

A text extraction software benchmark based on a synthesized dataset

Kresimir Duretec, Andreas Rauber, and Christoph Becker

Version Published PDF

Citation (published version) Kresimir Duretec, Andreas Rauber, Christoph Becker (2017). A text extraction software benchmark based on a synthesized dataset. Proceedings of the 17th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL). Toronto, ON, Canada. IEEE.

Publisher's Statement Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

How to cite TSpace items

Always cite the **published version**, so the author(s) will receive recognition through services that track citation counts, e.g. Scopus. If you need to cite the page number of the **author manuscript from TSpace** because you cannot access the published version, then cite the TSpace version **in addition to** the published version using the permanent URI (handle) found on the record page.

This article was made openly accessible by U of T Faculty.
Please [tell us](#) how this access benefits you. Your story matters.

A text extraction software benchmark based on a synthesized dataset

Kresimir Duretec
Vienna University of Technology
Vienna, Austria
kresimir.duretec@tuwien.ac.at

Andreas Rauber
Vienna University of Technology
Vienna, Austria
rauber@ifs.tuwien.ac.at

Christoph Becker
University of Toronto
Toronto, Canada
christoph.becker@utoronto.ca

ABSTRACT

Text extraction plays an important function for data processing workflows in digital libraries. For example, it is a crucial prerequisite for evaluating the quality of migrated textual documents. Complex file formats make the extraction process error-prone and have made it very challenging to verify the correctness of extraction components. Based on digital preservation and information retrieval scenarios, three quality requirements in terms of effectiveness of text extraction tools are identified: 1) is a certain text snippet correctly extracted from a document, 2) does the extracted text appear in the right order relative to other elements and, 3) is the structure of the text preserved. A number of text extraction tools is available fulfilling these three quality requirements to various degrees. However, systematic benchmarks to evaluate those tools are still missing, mainly due to the lack of datasets with accompanying ground truth. The contribution of this paper is two-fold. First we describe a dataset generation method based on model driven engineering principles and use it to synthesize a dataset and its ground truth directly from a model. Second, we define a benchmark for text extraction tools and complete an experiment to calculate performance measures for several tools that cover the three quality requirements. The results demonstrate the benefits of the approach in terms of scalability and effectiveness in generating ground truth for content and structure of text elements.

KEYWORDS

text extraction, software benchmark, software testing, model driven engineering, digital preservation, performance measures, dataset, ground truth

ACM Reference format:

Kresimir Duretec, Andreas Rauber, and Christoph Becker. 2017. A text extraction software benchmark based on a synthesized dataset. In *Proceedings of ACM/IEEE-CS Joint Conference on Digital Libraries 2017, Toronto, Canada, June 2017 (JCDL2017)*, 10 pages. DOI:

1 INTRODUCTION

Text extracted from digital objects forms the basis of various data management and analysis activities. In digital libraries, text extraction is often used as a part of larger data processing workflows. These workflows cover a variety of application use cases that range

from domains such as information retrieval (e.g. indexing and searching) to digital preservation (e.g. verifying that values in invoices are not lost or changed during the migration process). A number of software components address text extraction from different types of digital objects (e.g. images, page based documents, web pages, video). The scope of those components ranges from extracting all available text to specialized uses cases such as identifying relevant text in a web page or extracting tables or captions from scientific articles.

The quality of text extraction components has a direct impact on the quality of the data processing workflows they are used in. In digital preservation, additional steps need to be taken to minimize the risks data migration can introduce. For example, migrating textual documents from one file format to another must respect the original type [33]. As it is hard to accurately quantify the quality of the migration workflow [21], the original files will need to be kept. The first step in evaluating the quality of such a migration is to evaluate the text extraction components, which can then further be used to evaluate migration components. However, a proper software benchmark [27] that would allow the rigorous evaluation and sharing of evidence about individual components' performance is still missing [7].

