Bamberg, An der Weberei 5, 96052 Bamberg, Germany

1 https://www.latex-project.org (accessed 2023/03/08).

publications and SciKGs. RDFtex is a framework enabling a bidirectional knowledge exchange between LaTeX-based research publications and SciKGs. For this purpose, RDFtex comprises two functionalities:

1. The import functionality allows the import of research contributions from SciKGs in LaTeX documents via a custom import command.
2. The export functionality facilitates the export of original research contributions from LaTeX documents to a SciKG via two custom export commands.

To showcase RDFtex's functionalities, the present paper has been prepared using our proof-of-concept implementation of RDFtex[2] and a makeshift SciKG called *MinSKG* (cf. Sect. 4).

The present paper is an extended version of our paper accepted at the TPDL 2022 conference [6]. In comparison, this paper provides a broader investigation of related work, a more detailed explanation of RDFtex's workflow, a discussion of the compatibility between RDFtex and related technologies, a description of an end-to-end research and publication process incorporating RDFtex, a more sophisticated runtime evaluation, and the collection as well as examination of additional user feedback among others. Regarding the scope of the IJDL,[3] this paper contributes to the aspects *interoperability of different digital objects* and *user interfaces* as RDFtex helps researchers to increase the interoperability between their research publications and SciKGs.

The remainder of this paper is structured as follows: Sect. 2 describes relevant foundations. Then, Sect. 3 focuses on related work, followed by a thorough introduction of RDFtex's functionalities in Sect. 4. Building upon this, Sect. 5 shows how RDFtex can be embedded in an end-to-end research and publication process. In Sect. 6, RDFtex is discussed from different perspectives. The compatibility with other technologies and future work are addressed in this section as well. Finally, Sect. 7 draws a conclusion.

2 Foundations

The Resource Description Framework (RDF) [7] provides a generic approach for representing knowledge in the form of triples, where Internationalized Resource Identifier (IRIs), a generalization of Uniform Resource Identifiers (URIs), is used as identifiers. Each triple comprises a subject (an IRI or a blank node), a predicate (an IRI), and an object (an IRI, a literal, or a blank node). The predicate represents a property, i.e., a binary relation between the subject and the object. Although definitions in the community vary [8], one can state that collections of RDF triples that represent real world entities and their interrelations with respect to a predefined ontology constitute so-called knowledge graphs.

Typically, triplestores [9] are employed to handle and interact with RDF-based knowledge graphs. To retrieve information from knowledge graphs, they usually provide an interface that accepts queries written in the SPARQL Protocol And RDF Query Language (SPARQL) [10]. For exchanging and archiving knowledge graphs, various serialization formats such as Turtle, RDF/XML, and N3 exist [7, 11].

In contrast with more general knowledge graphs that gather information across domains like Wikidata[4] and DBpedia,[5] SciKGs, i.e., knowledge graphs that are tailored to a (certain) scientific domain, represent rich knowledge bases specifically for scientific information. For example, the Open Research Knowledge Graph (ORKG) [3] and the OpenAIRE research graph [12] gather scientific information across disciplines, whereas KnowLife [13] specifically focuses on scientific information from the life sciences. To acquire this information, research publications pose a central resource. Hence, SciKGs like the ORKG do not only aim to capture *contextual* information, i.e., metadata, of research publications but also *contentual* information, e.g., the original contributions the publications provide.[6] We use this terminology to make the distinction between metadata and content more clear since both comprise semantic information but at different levels of abstraction. Figure 1 gives an example of the difference between the two types of information.[7]

To build high-quality and complete SciKGs efficiently, the process of integrating scientific information from research publications should be as simple as possible and at best be performed automatically with only little human intervention for quality assurance. From the authors' perspective, SciKGs provide the opportunity to facilitate the preparation of new publications if the rich information they provide is leveraged. However, an investigation of three different available publi-

[2] The implementation is written in Python. The code and other used resources are available at https://github.com/uniba-mi/rdftex (accessed 2023/03/08). For future proofing, the repository has also been indexed in the Software Heritage Project's archive (https://archive.softwareheritage.org/; accessed 2023/03/08).

[3] See https://www.springer.com/journal/799 (accessed 2023/03/08).

[4] https://www.wikidata.org (accessed 2023/03/08).

[5] https://www.dbpedia.org (accessed 2023/03/08).

[6] There are other parts of a publication that can also be categorized as contentual information like the description of the research topic, applied methods, or metrics. The remainder of this paper will use contentual information mainly as a term for the original contributions, though.

[7] The use of the available vocabularies for the predicates has been omitted in the example for simplicity. Furthermore, following the official documentation of RDF [7, 14], https://example.org is employed as the domain of the namespaces used in the remainder of the present paper.

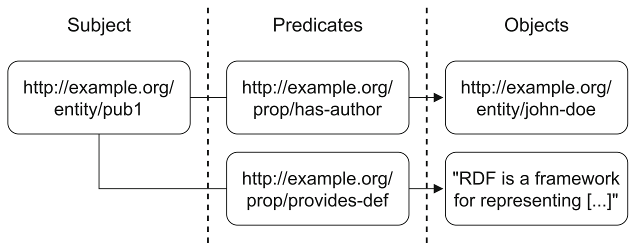

Subject
Predicates
Objects
http://example.org/ entity/pub1
http://example.org/ prop/has-author
http://example.org/ entity/john-doe
http://example.org/ prop/provides-def
"RDF is a framework for representing [...]"

Fig. 1 A simple exemplary knowledge graph consisting of two RDF triples. The upper triple provides contextual information, the lower triple contentual information of the publication *pub1*. All non-literal triple members are identified using IRIs. (Figure and caption adopted from [5].)

cation forms in [5] shows that all of them have disadvantages regarding knowledge graph augmented research.

First, *document-based publications*, commonly called papers, pose the problem that authors have to introduce the same concepts across multiple publications to the readers even if this information is readily available in SciKGs. This results in redundant passages—typically found in a paper's introductory sections—that provide no added value while consuming valuable preparation time and effort. Furthermore, keywords and other classification systems do not allow for a direct integration of a publication into a SciKG because they are usually not formatted in an RDF-compatible way.

RDF-transformed publications aim to close the gap between document-based publications and RDF by leveraging tools like *SciIE* [4], a framework for scientific information extraction, to generate RDF graphs from document-based publications. Although the performance of such approaches has increased in the recent years, the problems regarding the redundant passages continue to exist since authors still have to prepare regular papers in the first place.

Definition 1 Nanopublications [provide] a granular and principled way of publishing scientific (and other types of) data in a provenance-centric manner. Such a nanopublication consists of an atomic snippet of a formal statement [...] that comes with information about where this knowledge came from [...] and with metadata about the nanopublication as a whole [...]. All these three parts are represented as Linked Data (in RDF) [...]. [15]

Finally, *nanopublications* [15] (cf. Definition 1) focus on the integration with RDF by directly encoding contributions as RDF triples. This, however, implies additional training effort for authors while readers will have to get accustomed to this publication form. As long as nanopublications are not well-established in the scientific community, the need for an interim solution for integrating the predominant form of publications, i.e., document-based publications, in SciKGs arises, to which RDFtex is our proposal.

Table 1 A set of requirements for a future publication form tailored for use in scientific knowledge graphs

#	Requirement
1	Preparation of main contributions in natural language
2	Import of knowledge
3	Markup of knowledge
4	Enriched representation of publications at view time
5	Provision of tooling for obtaining IRIs

(Adapted from [5])

Based on the discussed advantages and disadvantages, five requirements (s. Table 1) for a publication form tailored for the use in SciKGs were proposed in [5]. The first requirement states that authors shall still be able to prepare their main contributions in natural language. RDFtex can meet this requirement only partially since the underlying LaTeX documents require non-natural language statements by design. The import and export commands introduced by RDFtex also do not use natural language. Hence, RDFtex permits the usage of natural language to the same degree regular LaTeX documents do. The second requirement prescribes a means of importing knowledge from SciKGs with the goal of avoiding the need to prepare redundant passages across publications. RDFtex allows importing contributions from a SciKG in LaTeX documents via import commands. In RDFtex's preprocessing step, actual LaTeX content based on the contributions' information retrieved from the SciKGs is inserted. Vice versa, the third requirement demands means of exporting knowledge from publications to a SciKG facilitating the integration of research publications. For this purpose, RDFtex allows marking up original contributions in a LaTeX-based publication via export commands that are then parsed in the preprocessing step to generate an RDF document representing the publication's contributions. The fourth requirement states that readers shall receive a version of the publication that is enriched using information imported from the SciKG. In the preprocessing step, the import commands are replaced by templates that are populated with information retrieved from the SciKG. Compiling the preprocessed LaTeX documents to PDF afterward results in one coherent document that is enriched using the information from the SciKG.

The remaining fifth requirement is concerned with the need for tooling to obtain relevant IRIs from SciKGs, thus addressing the usage context of RDFtex rather than the publication form itself. Its discussion is therefore postponed to Sect. 5 after the introduction of RDFtex in Sect. 4.

3 Related work

In the past, different approaches for linking parts of documents in other documents have been proposed. One early example, Project Xanadu [16], introduced *xanadocs*. Essentially, a xanadoc is a so called Edit Decision List (EDL) file that consists of a set of links—typically URIs—referring to other documents with numerical start and length information for identifying spans of text in the linked resources. Special xanadoc viewers generate one coherent document by retrieving and composing the text spans from the linked documents at the specified start positions with the specified lengths. Similar to xanadocs and with compliance to RDF, RDFtex makes use of IRIs to reference certain resources, which correspond to scientific contributions in our case. While text spans constitute the content of xanadocs, RDFtex retrieves the properties related to scientific contributions reflected in a SciKG to generate content in a template-based manner.

The idea of making contributions of research publications more accessible for other researchers has been explored, as well. One example in this regard is the platform SciSpace [17], where an AI chatbot can be asked to summarize the content of a paper, to provide further explanations on highlighted text passages, or to describe key contributions. This additional information can help other researchers discovering facts, citations etc., which can then be used for the related work of new publications, thus ultimately facilitating further research. The import and export functionality of RDFtex follows a similar goal but rely on explicit markup and SciKGs rather than AI-based natural language processing and generation methods.