The goal of this paper is to move forward the initiatives in benchmarking text extraction from documents. We propose establishing a software benchmark for text extraction components. A number of commercial and open source components are available and the community is still requiring better tools, both important benchmarking preconditions indicate a good timing for investing effort in establishing such a benchmark [27].

The main obstacle to creating effective text extraction benchmarks is the lack of proper datasets with accompanying ground truth. Current approaches of annotating available documents have shown to be expensive [15] and limited to narrowly defined domains for which annotations can readily be collected. Text extraction components are homogeneous in their functionality, with minimal amount of human interaction. The main challenge in testing functional correctness of those components lies in the complex input space. An effective approach to assembling datasets must therefore be able to cover a variety of documents in different file formats (DOCX, ODT, PDF, ...) created on different platforms (Windows, Linux,...) with different software (MS Word, LibreOffice ...), and contain text encoded in different text elements with varying formatting and layout.

In this paper we propose a text extraction benchmark based on a synthesized dataset and ground truth. The basis of our generation approach is the proof of concept described in [4] which we implement, extend and evaluate in the context of a text extraction

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

JCDL2017, Toronto, Canada

© 2017 Copyright held by the owner/author(s). ...\$15.00

DOI:

benchmark. We define a text extraction benchmark through five benchmark components [7] for the scope of general documents stored in file formats such as DOCX, ODT and PDF.

The contribution of this paper is thus two-fold. First, we implement a dataset generation method that generates data sets that can be extended and shared with no restrictions. Second, we define a text extraction benchmark and evaluate a set of tools using the dataset. We discuss the benefits and limitations of the approach in comparison to the prevailing approach of sampling and annotation.

In Section 2 we provide an overview of recent work on evaluating text extraction tools and identify different categories and main challenges. Furthermore we provide an overview of the theory in software benchmarking and briefly touch on the key aspects from software testing that provide important concepts for the dataset generation method. Section 3 defines the five main components for the text extraction benchmark and identifies key requirements. Section 4 and 5 respectively provide implementation details of the dataset generation process and software evaluation process. The benchmark results are reported in Section 6, followed by a detailed discussion of the approach in Section 7.

2 BACKGROUND

2.1 Text extraction evaluation and test datasets

Text extraction solution range from research prototypes to production ready solutions. These cover a range of tasks, from very narrow functions such as extracting document titles [13] or references [23] to broad features for extracting text from general documents in different file formats such as DOCX, ODT and PDF¹.

Generic tools are often used as a part of data processing workflows. To enable detection of more complex text units such as paragraphs and sentences, Tiedemann [31] builds text processing features on top of the outputs of tools such as Apache Tika². Similarly, Liu et al. [17] describe a system able to extract tables and table metadata from PDF documents that uses PDFBox³ for extracting raw text, which is then further processed to identify tables. Generic text extraction tools affect the output of tools relying on them. For example, the table extraction tool in some cases might miss a table because the underlying text extraction tool failed to correctly extract the table caption [17]. In digital preservation, it is important to ensure that migrated documents are an authentic reproduction of the original. Tools such as text extraction have an important role in enabling the comparison between the migrated version and the original [18]. Without a reliable evaluation of text extraction tools, the comparison they enable cannot be trusted.

Text extraction tools have been evaluated to differing degrees, and evaluations have covered various software quality characteristics. Quality aspects that are often considered address *reliability* (including exceptions, tool failures, tool hangs) and *functional correctness* (including missing or garbled text). Allison [1] identifies several approaches for evaluating functional correctness: comparing the tool output to ground truth, selecting random sub-samples

and manually reviewing results, and performing comparative evaluation based on the results from several tools. However, the evaluation results provided by Allison [1] focus only on reliability aspects, where the comparative study of the outputs of two versions of Apache Tika showed a lower number of exceptions with the newer version of the tool.

Hu et al. [12][13] manually provided labels containing title text in order to evaluate their title extraction approach. Similarly, Pasternack [24] provided manual labels for article extraction. Liu et al. [17] have performed a five-user evaluation study to evaluate the quality of their table detection tool. Deeper evaluations are required, but are limited by the available effort [1].

Test data collection is the most expensive component in creating proper evaluations [30]. To evaluate text extraction tools, many evaluations rely on existing datasets. One such dataset is the Govdocs dataset [11], which has become a defacto standard for many evaluation works in fields such as digital preservation. The data set is open and of considerable size and diversity in covered formats. In the domain of text extraction, Allison and Herzog [1] have used it to evaluate the number of failures and exceptions during the text extraction. However, several factors limit its usefulness in evaluating functional correctness. Above all, the main limitation is its lack of verified ground truth. Furthermore, due to its age the content does not cover new file formats and focuses on the government domain. This limits the dataset's use to certain use cases.

Other datasets with ground truth include CleanEval⁴ and L3S-GN1⁵. Unfortunately, these datasets cover only web page documents and are tailored for narrow needs of distinguishing real web text content from boilerplate text.

A common characteristic shared by all these datasets is that they were assembled using a similar process, which is based on randomly selecting examples from a single or several sources and providing manual annotations for the whole or a part of the dataset [13][17][25]. In some cases, the annotation process is supported by software tools with the goal of making the process more efficient [12]. In some cases, datasets annotations are done by other tools such as the one developed and used by Pasternack et al. [24]. However, since such annotations are computed by tools that are not evaluated themselves, their suitability for evaluation purposes is questionable [4, 5]. This has been recognized by Pasternack et al. [24] as well, and they used annotations created in such a way only for training purposes and provided manually generated annotations for the evaluation purposes. Manual annotation, even though considered to be expensive [15][4], thus is still the prevailing approach in creating evaluation datasets.

The opposite approach starts from annotations and synthesizes artificial data sets to match the annotations. This approach of synthesizing artificial datasets has been explored to a limited extent only. In the domain of file format identification, Spencer [28] provided a method of generating a skeleton corpus based on the file format signatures. However, the method and its corpus ignore all complexities of the format itself. Becker & Duretec [4] proposed a generic data generation framework based on the model driven

¹See e.g. <https://tika.apache.org/http://www.foolabs.com/xpdf/>

²<https://tika.apache.org/>

³<https://pdfbox.apache.org/>

⁴<https://cleaneval.sigwac.org.uk/>

⁵<http://www.l3s.de/~kohlschuetter/boilerplate/>

engineering methods. This framework represents the basis for the work presented here.

2.2 Software benchmarking and software testing

A software benchmark is “a test or set of tests used to compare the performance of alternative tools or techniques” [27]. The three main benchmark components defined in [27] have been further tailored to data processing needs in [7]:

- **Motivating comparison** identifies the main quality comparison between tools that motivates the benchmark.
- **Function** defines what a benchmarked tool needs to perform and what kind of output is expected.
- **Dataset** in the case of data processing tools includes a set of test cases on which the evaluation will be performed.
- **Ground Truth** provides for each test case a correct (expected) answer. In software testing, this concept is also known as the *test oracle* [29].
- **Performance measures** operationalize the motivating comparison by providing qualitative or quantitative values that address certain quality characteristics.

A benchmark should be accessible (i.e. all components must be available), should cover relevant tasks, and should not prefer certain solution approaches over others [27]. In the case of text extraction, the benchmark quality of test data will have a large impact on the quality and effectiveness of the benchmark as a whole. Initial quality criteria have been provided by Neumayer et al. [22] and Fetherston et al. [10]. However, these criteria do not cover specific technical aspects such as coverage and representativeness. This lack of test data quality models is a recognized problem for this domain [5], and more research is needed to define a model of quality characteristics and metrics.

Considering the defined problem and existing approaches in software testing, several additional aspects arise.

The problem of ground truth in software testing is also known as the test oracle problem, where a *test oracle* is defined as a mechanism that distinguishes correct from incorrect software behaviour [29][3]. This problem has received considerable attention, but in the data processing domain, much remains to be done [19][20].