Another approach related to RDFtex is the so-called Resource Description Framework in Attributes (RDFa) [18], which is a technology that provides a set of special attributes for XML-based languages to mark up elements with machine-readable RDF information. Listing 1 demonstrates RDFa's core functionality using the example of an HTML snippet. By exploiting HTML's tree structure, the RDFa information encoded in the example can be transformed into a knowledge graph featuring one blog post entity that acts as the subject of two triples, where the `property` attributes of the nested elements specify the predicates and the element contents correspond to the objects. To improve the readability of the source code, RDFa features a `prefix` attribute for specifying prefixes for certain vocabulary IRIs. The prefixes can then be used in place of the full vocabulary IRIs in the RDFa attributes of the nested elements. The first triple encoded in the example thus consists of the subject `.../trouble_with_bob`, the predicate `http:/.../dc/terms/title`, which leverages the `dc` prefix, and the object `The trouble with Bob`.

RDFtex's export functionality follows RDFa's idea of marking up content with RDF information to provide machine readability. In RDFtex's preprocessing step, an RDF document is generated based on the marked up information that can then be used as a basis for integrating the publication into a SciKG. To improve readability, RDFtex features a prefix syntax, too.

Listing 1 RDFa's core functionality using the example of an HTML snippet. The code was adapted from [18]. Note how the tree structure of XML-based languages is exploited by RDFa.

```
<body prefix="dc:␣http:/.../dc/terms/">
  <div resource=".../trouble_with_bob">
    ...
    <h2 property="dc:title">
      The trouble with Bob
    </h2>
    ...
    <h3 property="dc:creator"
        resource="#me">
      Alice
    </h3>
    ...
  </div>
</body>
```

Apart from this, there are projects investigating means of marking up the contents specifically in LaTeX documents with additional semantic information for further processing. For instance, *sTeX* and its extension *sTeX+* [19] allow marking up mathematical knowledge in LaTeX documents. For this, *sTeX* leverages semantic macros, i.e., sequences of TeX commands that represent mathematical concepts. The annotations can be transformed into mathematical knowledge management representation formats, based on which human-readable XHTML+MathML documents can be generated subsequently. As an alternative, the annotations of the sTeX+ extension can be translated into a typical Web Ontology Language (OWL) [20] serialization format and also XHTML+MathML+RDFa, thereby providing compatibility with linked data.

Compared to sTeX and sTeX+, RDFtex provides the novelty that knowledge cannot only be marked up and exported but also imported. Another key difference is that sTeX and sTeX+ do not aim to interoperate with a SciKG despite sTeX+'s RDF compatible output formats.

As another example, *SALT* [21] is a comprehensive framework that aims to capture the content of LaTeX documents using a federation of three interlinked ontologies:

1. The *Document Ontology* captures the structural layout of the publication.
2. The *Rhetorical Ontology* is used to reflect the rhetorical structure of the paper, which comprises individual claims and explanations that are supported or refuted by certain arguments.
3. The *Annotation Ontology* provides the publication's metadata and establishes a link between the rhetorical content from the Rhetorical Ontology and the physical structure represented in the Document Ontology.

The granularity of these ontologies explains SALT's expressive power but also its complexity. In comparison, RDFtex's level of abstraction directly depends on the structure of the SciKG with which the knowledge exchange is practiced. Using SALT's terminology, one could state that RDFtex currently, i.e., in combination with the MinSKG (cf. Sect. 4), operates mostly on the levels of the Document Ontology and the Annotation Ontology. As a result, RDFtex uses fewer properties, which is beneficial regarding the ease of use. Switching to a SciKG that represents contributions on a more rhetorical level would shift RDFtex's level of abstraction as well.

Similar to sTeX and sTeX+ and in contrast with RDFtex, SALT does not provide means of importing contributions but only of exporting content. Regarding the interoperability with a SciKG, the employed federation of the three ontologies itself is actually almost powerful enough to establish a SciKG itself but has not yet been applied for this purpose. Instead, the resulting RDF information is transformed to a set of HTML files by the so called *SALT-WebPub* [21] application facilitating the publication on the web.

Another example in this direction proposed in 2022 is SciKGTeX [22], a LaTeX package for annotating TeX source code to describe contributions following the ORKG schema. For this, it supports the usage of properties available in the ORKG but also allows defining custom properties. The annotations are embedded into the metadata of the compiled PDF files, thus facilitating the integration of the contributions in the ORKG.

Further emphasizing the novelty of RDFtex's import functionality in the context of LaTeX-based research publications, SciKGTeX also only addresses the export side of the knowledge exchange. Moreover, it is strictly limited to the ORKG in contrast with RDFtex, which is designed to interoperate with different SciKGs.

The previous remarks indicate that there are certain overlaps between the presented LaTeX-related technologies and RDFtex despite the stated differences. Section 6 will therefore address whether and how RDFtex could be used in combination with sTeX(+), SALT, and SciKGTeX.

4 Concept

RDFtex's concept comprises the import and export of research contributions from and to SciKGs. Both being processes, the concept will be introduced in the following sections from a process-oriented point of view.

Introducing the framework, Fig. 2 shows that RDFtex operates on files with the `.rdf.tex` file extension and adds a composite preprocessing step to the basic workflow for producing a PDF document from `.tex` source files. This preprocessing step includes the export and the import of

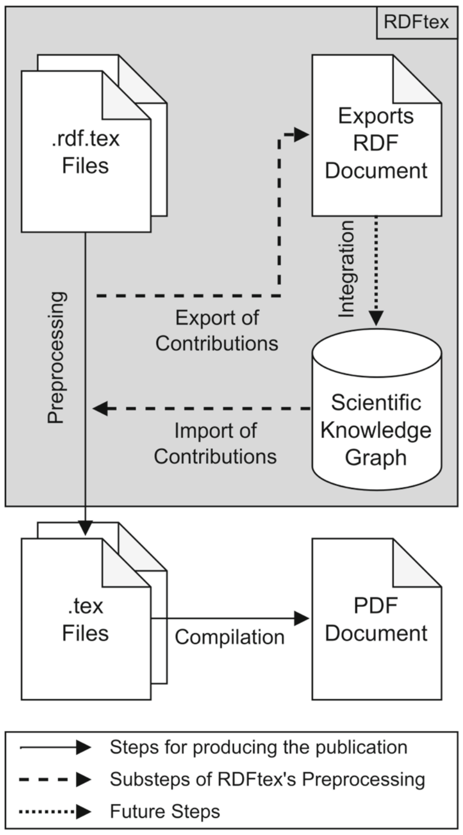


Fig. 2 General process for producing a PDF file using RDFtex. The gray box indicates additional steps and resources introduced or leveraged by the RDFtex framework

contributions, which will be explained below. Syntactically, `.rdf.tex` files are regular `.tex` files that, moreover, permit the usage of custom RDFtex commands. The special file extension facilitates identifying the source files with the custom commands for preprocessing. Apart from the occasional usage of the custom RDFtex commands, authors can prepare their publication using `.rdf.tex` files like they are used to.

In the preprocessing step, a regular `.tex` document is generated for each `.rdf.tex` document in the root folder and all subdirectories by either replacing the custom commands with a snippet of actual LaTeX content or removing them. The resulting `.tex` files can then be compiled to PDF as usual. As indicated by the dotted arrow, the RDF document resulting from the export substep of the preprocessing is supposed to be integrated in a target SciKG later on (s. Sect. 5).

For convenience, RDFtex is designed to be compatible with a fully automated workflow: Our proof-of-concept preprocessor implementation can be configured to be executed whenever a `.rdf.tex` file changes. Similarly, tools like *Latexmk*[8] [23] provide the option to compile LaTeX projects

[8] Latexmk is used for the runtime evaluation in Sect. 6 and for the implementation of RDFtex due to its high LaTeX compilation speed and configurability.

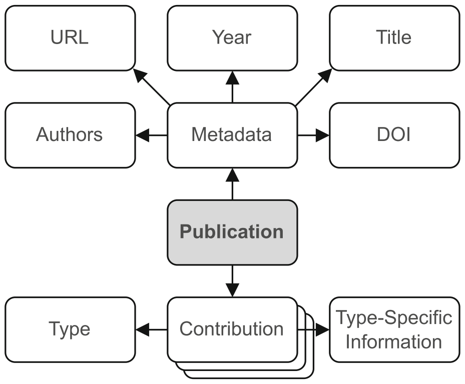


Fig. 3 Information for each publication represented in the MinSKG. Edge labels, i.e., predicates, and IRIs are omitted for readability

whenever a `.tex` file changes. Combining the two, the PDF is compiled fully automatically whenever changes are made to any `.rdf.tex` file.

The present paper has been prepared using our proof-of-concept implementation of RDFtex. Thus, examples originating from this paper will be used in the following to explain the import and export functionality. As said before, preparing a LaTeX-based research publication using RDFtex requires suitable SciKGs with which the knowledge exchange is practiced. To obtain community feedback on the concept of RDFtex before definite decisions are made that are not received well and are difficult to change afterward, a makeshift SciKG called *MinSKG* was created. The MinSKG serves as a sandbox for exploring and demonstrating the concept of RDFtex without being subject to dependencies to other technologies or ecosystems, more specifically limitations, that actual SciKGs would potentially impose.

Available SciKGs do not yet contain all publications and contributions referenced in the present paper. However, a SciKG comprising the said publications and contributions is required to showcase RDFtex. Hence, the MinSKG has been populated with the necessary information. The contextual information of the publications, i.e., the metadata, was added automatically by parsing their entries from the `bibtex` file. Their original contributions were added manually. Note that only the research contributions that are actually imported in the present paper have been added to the MinSKG. Currently, the MinSKG and our proof-of-concept implementation of RDFtex support datasets, definitions, simple experimental results, figures, and software artifacts as contribution types for import and export. The MinSKG itself as well as the scripts for building and running it is provided in RDFtex's repository [2].

Figure 3 shows the structure of the information for publications represented in the MinSKG. The structure is loosely based on the ORKG to ensure that RDFtex operates on a SciKG with a realistic structure. Still, the MinSKG just serves as a small makeshift SciKG that has to be replaced by an actual SciKG in the future (s. Sect. 6). As indicated, contributions reflected in the MinSKG comprise different type-specific information with respect to the type of the contribution. Switching to an actual SciKG will also automatically result in the replacement of the following makeshift properties that are currently used to describe the contribution types in the MinSKG:

- Definition
  - `https://.../terms/type`
  - `https://.../terms/definition_content`
- Dataset
  - `https://.../terms/type`
  - `https://.../terms/dataset_name`
  - `https://.../terms/dataset_description`
  - `https://.../terms/dataset_domain`
  - `https://.../terms/dataset_url`
- Figure
  - `https://.../terms/type`
  - `https://.../terms/figure_description`
  - `https://.../terms/figure_url`
  - `https://.../terms/figure_mime`
- Simple Experimental Result
  - `https://.../terms/type`
  - `https://.../terms/expresult_description`
  - `https://.../terms/expresult_result`
  - `https://.../terms/expresult_samplesize`
- Software
  - `https://.../terms/type`
  - `https://.../terms/software_name`
  - `https://.../terms/software_description`
  - `https://.../terms/software_url`

4.1 Import of contributions

RDFtex's import command enables authors to import contributions from a SciKG. During the preprocessing step, the import commands within the `.rdf.tex` files are parsed and snippets of actual content are produced based on the contribution information retrieved from the SciKG of choice. The snippets are added to the `.tex` files in place of the import commands. To identify the contributions to be imported, IRIs are used. Hence, the import commands use this syntax[9]:

```
\rdfimport{<label>}{<citation-key>} ↩
  {<contrib-iri>}{<skg>}
```

As depicted, the `rdfimport` command expects four parameters. The first parameter `<label>` corresponds to the label the authors want to assign to the generated content for future reference in their paper. Second, `<citation-key>` is

[9] In the following, ↩ symbols indicate word wrapping. In reality, there are no line breaks.

a placeholder for the `bibtex` citation key of the publication from which the imported contribution originates. The third parameter `<contrib-iri>` denotes the IRI of the contribution. Finally, the fourth parameter `<skg>` states the SciKG from which the contribution is to be imported. The next paragraphs will build on the example of Fig. 1, which we imported in our `.rdf.tex` file as follows:

```
\rdfimport{fig:contentual-contextual} ↩
  {Martin21} ↩
  {https://.../Martin21/contrib2} ↩
  MinSKG
```

The example shows that the import command is agnostic regarding the type of the imported contribution such that there is no need to learn different import syntaxes. In the example, the MinSKG is specified as the SciKG from which the contribution is to be imported. With respect to the MinSKG's structure (cf. Figure 3), a SPARQL query of the form

```
SELECT ?p ?o
WHERE {<contrib-iri> ?p ?o .}
```

is therefore executed on the MinSKG to obtain the contribution's information. This query yields all predicates and objects of triples where `<contrib-iri>` is the subject. The retrieved information as well as the remaining two parameters is used in the next step to generate the content snippets. The number of triples available depends on the type of the contribution. In the case of figures, the MinSKG provides information about the contribution type itself which in this case would be `Figure`, a description, the URL at which the actual figure can be found, and the figure's MIME type, as the list above shows.

The specified SciKG and the type of the contribution to be imported determine the template for generating the content. The import templates are written in LaTeX to allow for a direct injection in the `.tex` files and incorporate the respective information that different contribution types in the specified SciKG provide. The two pieces of information that are used by all templates are `<citation-key>` and `<label>`. To make the imported parts of the resulting `.tex` files salient, all import templates are enclosed with LaTeX comments. The template that is applied for importing figures from the MinSKG looks as follows:

```
% RDFtex Figure Import Start
\begin{figure}[t]
  \centering
  \includegraphics[...]{<figure-url>}
  \caption{<figure-desc> ↩
    (Figure and caption adopted ↩
    from \cite{<citation-keys>}.)}
  \label{<label>}
\end{figure}
% RDFtex Figure Import End
```

As shown, the template uses the placeholders `<figure-url>` and `<figure-desc>`, which correspond to the location of the figure and its description. Another example is the following template for importing definitions:

```
% RDFtex Definition Import Start
\begin{definition}
  \label{<label>}
  <definition-content> ↩
     \cite{<citation-key>}
\end{definition}
% RDFtex Definition Import End
```

Apart from the placeholders `<citation-key>` and `<label>`, this template comprises just a `<definition-content>` placeholder, which is replaced with the actual text of the definition during preprocessing. This template was used to import the definition for nanopublications, i.e., Definition 1, from [15].

While the values for `<citation-key>` and `<label>` are directly taken over from the `rdfimport` command, all other placeholders within the import templates are populated with information retrieved from the MinSKG. Thus, it is important to ensure that no contributions are added to SciKGs without the necessary predicates and objects. Otherwise, they cannot be imported in other publications successfully. If contributions with incomplete information are encountered, the preprocessor shows meaningful error messages. For some of the supported contribution types, templates with custom LaTeX environments are used whose definitions are automatically inserted into the `.tex` files if applicable.

At this point, it is important to discuss the problem that different SciKGs might use different properties and structures to describe contributions of different types. Some SciKGs might even offer contribution types that others do not. The MinSKG and other SciKGs like the ORKG employ tree-like structures[10] to represent the information of contributions (cf. Figure 3). However, the tree-like structures representing contributions in the MinSKG only have a depth of one, i.e., starting from each contribution entity there are only outgoing paths that have a maximum length of one. In contrast, the tree-like structures in the ORKG can theoretically be arbitrarily deep. Because of this, the SPARQL query to retrieve the contribution information from above does not suffice. Instead, a brute-force solution is to use a query of the form:

```
prefix x: <>
construct {?s ?p ?o}
where {
  <contrib-iri> (x:|!x:)* ?s .
  ?s ?p ?o .
}
```

This query yields all triples that constitute the tree-like structure representing a contribution starting from the contribution entity identified via `<contrib-iri>`. The triples

[10] The structures are acyclic, but the nodes can have an indegree of more than one.

therefore contain *all* the available information of the contribution disregarding the depth of the nodes. To give an example, executing the brute-force solution's query with `<http://orkg.org/orkg/resource/R8199>`, i.e., the *ORKG System* as proposed in [3], in place of `<contrib-iri>` using the ORKG SPARQL interface[11] returns 246 triples constituting a tree-like structure with a maximum depth of five[12]. To generate a snippet using the brute-force solution, a subset of these triples is accessed to populate the import template with the required information of the particular contribution type.

That being said, the brute-force solution is only applicable if the contributions are represented using tree-like structures in the queried SciKG. Otherwise, a more sophisticated solution is required. Such a solution could use, for example, a first query to identify the contribution type and a second query to retrieve exactly the necessary type-specific information with respect to the import template. While this solution can handle arbitrary SciKG structures, it also requires the manual creation of SPARQL queries for all supported contribution types of the queried SciKGs. The large resulting set of queries significantly impedes maintainability. Adding support for new contribution types and SciKGs is more costly compared to the brute-force solution, as well. For these reasons, the brute-force solution is strongly recommended if SciKGs with contributions made of tree-like structures are leveraged. Since both the MinSKG and the ORKG use such tree-like structures, the brute-force solution is adopted for the proof-of-concept RDFtex implementation, thus replacing the initial SPARQL query proposed in the beginning of this section.

The potential difference in structure and available properties entails that different templates might be required for importing contributions from different SciKGs even for contributions of the same type. Accordingly, import templates are to a certain degree SciKG-specific. Thus, Sect. 5 discusses which parties in the research and publication process are responsible for providing the import templates.

4.2 Export of contributions

For exporting original contributions, RDFtex enables authors to mark up relevant text passages in `.rdf.tex` files. Yet it is flexible enough to let authors choose which contributions they want to export. In the following, the export of the proof-of-concept preprocessor implementation[13] serves as the running example for demonstrating RDFtex's export functionality. The export functionality leverages two custom LaTeX commands. The first command is used to declare the export of a certain contribution. The rationale for introducing a declaration statement is that the exact string specifying the contribution type in the MinSKG, e.g., `ExpResult` (indicating an experimental result), might not be explicitly mentioned in the text. At the same time, the type information is mandatory since it determines the templates to be applied in the import process as explained above. The same applies to other predicates and objects, e.g., denoting the MIME type of a figure. To tackle this, the export declaration command

```
\rdfexport{<contrib-name>} ↩
  {<type>}{<predicate-iri=object>,...}
```

is introduced. `<contrib-name>` corresponds to the name of the contribution. The specified name serves as a local identifier for the contribution, which can be chosen arbitrarily by the users. The second parameter `<type>` denotes the contribution's type and the third parameter accommodates optional `<predicate-iri=object>` statements, which are placeholders for predicate object combinations that are required for a valid export but cannot be included appropriately in the publication text. For example, to add the MIME type information for a figure to be exported, the third parameter could be set to:

```
https://.../terms/figure_mime = image/png
```

Applied to the running example, the following snippet declares the export of the RDFtex implementation as a software artifact:

```
\rdfexport{RDFtex Implementation} ↩
  {Software}{}
```

Note that the third parameter was omitted here since all information required for a valid software export are stated explicitly in the publication text such that they can be marked up directly using the second export command.

The second export command is similar to the `property` attributes of RDFa (cf. Listing 1) for marking up predicates. While RDFa can exploit the nested element structures of XML-based languages to associate subjects with predicates and objects, the triple components are potentially spread across far apart sections, in our case. Therefore, each RDFtex property command has to explicitly reference the contribution they belong to via the `<contrib-name>` used in the export declaration command, resulting in this syntax:

```
\rdfproperty{<contrib-name >} ↩
  {<predicate-iri>}{<object>}
```

The parameter `<predicate-iri>` denotes a predicate used for describing contributions in the MinSKG and `<object>` encloses the part of the content that is to be used as the triple's object. For instance, to export the description of the RDFtex

[11] https://orkg.org/sparql (accessed 2023/03/08).

[12] As of 2023/03/08. The assessment in Sect. 6 shows that this solution does not significantly impede RDFtex's runtime.

[13] Note that not the actual preprocessor implementation is exported but a contribution of the type *Software* comprising the type-specific properties as listed in Sect. 4.

software, we modified the line from Sect. 1 in our `.rdf.tex` file to the following:

```
\rdfproperty{RDFtex Implementation} ↩
  {https://.../software_description} ↩
  {RDFtex is a framework enabling...}
```

When lines with `rdfexport` or `rdfproperty` commands are encountered during preprocessing, their arguments are collected for the respective contributions identified via `<contrib-name>`. For the `.tex` file generation, lines with `rdfexport` are omitted and `rdfproperty` commands are replaced with only their `<object>` argument, thereby removing the custom export commands, which would otherwise hinder the PDF compilation.

Theoretically, the export of a contribution could be performed using only the `rdfexport` command due to the power of its third parameter. However, the usage of the `rdfproperty` command offers the possibility to use parts of the text in the publication directly as objects for the predicates. Among others, this serves the objective of avoiding redundant text passages. Nevertheless, some users might find annotating the LATEX source code with the `rdfproperty` commands somewhat cumbersome while typesetting, especially since there is no change in the visible output. The third parameter of the `rdfexport` command can alleviate this problem by allowing to provide multiple predicate object combinations in one location.