In model-based testing, the test generation generates the required test cases based on the defined model(s) and test selection criteria. Approaches range from random-based generation and search-based generation to formal theorem proving and constraint solving [2][32].

2.3 Summary

The development of high quality test datasets is a key concern for any kind of data processing evaluation [9][5]. This has been a well recognized challenge for years in domains such as digital preservation[22]. However, several initiatives to address the challenge have done it only to a limited degree. The available datasets therefore belong to one of the two categories. Either they are of significant size and feature coverage such as the Govdocs dataset [11] but lack the necessary ground truth data, or they have available ground truth data, but are small or tailored to very specific types of documents in well-defined scenarios and use cases. These shortfalls

drive the current evaluations in two directions. First, the lack of ability to evaluate correctness shifts the focus on other quality characteristics such as tool reliability and efficiency. While those are important aspects, functional correctness, arguably the key quality characteristic, remains under-evaluated. Second, in the evaluation of functional correctness, we see a disproportion in the coverage of file formats and genres. While a significant number of evaluations address scientific articles in PDF file format, other file formats and genres have received much less effort. These raise concerns in the broader digital preservation field, where heterogeneous collections in complex formats still pose challenges.

New datasets need to be created, but the prevailing approach of manual annotation is too expensive. Approaches based on synthesis of artificial datasets are starting to emerge. Model based approaches such as [4] offer an appealing set of characteristics, but have not been widely applied. Research is needed to develop those approaches further and evaluate their capability of creating datasets suitable for benchmarking needs.

3 TEXT EXTRACTION BENCHMARK

In this section we define the text extraction benchmark according to the five benchmark components [7].

3.1 Motivating comparison

This benchmark addresses text extraction from page based general documents. Following the definition provided by [13], we define *general documents* as documents belonging to any genre such as presentations, letters, or books. We specialize this definition further to page-based general documents where we include documents such as books and letters, but exclude genres such as presentations or sheets. This scope is based on expert input through digital preservation workshops and community discussions [6].

The text extraction benchmark can be defined on various levels of granularity with different units of interest. The smallest unit of interest is set as the word, and the comparison done by this benchmark is based on text snippets. We define a text snippet as a sequence of one or more words. Each document thus contains zero or more text snippets. Each text snippet can be formatted in different ways. Additionally, each document can contain zero or more other elements (e.g. images).

The main motivation of this benchmark is to enable the evaluation of text extraction tools when extracting different text snippets from page based general documents. Depending on the use of extracted text, extraction quality is judged on different levels. The following three are covered by this benchmark and provide the main motivating comparison.

- **Correctness of extracted text from a specific text snippet:** On this level, we judge the number of words in a text snippet that are correctly extracted by the text extraction tool. This correctness aspect of text extraction is important regardless of the application domain.
- **Order of extracted (identified) text snippets:** On this level we judge if the order of text snippets as defined in a document is preserved after the text extraction is performed. This aspect can be important for many applications such as document

	Format			Paragraph			Table	
	DOCX	ODT	PDF	normal	text box	control box	normal	embedded
MS Word 2007 on Windows 7	✓	x	✓	✓	✓	✓	✓	✓
MS Word 2010 on Windows 7	✓	x	✓	✓	✓	✓	✓	✓
LibreOffice 4 on Ubuntu 14.04	x	✓	✓	✓	x	x	✓	x

Table 1: Generating platforms and generated file formats and text snippet implementation details

translation and text to speech synthesis, but may not matter for others.

- **Layout of extracted text snippet:** On this level we judge how well the structure of a text snippet in a document is preserved after the text extraction. It is clear that the extraction of characters from a document with visual layout can not preserve all formatting, in particular aspects that rely on visual appearance. However, structural aspects such as lines and columns can and must be represented correctly. This information is an important aspect in extracting specific elements such as tables and can have an important role in evaluating how well the layout of the document is preserved after a document has been migrated to a new file format.

These levels cover a broad range of application domains, from those where only word level correctness is important (text analysis) to those where it is required that as much as possible formatting information and structure is captured in the extracted text.