When all files have been processed, the collected information is validated, i.e., checked for completeness in terms of properties required for the supported contribution types. This ensures that the exported contributions can be successfully imported in other publications later on. Hence, a good documentation is necessary to inform authors about the required properties for each contribution type, thus avoiding validation errors that lead to frustration. Based on the validated information, RDF triples representing the contributions are generated, as follows: First, a placeholder IRI based on a randomly generated UUID[14] is created and assigned to the publication entity. This IRI is replaced with a persistent identifier by the SciKG maintainers only in the integration step of the research and publication process (s. Sect. 5), right before the contributions are integrated in the SciKG. Then, an entity for each exported contribution is created using an IRI that is derived from the IRI of the publication's entity. Afterward, triples are created that link the publication's entity with each contribution entity and additional ones that link each contribution entity with its previously collected objects using the corresponding predicates. Figure 4 shows the triples that describe the contribution of the running example.

All triples created this way constitute the *exports RDF document*. At the end of the preprocessing, this document is

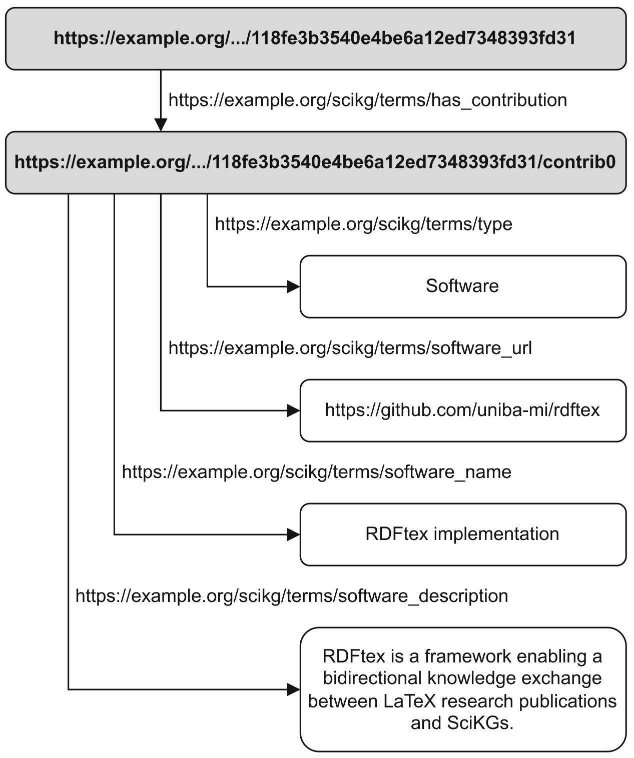

https://example.org/.../118fe3b3540e4be6a12ed7348393fd31
https://example.org/scikg/terms/has_contribution
https://example.org/.../118fe3b3540e4be6a12ed7348393fd31/contrib0
https://example.org/scikg/terms/type
Software
https://example.org/scikg/terms/software_url
https://github.com/uniba-mi/rdftex
https://example.org/scikg/terms/software_name
RDFtex implementation
https://example.org/scikg/terms/software_description
RDFtex is a framework enabling a bidirectional knowledge exchange between LaTeX research publications and SciKGs.

Fig. 4 The set of triples describing the contribution of the RDFtex implementation depicted as a graph. The nodes with a gray background represent the publication and the contribution entity, the others the type-specific contribution information. Note the tree-like structure of the graph and the UUID serving as a placeholder IRI for the publication and the contribution entity.

persisted, using the Turtle [11] serialization format because of its readability and thus maintainability. The exports RDF document resulting from the present paper can be found at the provided repository[2]. Note that the exports RDF document only contains the contentual information of a publication. The reason for this is that some metadata of the publication are not obtainable yet, e.g., the DOI has to be assigned by publishers first. Hence, the contentual information within the exports RDF document is fused with the finalized contextual information provided by the publishers in the integration step of the research and publication process (s. Sect. 5) and integrated in the SciKG by the SciKG maintainers afterward.

Similar to the import templates, the exports RDF document is bound to a particular SciKG since it comprises the exact contribution properties that are prescribed by the target SciKG for the individual contribution types. Thus, the exports RDF document cannot be integrated into arbitrary SciKGs—without additional effort—as long as they follow a different structure and build upon different properties. This issue is addressed more thoroughly in Sect. 5.

[14] See https://datatracker.ietf.org/doc/html/rfc4122.html (accessed 2023/03/08).

4.3 Prefix syntax

To mitigate the readability problems arising from lengthy IRIs, many RDF-related technologies including RDFa allow defining prefixes that can then be used to abbreviate full IRIs. Following this approach, RDFtex provides another command for registering prefixes:

```
\rdfprefix{<prefix>}{<path>}
```

The parameter `<prefix>` states the prefix to be used in place of `<path>` in the `.rdf.tex` files. For this, the `rdfprefix` commands have to be placed above the lines that leverage the prefixes in each `.rdf.tex` file. The prefixes registered this way are compatible with all other RDFtex commands. That being said, `rdfproperty` commands draw the greatest benefit from the more concise prefix syntax because they are typically located in blocks of continuous text. To give an example, the `rdfproperty` command for adding the software description from above can be rewritten by means of the `rdfprefix` command as follows:

```
\rdfprefix{mskg} ↩
  {https://example.org/scikg/terms/}

\rdfproperty{RDFtex Implementation} ↩
  {mskg:software_description} ↩
  {RDFtex is a framework enabling...}
```

4.4 Overview of RDFtex's commands and workflow

Summarizing Sect. 4, Table 2 gives an overview of the custom RDFtex commands, their purpose, and their parameters. To give an impression of RDFtex's preprocessing workflow, Fig. 4 shows a flowchart of the steps taking place during the preprocessing. As depicted, every line of every `.rdf.tex` file is processed only once while the lines of the corresponding `.tex` files are written simultaneously, thus combining the export and the import functionality in a single efficient process. When one of the custom RDFtex commands is detected in the current line (regular expressions are employed for the detection), the respective command and its parameters are processed as described in the previous sections.

5 An end-to-end research and publication process

The previous section described the process for producing a LaTeX-based publication using RDFtex. Addressing the bigger picture, this section investigates how this process can be embedded in an end-to-end research and publication process with respect to knowledge graph augmented research. For this, some important open problems have to be discussed beforehand. First, the responsibility for selecting the SciKGs with which the knowledge exchange is practiced and for providing the import templates has to be discussed. Second, it was not addressed yet how the RDF document containing the exported contributions is integrated in the SciKG. Third, the contributions within the SciKG have to be made discoverable by means of adequate tooling such that researchers can use RDFtex's import functionality efficiently. After addressing these points, the end-to-end research and publication process will be outlined.

5.1 SciKG selection and its implications

The remarks in Sect. 4 indicate the central role of the one or even mulitple SciKGs with which the knowledge exchange is practiced regarding RDFtex. It is thus necessary to discuss which parties are responsible for selecting the SciKGs in the research and publication process as well as the implications thereof, e.g., regarding the provision of import templates.

From the import perspective, SciKGs determine the available types of contributions, the SPARQL queries to retrieve the contribution information (if the brute-force solution does not suffice), and the templates for importing contributions. As described in Sect. 4, authors select the SciKG, from which a particular contribution is to be imported by means of the fourth parameter of the `rdfimport` command. While this flexibility solves the problem that contributions might only exist in certain SciKGs, it also leads to problems regarding the import templates. In the end, the publishers determine the design guidelines for the final product, i.e., the PDF publication. To help authors, they typically provide LaTeX publication templates that comply with these guidelines. If a candidate publication does not follow the guidelines, it is either edited or, in the worst case, rejected and thus not published. Hence, it is mandatory that the templates employed to import contributions from the SciKG comply with the guidelines as well. Additionally, different SciKGs use different properties and structures to represent different types of contributions. Because of this, the import templates do not only have to comply with the guidelines of the targeted outlet but also have to be compatible with the SciKG, from which the contribution is to be imported.

For now, authors have to create the import templates for certain outlet SciKG combinations themselves. However, there is a high chance that publishers will provide official import templates in the future if RDFtex becomes recognized enough. For instance, a publisher of computer science papers should provide templates for importing contributions from major general SciKGs as well as major SciKGs within the computer science domain. This way, authors can easily leverage one or more of the supported SciKGs to prepare their publication according to their needs. For SciKGs that do not employ tree-like structures to describe research contributions, additional SPARQL queries are required (cf. Sect. 4).

Table 2 An overview of RDFtex's custom LaTeX commands

Command	Purpose	Parameters
`rdfimport`	Denotes the import of a contribution from a SciKG	A label that will be assigned to the generated content snippet, the citation key, the IRI pointing to the contribution, and the name of the SciKG from which the contribution is to be imported
`rdfexport`	Registers an original contribution for export	A contribution name to reference the export locally, the type of the original contribution, and an optional set of other predicates and objects that are assigned to the contribution entity if applicable
`rdfproperty`	Marks up a property of an original contribution to be exported	The contribution name of the export, the predicate, and the content representing the object
`rdfprefix`	Registers a prefix for abbreviating a namespace or vocabulary with a prefix	The prefix and the written out form of the namespace or vocabulary

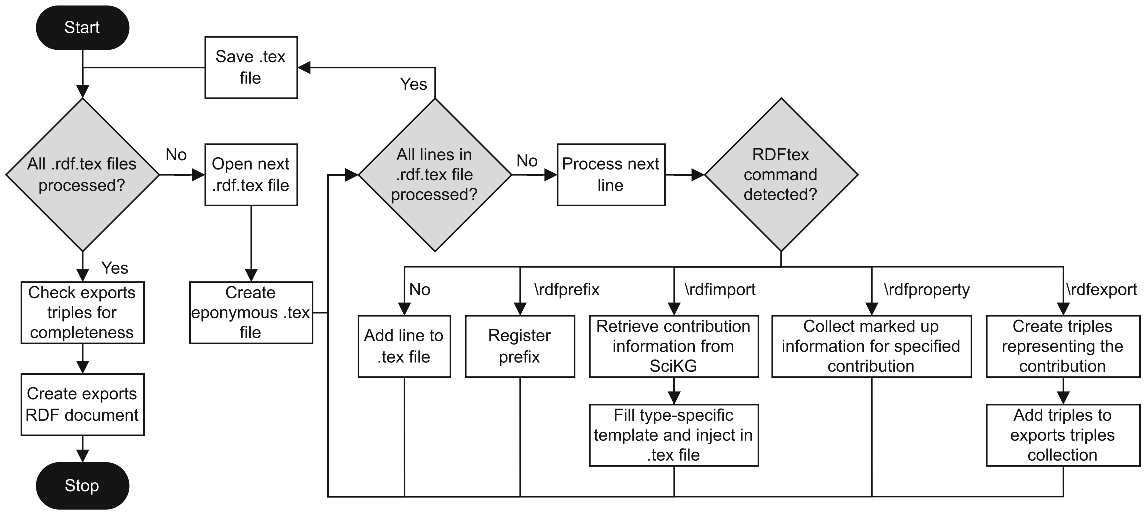

Start
Save .tex file
Yes
All .rdf.tex files processed?
No
Open next .rdf.tex file
All lines in .rdf.tex file processed?
No
Process next line
RDFtex command detected?
Yes
Check exports triples for completeness
Create eponymous .tex file
No
\rdfprefix
\rdfimport
\rdfproperty
\rdfexport
Add line to .tex file
Register prefix
Retrieve contribution information from SciKG
Collect marked up information for specified contribution
Create triples representing the contribution
Create exports RDF document
Fill type-specific template and inject in .tex file
Add triples to exports triples collection
Stop

Fig. 5 A flowchart of RDFtex's preprocessing workflow. For conciseness, the injection of custom LaTeX environments, which are necessary for importing some of the supported contribution types, was omitted

The creation of these queries implies significant effort due to their high number. Hence, it has to be discussed with the community and the involved parties what the best solution to this problem is.

From the export perspective, SciKGs determine the supported contributions types and the contribution information that has to be marked up by the authors in the `.rdf.tex` files for a valid export. Therefore, authors have to know the required properties during the preparation of their publication. The basic solution is to make this information available in the documentation of the SciKGs. Additionally, tooling with auto-completion capabilities that integrates with LaTeX editors could be provided in the future to help authors identifying the necessary properties.

In some cases, authors might want to add custom contribution types or add custom properties to existing contribution types. However, this should not be allowed directly through RDFtex. Instead, authors should only be able to perform such substantial operations through an official interface provided by the target SciKG[15] as this allows for proper authorization, validation, and governance. Once the changes have been made via the SciKG interface, authors can use the new contribution types and/or properties in RDFtex.

[15] The ORKG, for example, provides interfaces to define custom properties for describing contributions, as explained at https://orkg.org/about/19/Templates (accessed 2023/03/08).

In our vision, each outlet has a default SciKG, to which the publications' contributions are added. This has the benefit that all contributions of an outlet can be found in the same SciKG. For instance, on the web page of the IJDL[3] a remark like "Contributions marked up via RDFtex are integrated in the ORKG." could be added right next to "Abstracted and indexed in ACM Digital Library, BFI List,...". If desired, authors can add their contributions to other SciKGs, too. However, this might require additional effort from the authors since the properties employed for describing contributions in the default SciKG might not be compatible with other SciKGs.

5.2 Integration of exported contributions

Running RDFtex's preprocessor yields two types of artifacts: the `.tex` files that result from the preprocessing, which are compiled to PDF afterward, and a single RDF document representing the exported contributions. A publication process incorporating the handling of these artifacts might be as follows: Similar to the typical publication process, the authors submit the final draft of their publication in the form of the compiled PDF document for review. At this stage, the exports RDF document is not relevant yet. If the paper is accepted for publication, however, the integration of the exported contributions has to take place and should be ideally finished as soon as possible. Hence, the authors prepare the camera-ready version of their paper and also finalize the exports document, i.e., check for missing information and wrong properties. Afterward, both are submitted. As usual, it is the publishers' responsibility to handle the paper itself, which includes editing, preparing the actual publication, assigning DOIs, validating the authors' data, etc. These activities yield the finalized metadata of the paper. Together with the exports RDF document (representing the contentual information), the finalized metadata (representing the contextual information) are then relayed to the maintainers of the default SciKG of the outlet. After replacing the preliminary IRIs referring to the paper's entity and contributions with persistent identifiers in the exports RDF document, they finally perform the integration in the SciKG by creating new triples reflecting the contextual information of the publication and adding the triples from the exports RDF document.

At this point, the question arises how (near) duplicates are handled, e.g., what happens when the exports RDF documents of two different publications comprise contributions with similar or even identical information that are to be integrated in the same SciKG. Fundamentally, all contributions are bound to a specific publication by means of the persistent identifier that is assigned by the SciKG maintainers. Hence, contributions with similar or even identical information might be added to a SciKG, but they can be referenced individually, thus posing no problem for querying the SciKG or importing contributions, for example. However, contributions with very similar or even identical content are a quality issue for SciKGs. Therefore, SciKG maintainers should implement mechanisms for deduplication—in the best case, as a part of their data integration process—since they are responsible for performing the actual integration in the SciKG according to the envisaged research and publication process. In the future, RDFtex could be further augmented with a component seeking for identical or similar contributions in a target SciKG to warn authors about (near) duplicates they intend to export.

5.3 Discovery of contribution IRIs

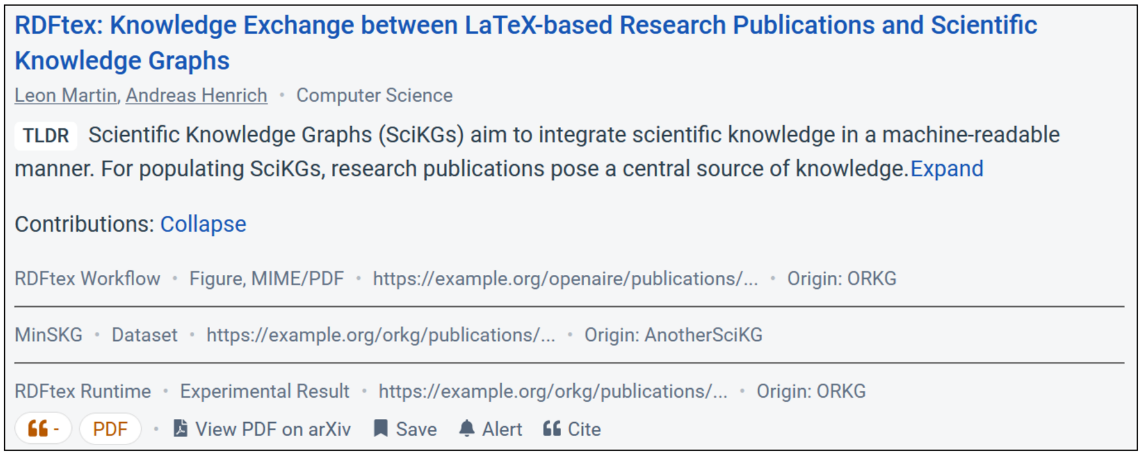
As demonstrated by the code snippets in Sect. 4, knowing the exact IRIs of the contributions to be imported is a prerequisite for using RDFtex's import functionality. Hence, adequate tooling for conveniently obtaining them is mandatory. This corresponds to the remaining requirement for a publication form tailored to knowledge graph augmented research (cf. Sect. 2), which has been left out so far as it addresses the usage context rather than RDFtex itself. In the following, two solutions are discussed.

5.3.1 Special-purpose search engines

Many popular knowledge graphs provide web interfaces for issuing SPARQL queries. Such interfaces can also be used to obtain the IRIs of entities in the graph. However, this form of interaction requires expert knowledge, which interferes with the goal of making RDFtex accessible for researchers from different areas. Instead, users shall be able to formulate queries in natural language. To give an example, the web interface of the ORKG provides this capability via a typical search engine interface. That being said, the result page of ORKG's search does not include the IRIs of the presented search results but proprietary identifiers.[16] Implementation-wise, adding IRIs to the search engine result page is trivial, though, such that this approach poses an easily feasible solution serving the tooling requirement.

The downsides of this solution are two-fold: First, the search engine is confined to one specific SciKG. This can cause problems since there is not one single SciKG comprising all scientific information but rather multiple ones that partially overlap in terms of their content. As a result, researchers might miss important research contributions because they are not present in the searched SciKG. Second, researchers are used to popular academic search engines, thus

[16] For reference, see https://orkg.org/search/rdf (accessed 2023/03/08), which yields the search engine result page for the query *rdf*.


RDFtex: Knowledge Exchange between LaTeX-based Research Publications and Scientific Knowledge Graphs

Leon Martin, Andreas Henrich • Computer Science

TLDR Scientific Knowledge Graphs (SciKGs) aim to integrate scientific knowledge in a machine-readable manner. For populating SciKGs, research publications pose a central source of knowledge.Expand

Contributions: Collapse

RDFtex Workflow • Figure, MIME/PDF • https://example.org/openaire/publications/... • Origin: ORKG

MinSKG • Dataset • https://example.org/orkg/publications/... • Origin: AnotherSciKG

RDFtex Runtime • Experimental Result • https://example.org/orkg/publications/... • Origin: ORKG

Fig. 6 An HTML mockup following the Semantic Scholar design that shows how the contributions of a research publication could be presented on a search engine result page. For conciseness, the example only includes a subset of all exported contributions of the present paper with made-up origin information

making it difficult to establish an alternative special-purpose search engine with less coverage.

5.3.2 Adaptation of academic search engines

Following the closing remarks of the previous paragraph, another conceivable solution for the tooling requirement is to extend the search engine result pages of popular academic search engines like Semantic Scholar[17] and Google Scholar.[18] Demonstrating this approach, Fig. 6 shows how names, types, and IRIs of a publication's contributions could be presented alongside the typical search result information. Clicking the depicted IRIs should send users to the web interface of the SciKG where the respective contribution is represented, thus providing more information. This way, contributions from different SciKGs could be made accessible via a single search interface. At the same time, this implies an increased effort for the search engine providers as more web pages have to be crawled and indexed.

5.4 A three-staged process

Building upon the insights of the previous sections, a simple end-to-end research and publication process incorporating RDFtex can be outlined. The process comprises a preparation stage, a review stage, and a publication stage. The diagram in Fig. 7 summarizes the activities taking place during each stage and compares it to a similar process without RDFtex.