3.2 Function

The function that a benchmarked tool will be expected to perform is to extract text from a number of page based general documents. Tools are able to participate in the benchmark regardless of the coverage of the three defined levels according to which they will be judged. It is expected that a tool can be executed as a command line interface and that it produces a single text file which contains text in the document. This file is further processed by the script used to automated the benchmarking workflow. Other output formats such as XML are not included at the moment but can be integrated into the benchmark, provided that the influence of their processing on the final results can be ruled out.

3.3 Dataset and ground truth

The provided dataset contains a number of documents encoded in three file formats chosen for their current use.

- (1) **DOCX Transitional (Office Open XML)**⁶, standardized as ISO 29500, is the format produced by the widely used document editing software MS Word.
- (2) **OpenDocument Text Document Format (ODT)**⁷, standardized as ISO 26300, is the commonly used open source file format for electronic documents.
- (3) **PDF**⁸ is the family of formats and standards for sharing textual documents in fixed layouts.

Documents in these file formats are produced on three platforms: MS Word 2007 on Windows 7, MS Word 2010 on Windows 7 and LibreOffice 4 on Ubuntu 14.04 (Table 1). This is expected to cause

some variation in the formats. For example, in this data set, PDF is produced in versions 1.5 (by Word) and 1.4 (by LibreOffice).

The dataset covers two main types of text snippets: paragraphs and tables. It is important to highlight that the dataset covers the variety of features by which different platforms and formats implement such generic document elements such as tables and paragraphs. The aim is to represent real-world diversity and facilitate the evaluation of correctness across tools when encountering particular format features.

The dataset was generated by the method described in Section 4. Each document in the dataset is accompanied by ground truth data. This data contains metadata about the document (e.g. number of pages, the number of tables in a document, the number of content control boxes, ...) and text data for each text snippet. This text data contains the raw text of each text snippet and layout information. This structural layout information is currently limited to the number of lines a text snippet is displayed on and the text of each line.

3.4 Performance measures

Several performance measures are defined to judge the correctness of text extraction tools and their fitness for different purposes. These measures address the three levels defined in the motivating comparison section. Calculating performance measures should satisfy the following requirements.

- Measures addressing one level should not be affected by measures that address other levels. So if a tool does not perform well on the order level, it still needs to be able to perform well on the layout or word correctness level.
- Extractors that add additional words to denote special text elements (e.g. HYPERLINK to mark extracted hyperlink from a document) should not be penalized.

Measures are defined for the three levels defined in the motivating comparison and listed in Table 2. Each level can apply to a specific text snippet in a document processed by a text extraction tool, to a specific document processed by a tool, or to a tool applied to many documents. An exception is the order of a single text snippet which is not judged. The order level is first evaluated on a document and afterwards aggregated per tool.

4 DATASET GENERATION

4.1 Model driven engineering data generation framework

The goal of data generation is to automatically synthesize a set of documents accompanied by ground truth data. We ground our approach in model driven engineering. A detailed description of the approach can be found in [4]. Here we discuss briefly the

⁶<http://www.digitalpreservation.gov/formats/fdd/fdd000397.shtml>

⁷<http://www.digitalpreservation.gov/formats/fdd/fdd000427.shtml>

⁸<http://www.digitalpreservation.gov/formats/fdd/fdd000030.shtml>

	Level	Measure Name	Definition
Text snippet	Correctness	C, D, I, S, N	For each text snippet we report four values : C - number of correct words, D - number of deleted words, I - number of inserted words, and S - number of substituted words. N is the number of words in a text snippet.
	Layout	layoutPreserved	True if the layout is preserved, false otherwise.
Document	Correctness	percCorrect	Ratio of correct words from all text snippets and the number of words in the whole document.
	Order	orderPreserved	True/False
	Layout	percLayout	Percentage of text snippets with a preserved layout.
Tool	Correctness	avgCorrect	Average value of correctness across the whole dataset
	Order	avgOrder	Percentage of documents with correct order of text snippets
	Layout	avgLayout	Average percentage of text snippets with correct layout

Table 2: Performance measures calculated per each snippet, document and tool

major components and demonstrate how different text extraction benchmark requirements are met by those components.