As shown, there are many similarities compared to a typical research and publication process, thus minimizing the risk of discouraging inexperienced users. In addition, the most complex aspect of RDFtex, i.e., finalizing the exports RDF document, does not take place until after the paper has been accepted, thereby ensuring that authors are not unnecessarily burdened during the critical preparation stage. Regarding the final PDF publication, we recommend that publishers include the exports RDF document in the PDF metadata to establish a SciKG-independent link between the publication and the exported contributions. This way, a copy of the exports RDF document is preserved even if the SciKG becomes unavailable in the future such that the exported contributions can be integrated in a suitable surrogate SciKG. In contrast, a SciKG-independent link between the publication and the imported contributions is already established during the preprocessing by means of the explicit reference to the sources in each import template.

6 Discussion

RDFtex and the underlying concept of exchanging knowledge between research publications and SciKGs represent one option to facilitate the move toward knowledge graph augmented research. Our goal was to provide an extension to a well-established publication form that feels natural while providing compatibility to the SciKG world. Using RDFtex's import functionality, authors gain the ability to make direct use of contributions that are already represented in SciKGs. They can also prepare the integration of their own research contributions in SciKGs—and in sequel their usage in other publications—via RDFtex's export functionality, which is vital for being visible in the new research paradigm. Simultaneously, RDFtex introduces only minor changes to the familiar way of preparing a LATEX publication.

From the authors' perspective, leveraging the custom RDFtex commands requires some training and somewhat

[17] https://www.semanticscholar.org (accessed 2023/03/08).

[18] https://scholar.google.com (accessed 2023/03/08).

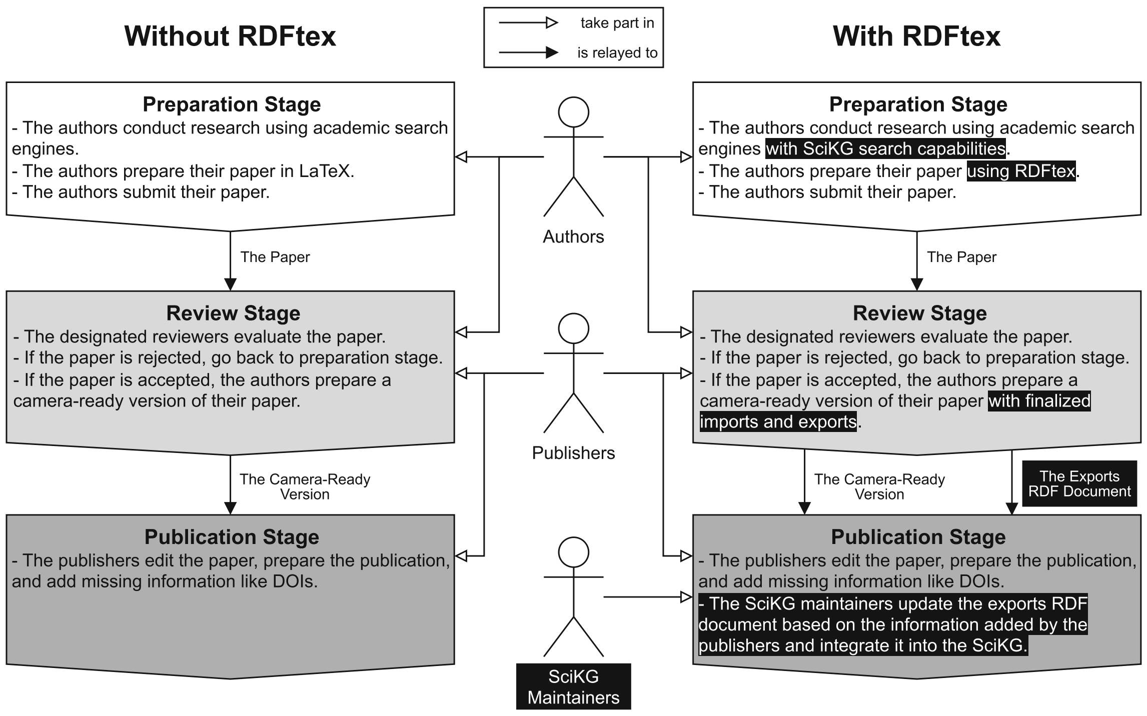

Without RDFtex
take part in
is relayed to
With RDFtex
Preparation Stage
- The authors conduct research using academic search engines.
- The authors prepare their paper in LaTeX.
- The authors submit their paper.
Authors
Preparation Stage
- The authors conduct research using academic search engines with SciKG search capabilities.
- The authors prepare their paper using RDFtex.
- The authors submit their paper.
The Paper
The Paper
Review Stage
- The designated reviewers evaluate the paper.
- If the paper is rejected, go back to preparation stage.
- If the paper is accepted, the authors prepare a camera-ready version of their paper.
Publishers
Review Stage
- The designated reviewers evaluate the paper.
- If the paper is rejected, go back to preparation stage.
- If the paper is accepted, the authors prepare a camera-ready version of their paper with finalized imports and exports.
The Camera-Ready Version
The Camera-Ready Version
The Exports RDF Document
Publication Stage
- The publishers edit the paper, prepare the publication, and add missing information like DOIs.
Publication Stage
- The publishers edit the paper, prepare the publication, and add missing information like DOIs.
- The SciKG maintainers update the exports RDF document based on the information added by the publishers and integrate it into the SciKG.
SciKG Maintainers

Fig. 7 A comparison between a standard three-staged research and publication process on the left with another incorporating RDFtex on the right. The parts with a black background are changes to the process introduced by RDFtex

impairs the readability of the LaTeX code. However, the concise syntax and the introduced prefix command alleviate this problem to a certain degree. Naturally, popular tools for collaboratively preparing publications like Overleaf[19] do not support RDFtex yet. Implementation-wise, integrating RDFtex into such tools is easily feasible, though. Furthermore, one can already set up an automatic build pipeline as described in Sect. 4 and use a version control system to enable collaboration.

In practice, another important point relates to the compatibility with LaTeX publication templates, whose usage is mandatory for many outlets. The export functionality of RDFtex is actually compatible with most LaTeX templates out of the box since the export commands, i.e., `rdfexport` and `rdfproperty`, are removed from the files during preprocessing. Regarding the import functionality, there might occur some issues due to the static nature of the employed import templates. For instance, the width of imported figures should often span an entire column in double-columned layouts while such a configuration might result in exceedingly large figures in single-columned layouts. Currently, the two options to solve this problem are to either customize the import templates in RDFtex's source code or to adjust the `tex` files resulting from the preprocessing manually. Most of the time, the former option is recommended because any changes made to the `tex` files are lost when the preprocessor is issued again. Nevertheless, the latter option can be useful for putting the finishing touches on the publication. Another option that can be explored in the future is the addition of optional customization parameters to the `rdfimport` command such that manual changes of the finalized `tex` files are required less often. However, it is important to find a good level of customizability in collaboration with the community, as too many additional parameters would impede the readability of the LaTeX code.

Due to its plug-and-play nature, RDFtex can also be added to a LaTeX project retrospectively. For this, authors have to create copies of their existing `.tex` files with the `.rdf.tex` file extension. Then, after pointing the preprocessor to the project directory, they can already use the custom RDFtex commands and issue the production of the `.tex` as described above.

6.1 Runtime evaluation

In accordance with Fig. 5, our proof-of-concept implementation demonstrates that it is feasible to scan the `.rdf.tex` files for the custom RDFtex commands line by line and only once, while the lines of the `.tex` files are written simultaneously, thus requiring linear time. To assess the runtime of RDFtex, the preprocessor was tested on three LaTeX projects with different characteristics using the MinSKG for importing and exporting contributions. P_1 is the project underlying

[19] https://www.overleaf.com (accessed 2023/03/08).

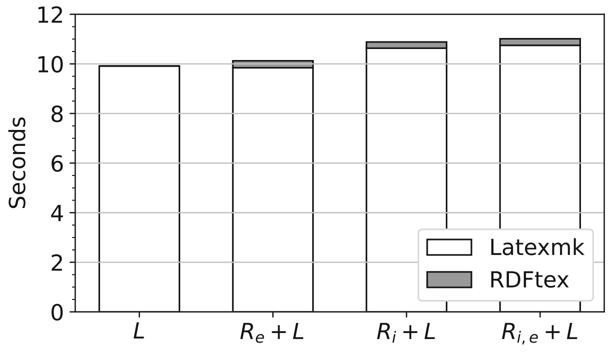
the present paper itself and therefore comprises a realistic amount of content for a research publication. Overall, this paper comprises two imports, namely Fig. 1 and Definition 1. Furthermore, the proof-of-concept RDFtex implementation as a software artifact, Figs. 2, 3, 4, 5, 6 and 7, the MinSKG as a dataset, and the experimental results of the runtime evaluation that follow next have been exported as original contributions.

P_2, the second project considered in the assessment, is an exemplary project that comprises one import and one export for each supported contribution type, i.e., five cases each, with very little content aside from the imported and exported contributions. The last project P_3 is a derivation of P_2, where each import and export is included 20 times in a row, thus resulting in a total of 100 imports and 100 exports. For reproducibility, P_2 and P_3 as well as all benchmark scripts can be found in the provided repository[2]. To measure the impact of the import and the export functionality on the total runtime, four configurations were investigated:

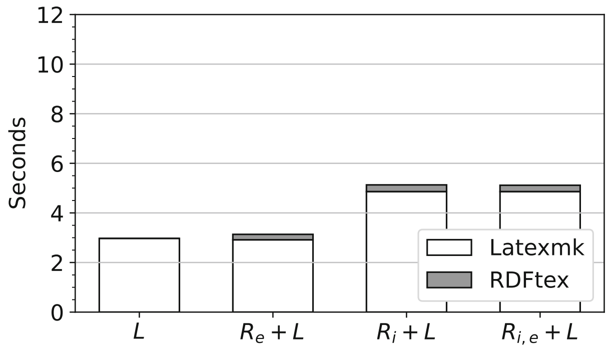
1. Latexmk [8] only; representing the baseline (L)
2. RDftex with only the export functionality enabled in combination with Latexmk ($R_e + L$)
3. RDftex with only the import functionality enabled in combination with Latexmk ($R_i + L$)
4. RDftex with both the import and the export functionality enabled in combination with Latexmk ($R_{i,e} + L$)

For the experiments, the preprocessor was executed on a system featuring an Intel Core i7-1185G7 processor at stock clock speeds, 32 GB of DDR4 RAM, an NVMe SSD, and a wired internet connection. To take the network overhead into account, the MinSKG was deployed on a server. In total, the preprocessor was run 100 times per configuration per project. Figure 8 presents the average time required to produce the PDFs using RDFtex and Latexmk with respect to the described LaTeX projects and configurations. The significant runtime difference regarding Latexmk observed for P_2 and P_3 is caused by the fact that the configurations with the import functionality enabled generate `.tex` files with considerably more content due to the imported contributions in relation to the other content of these particular projects. In contrast, P_1 comprises much content apart from the imported contributions, thus resulting in a more balanced Latexmk runtime regardless of whether the import functionality is enabled or not. The results show that RDFtex increases the time to produce PDF files only by a fraction, even for the third project with its exceptionally large number of imports and exports. Furthermore, the runtime of the export functionality turns out to be negligible compared to the runtime of the import functionality.

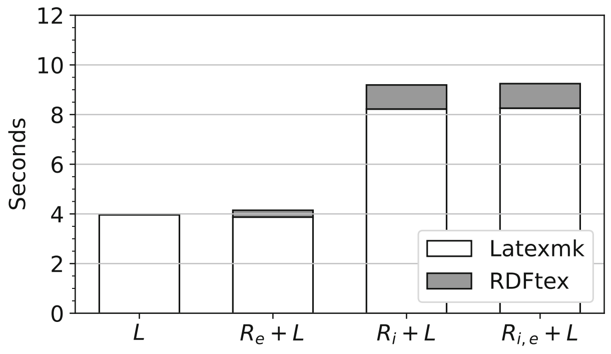
Still, the employed MinSKG is a makeshift SciKG, which results in faster than normal SPARQL query response times.



(a) P_1 (2 imports/9 exports, full publication content)



(b) P_2 (5 imports/5 exports, little content aside from imported/exported contributions)



(c) P_3 (100 imports/100 exports, little content aside from imported/exported contributions)

Fig. 8 Runtime of RDFtex and Latexmk to produce PDFs of three different LaTeX projects P_1–P_3 using four different configurations. Each bar represents the average runtime across 100 runs

To get an impression of the runtime differences that would result from the change to an actual SciKG, another set of experiments investigating the query response times of the ORKG in comparison with the query response times of the MinSKG was conducted. For this, three different contributions in the MinSKG and three different contributions in the ORKG were queried 100 times each using the brute-force

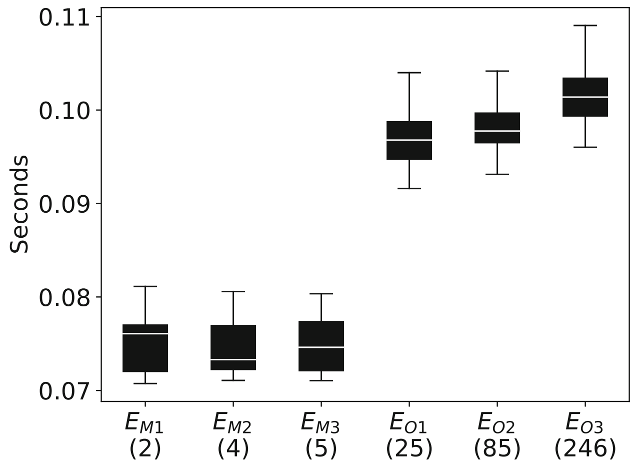


Fig. 9 Query response times of the MinSKG and the ORKG when querying contribution entities using the brute-force solution across 100 runs. The labels on the *x*-axis denote the SciKG that is queried (*M* for MinSKG, *O* for ORKG) and the entity number in subscript. The numbers in brackets denote the number of triples yielded for the respective entity. The few outliers that occurred are not shown in this plot for readability

solution (cf. Sect. 4.1), which yields the tree-like structures describing the contributions. The exact IRIs of the contributions can be found in the provided repository[2]. The experiments were conducted on the same system as before and the MinSKG was again deployed on a server to take the network overhead into account. For fairness, both the MinSKG SPARQL interface and the ORKG SPARQL interface[11] were configured to return the results in the Turtle format.

Figure 9 shows that the query response times of the ORKG are typically about 40% longer than the query response times of the MinSKG. Considering the drastically higher number of triples describing the contributions in the ORKG,[20] this is a positive outcome. Based on the results, an increase in RDFtex's overall runtime of about 40% can be expected when the ORKG is employed instead of the MinSKG since the increase applies to each import and the runtime of the export functionality is negligible. Latexmk's runtime is not affected by the employed SciKG such that the total time required to produce a PDF using RDFtex and Latexmk is not increased significantly.

Using P_3 as an example, which represents the worst project regarding RDFtex's runtime in the set, a typical run of RDFtex with the import and export functionality enabled takes about 0.9 s. The subsequent compilation of the `.tex` files using Latexmk typically takes about 8.3 s, resulting in a total runtime of 9.2 s. Adding the estimated 40% increase to RDFtex's runtime yields a total runtime of about 9.6 s, a negligible difference from the users' perspective. Nevertheless, caching could be implemented in the future such that the same entities are not queried multiple times, thus reducing the runtime and the load on the SciKG servers to a certain degree.

[20] The MinSKG only includes the contribution information required for the import templates, which causes the high difference in the number of triples.

Table 3 Distribution of the test subjects with respect to their academic level, field, and self-estimated LaTeX proficiency level (LPL) on a scale of 1 (the lowest level) to 5 (the highest level)

LPL	Undergraduate Student	Graduate Student	Research Associate
5	–	–	–
4	–	$2 \times CS$	$1 \times CS$
3	$3 \times CS$	–	$1 \times CS, 1 \times AP$
2	–	$1 \times AP$	–
1	$1 \times CS$	–	–

CS denotes the field of *Computer Science* and *AP* the field of *Astrophysics*

6.2 User study

To obtain initial user feedback, a small user study with $N = 10$ test subjects was conducted. Table 3 shows the distribution of the subjects with respect to their academic level, their field, and their self-estimated LaTeX proficiency level. Although the set of subjects is biased toward computer science, it still covers different academic and LaTeX proficiency levels, thus sufficing for getting an impression of the strengths and weaknesses of RDFtex in its current state.

In a controlled test environment, the subjects were asked to solve the following tasks by means of our implementation on a blank `.rdf.tex` file using the think-aloud method:

1. Define a prefix that can be used to abbreviate the full IRIs of the MinSKG.
2. Given its IRI, import a specific definition and generate the PDF.
3. Import a specific figure whose IRI is, however, not known and generate the PDF. (The textual description of the figure was provided such that the figure can be identified in the MinSKG.)
4. Write an arbitrary definition, mark it up for export, and generate the PDF as well as the exports RDF document.

During the tests, the test subjects had access to the documentation of the provided repository[2] and were allowed to use a search engine to solve occurring problems, if necessary. Beginning with the first task, all subjects that reported a LaTeX proficiency level of 3 or higher were able to solve the task easily, except one of the undergraduate students with a level of 3 who had problems using the command line to enter the RDFtex commands at first. The two subjects with a level of 1 and 2 had difficulty setting up the initial LaTeX structure, an aspect unrelated to the usability of RDFtex. Despite its utility, four subjects did not leverage the prefix syntax for the other tasks, though. The low complexity of the test project might be the reason for this.

The second task was solved by the subjects in the least amount of time without any noteworthy problems. This is likely due to the fact that they were getting used to RDFtex's documentation during the first task and the low complexity of the import functionality. In fact, the import functionality was commended several times for its simplicity. Some subjects further showed feelings of satisfaction, when they generated LaTeX content with a single RDFtex import command.

In contrast, the test subjects had considerable problems finding the correct IRI in the serialized MinSKG to solve the third task. Overall, only three subjects were able to identify the IRI in a reasonable amount of time. Two of the remaining subjects gave up and asked for help. However, this result was expected and merely confirms that there is a strong need for tooling to obtain the necessary IRIs. Thus, the solutions discussed in Sect. 5 must be explored in the future to help users obtaining IRIs from SciKGs. After they either found or were provided the IRI in question, the test subjects were able to apply the import functionality as efficiently as during the second task, though.

Proceeding to the fourth task, the export functionality proved to be more complex for the test subjects compared to the import functionality due to the larger number of involved commands. Nevertheless, the task was solved in a reasonable amount of time by most subjects. One of the subjects from the research associate group proceeded writing and exporting contributions of other types and noted the intuitiveness of the invisible markup, as she called it, that the export-related RDFtex commands represent since they are removed during the preprocessing.

Aside from the task specific remarks, three subjects mentioned that they would use the framework, especially the export functionality, only when the benefits of knowledge graph augmented research become noticeable. Other subjects reported that they would gladly use the novel import functionality of RDFtex but questioned the current availability of the contributions relevant for their research in SciKGs. Both statements are reasonable and highlight the importance of the shift toward of knowledge graph augmented research regarding the adoption of RDFtex.

6.3 Compatibility with existing technologies

When it comes to the compatibility between RDFtex and existing technologies, one has to consider the export and the import perspective. Regarding the export perspective, the previous sections demonstrated that RDFtex's export commands themselves are not bound to a particular SciKG because the properties are specified directly in the commands. As described in Sect. 3, SciKGTeX employs properties that are already available in the ORKG as well as custom properties to annotate contributions. The annotations also use the RDF format, thus facilitating the integration in the ORKG. Apart from being limited to the ORKG, SciKGTeX therefore follows similar goals as RDFtex's export functionality. As a result, compatibility between SciKGTeX and RDFtex's export functionality could definitely be established but would introduce redundancy rather than provide a tangible benefit. In contrast, sTeX(+) and SALT do not aim to interoperate with SciKGs. Hence, there is no SciKG that particularly uses the properties introduced by them. Due to the flexibility of RDF, it is theoretically possible to implement a unified export format that combines the information marked up via RDFtex's export commands and the other technologies, though. For example, the XHTML+MathML+RDFa export format of sTeX+ could be extended to also include a human-readable representation of the contributions exported using RDFtex. For this, templates could be used that are populated with the information from RDFtex's exports RDF document.[21]

Regarding the import perspective, RDFtex's import functionality requires a SciKG from which the contribution information is retrieved. Even though the marked up information of sTeX(+) and SALT is not reflected in a SciKG, it would be possible to implement import templates that incorporate the information extracted using these technologies, too. Nevertheless, there has to be a database or other source providing the information such that the preprocessor can retrieve it. Despite being interoperable with a SciKG for exporting contributions, SciKGTeX does not provide means of importing contributions in LaTeX documents. As demonstrated by RDFtex, implementing import templates that leverage arbitrary properties to create snippets of LaTeX content is feasible, though. Hence, RDFtex can be adapted to handle properties employed by SciKGTeX, as well. However, only the properties that are registered in the ORKG and not the custom ones specified directly through SciKGTeX should be supported for the reasons stated in Sect. 5.

6.4 Future work

Aside from the hints at future work already mentioned in the previous sections, one obvious next step for the RDFtex framework is the adaptation to an actual SciKG. In this regard, the ORKG is eligible as it offers both contextual and contentual information of research publications and provides a SPARQL interface[11], which is required for the import func-

[21] Even though the target language, i.e., LaTeX vs. XHTML+MathML+RDFa, differs, such a template-based approach for generating content actually reminds of RDFtex's import functionality.

tionality. After the adaptation, a more elaborate user study can be conducted. In the end, RDFtex is supposed to be as usable for persons without linked data know how as for people with a linked data background. Otherwise, RDFtex would remain a niche technology. Hence, this study is supposed to investigate the usability of RDFtex when used with an actual SciKG and, by selecting a more diverse set of users familiar with LATEX as test subjects, to identify optimization points for the syntax as well as the workflow.

Regarding the application of RDFtex, note that it would also be possible to extend the scope of RDFtex beyond SciKGs, i.e., knowledge graphs from other domains or more general knowledge graphs. Regarding the import functionality, this requires the addition of further templates that are able to digest the properties of the new entities to be imported. To implement the export functionality, appropriate processes for integrating the exports RDF document into the targeted knowledge graph are necessary.

Another interesting lead on future work relates to the coding language that is employed to prepare the publication. Recent editions of conferences like the ESWC22[22] and the ISWC2022[23] give authors the opportunity to submit HTML-based publications. As shown in Sect. 3, there already are technologies like RDFa for adding semantic markup information to HTML content, which could be leveraged for implementing an export functionality. Furthermore, parsing and manipulating HTML code, which is required for the import functionality, is a common task. Therefore, implementing a framework similar to RDFtex enabling the knowledge exchange between HTML-based publications and SciKGs by means of importing and exporting contributions is feasible, too. Actually, even a technology that enables the knowledge exchange between publications prepared using WYSIWYG editors like Microsoft Word[24] and a SciKG is conceivable. That being said, the implementation of such a technology raises new questions due to the significant differences between LATEX and WYSIWYG editors, but the basic concept of exchanging knowledge with a SciKG via a few custom commands could still be adopted from the RDFtex framework.

7 Conclusion

The present paper proposed RDFtex, a framework for producing LATEX-based research publications comprising imports and exports from and to SciKGs. For this purpose, RDFtex introduces four custom LATEX commands. Furthermore, the integration of RDFtex into a research and publication process and its compatibility with existing technologies have been discussed. For the evaluation, the runtime of RDFtex's preprocessor was investigated and a small user study was conducted.

The envisaged paradigm shift toward knowledge graph augmented research, which motivated this work, poses a significant change for the research culture. It depends on the scientific community if this change will actually take place, though. If so, we think that RDFtex is a promising approach to facilitate the move toward the new research paradigm since it only introduces minor differences compared to the preparation of traditional LATEX-based publications while narrowing the gap between papers and RDF.

[22] https://2022.eswc-conferences.org (accessed 2023/03/08).

[23] https://iswc2022.semanticweb.org (accessed 2023/03/08).

[24] https://www.microsoft.com/en-ww/microsoft-365/word (accessed 2023/03/08).

Funding Open Access funding enabled and organized by Projekt DEAL.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. White, K.: Publications output: US trends and international comparisons. Science & Engineering Indicators 2020. nsb-2020-6. National Science Foundation (2019). https://ncses.nsf.gov/pubs/nsb20206. Accessed 08 Mar 2023
2. Auer, S., Kovtun, V., Prinz, M., Kasprzik, A., Stocker, M., Vidal, M.: Towards a knowledge graph for science. In: Proceedings of the 8th International Conference on Web Intelligence, Mining and Semantics, WIMS 2018, pp. 1–116. ACM, Novi Sad, Serbia (2018). https://doi.org/10.1145/3227609.3227689
3. Jaradeh, M.Y., Oelen, A., Farfar, K.E., Prinz, M., D'Souza, J., Kismihók, G., Stocker, M., Auer, S.: Open research knowledge graph: next generation infrastructure for semantic scholarly knowledge. In: Proceedings of the 10th International Conference on Knowledge Capture, K-CAP 2019, November 19-21, 2019, pp. 243–246. ACM, Marina Del Rey, CA, USA (2019). https://doi.org/10.1145/3360901.3364435
4. Luan, Y., He, L., Ostendorf, M., Hajishirzi, H.: Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pp. 3219–3232. Association for Computational Linguistics, Brussels, Belgium (2018). https://doi.org/10.18653/v1/d18-1360
5. Martin, L., Jegan, R., Henrich, A.: On the form of research publications for use in scientific knowledge graphs. In: Wissensorganisation 2021: 16. Tagung der Deutschen Sektion der Inter-

nationalen Gesellschaft Für Wissensorganisation (ISKO) (WissOrg'21), Online (accepted for publication 2021)

6. Martin, L., Henrich, A.: RDFtex: Knowledge exchange between LaTeX-based research publications and scientific knowledge graphs. In: Silvello, G., Corcho, Ó., Manghi, P., Nunzio, G.M.D., Golub, K., Ferro, N., Poggi, A. (eds.) Linking Theory and Practice of Digital Libraries - 26th International Conference on Theory and Practice of Digital Libraries, TPDL 2022, September 20-23, 2022, Proceedings. Lecture Notes in Computer Science, vol. 13541, pp. 26–38. Springer, Padua, Italy (2022). https://doi.org/10.1007/978-3-031-16802-4_3
7. Cyganiak, R., Hyland-Wood, D., Lanthaler, M.: RDF 1.1 concepts and abstract syntax (2014). W3C. https://www.w3.org/TR/rdf11-concepts. Accessed 08 Mar 2023
8. Ehrlinger, L., Wöß, W.: Towards a definition of knowledge graphs. In: Martin, M., Cuquet, M., Folmer, E. (eds.) Joint Proceedings of the Posters and Demos Track of the 12th International Conference on Semantic Systems - SEMANTiCS2016 and the 1st International Workshop on Semantic Change & Evolving Semantics (SuCCESS'16) Co-located with the 12th International Conference on Semantic Systems (SEMANTiCS 2016). CEUR Workshop Proceedings, vol. 1695. CEUR-WS.org, Leipzig, Germany (2016). http://ceur-ws.org/Vol-1695/paper4.pdf
9. Saleem, M., Szárnyas, G., Conrads, F., Bukhari, S.A.C., Mehmood, Q., Ngomo, A.N.: How representative is a SPARQL benchmark? An analysis of RDF triplestore benchmarks. In: Liu, L., White, R.W., Mantrach, A., Silvestri, F., McAuley, J.J., Baeza-Yates, R., Zia, L. (eds.) The World Wide Web Conference, WWW 2019, pp. 1623–1633. ACM, San Francisco, CA, USA (2019). https://doi.org/10.1145/3308558.3313556
10. W3C SPARQL Working Group: SPARQL 1.1 overview (2013). W3C. https://www.w3.org/TR/sparql11-overview. Accessed 08 Mar 2023
11. Beckett, D., Berners-Lee, T., Prud'hommeaux, E., Carothers, G.: RDF 1.1 Terse RDF Triple Language (2014). W3C. https://www.w3.org/TR/turtle. Accessed 08 Mar 2023
12. Manghi, P., Bardi, A., Atzori, C., Baglioni, M., Manola, N., Schirrwagen, J., Principe, P.: The OpenAIRE research graph data model (2019)
13. Ernst, P., Siu, A., Weikum, G.: Knowlife: a versatile approach for constructing a large knowledge graph for biomedical sciences. BMC Bioinform. **16**, 157–115713 (2015)
14. Schreiber, G., Raimond, Y.: RDF 1.1 primer (2014). W3C. https://www.w3.org/TR/rdf11-primer. Accessed 08 Mar 2023
15. Kuhn, T., Meroño-Peñuela, A., Malic, A., Poelen, J.H., Hurlbert, A.H., Ortiz, E.C., Furlong, L.I., Queralt-Rosinach, N., Chichester, C., Banda, J.M., Willighagen, E.L., Ehrhart, F., Evelo, C.T.A., Malas, T.B., Dumontier, M.: Nanopublications: A growing resource of provenance-centric scientific linked data. In: 14th IEEE International Conference on e-Science, e-Science 2018, pp. 83–92. IEEE Computer Society, Amsterdam, The Netherlands (2018). https://doi.org/10.1109/eScience.2018.00024
16. Project Xanadu: Project Xanadu (2007). Project Xanadu. https://www.xanadu.net. Accessed 08 Mar 2023
17. PubGenius Inc.: Discover, create, and publish your research paper (2023). PubGenius Inc. https://typeset.io. Accessed 08 Mar 2023
18. Herman, I., Adida, B., Sporny, M., Birbeck, M.: RDFa 1.1 primer (2015). W3C. https://www.w3.org/TR/rdfa-primer. Accessed 08 Mar 2023
19. Kohlhase, A., Kohlhase, M., Lange, C.: S^{t}e^{X}+: a system for flexible formalization of linked data. In: I-SEMANTICS. ACM, Online (2010)
20. W3C OWL Working Group: OWL 2 Web Ontology Language document overview (second edition) (2013). W3C. https://www.w3.org/TR/owl2-overview. Accessed 08 Mar 2023
21. Groza, T., Handschuh, S., Möller, K., Decker, S.: SALT - Semantically annotated LaTeX for scientific publications. In: Franconi, E., Kifer, M., May, W. (eds.) The Semantic Web: Research and Applications, 4th European Semantic Web Conference, ESWC 2007, Proceedings. Lecture Notes in Computer Science, vol. 4519, pp. 518–532. Springer, Innsbruck, Austria (2007). https://doi.org/10.1007/978-3-540-72667-8_37
22. Technische Informationsbibliothek (TIB): SciKGTeX - ORKG (2023). Technische Informationsbibliothek (TIB). https://orkg.org/about/33/SciKGTeX. Accessed 08 Mar 2023
23. Collins, J.: Latexmk (2023). Pennsylvania State University. https://personal.psu.edu/~jcc8/software/latexmk. Accessed 08 Mar 2023

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.