This is a novel approach compared to manual annotation. Our aim is to fully automate the dataset and ground truth generation while retaining full control with minimal human effort. From the text extraction benchmark defined in Section 3 we can identify several major requirements.

Table 3 maps key aspects of data generation to the requirements they address and illustrates how specific features are represented across models. The modelling itself is completed in two phases.

First, structural and implementation details of a document are represented by creating *platform independent* and *platform specific* metamodels. These metamodels define a document by modelling several key elements and their relationships.

Second, to introduce variety into the dataset, two sets of model transformations are defined. These support the diversification of (a) platform independent and (b) platform specific models in terms of their structural and implementation aspects. Each individual transformation adapts a particular element or set of elements in a document to change how its features are represented. The set of target states is provided through aggregate feature distributions that determine the probability by which any particular element is to be encoded in a particular technical form. These feature distributions are derived from aggregate statistics obtained from content profiles.

Using these metamodels, a set of documents is then generated where each document follows the modelled structure described by the metamodels, and the entire dataset follows the feature distributions as modelled in the second phase.

The required ground truth is extracted in two ways. First, platform independent and platform specific models are a rich source for ground truth data. In order to extract information about a modelled document, model queries are defined. However, here it is important to pay attention if the information extracted via the query is affected when the document is actually generated. For example, the number of pages in a document is often not known until the document is generated. In such cases we must extract that kind of information during the generation process.

The manual annotation approach hits several obstacles in satisfying the listed requirements. First, since the data already exists, it does not have any control of the structural and implementation aspects of the documents. Second, because of the need for detailed ground truth, the approach will require significant amounts of human effort per document with a high risk of mistakes.

4.2 Instantiating the data generation framework for this benchmark

Our implementation of these components is based on the Eclipse Modelling Environment, which covers Eclipse Modelling Framework (EMF), Query View Transform (QvT) and Object Constraint language (OCL).

To capture different text snippets and their relationships in terms of order and hierarchy, the platform independent metamodel (PIMM) models two types of text snippets (paragraphs and tables) that can appear in a page based document, and enables adding images to a document.

As our benchmark targets two main platforms (MS Word and LibreOffice), we provide a platform specific metamodel (PSMM) for each. A PSMM provides implementation details of each element from the PIMM. The most important aspects are sets of implementation options of text snippets. For instance, on a MS Word platform, a paragraph from the PIMM can be implemented as a regular paragraph, a text box, or a content control box. Similarly, a table can be implemented as a normal MS Word table or an embedded MS Excel table.

To diversify aspects such as the number of text snippets and their order and implementation possibilities, a set of model transformations is defined that operate on the PIM and PSM level. To achieve a realistic dataset in terms of the feature distributions, we use the content profiling tool C3PO [14] to sample initial real world distributions from the *Govdocs* data set. Where feature distribution data is not available, the transformation falls back to its default distribution. For example, the percentage of text boxes in a document is unavailable through the features covered in tools integrated in

Text extraction benchmark dataset and ground truth requirements	Model Driven Engineering Concept	Features covered	Technology
Capture structural and content aspects of documents in terms of the number, arrangement and content of document elements	Platform Independent Metamodel (PIMM), Platform Independent Model (PIM)	Paragraph, Table	EMF , Ecore
Capture implementation details of the whole document (file format, tool used) and single elements (formatting, implementation)	Platform Specific Metamodel (PSMM), Platform Specific Model (PSM)	DOCX, ODT, PDF, Normal Paragraph, Text Box	EMF, Ecore
Support a variety of structural, content and implementation aspects of documents	Model Transformations	Number of tables in a document, order of text boxes, number of words in a paragraph	Query View Transform (QvT)
Supply detailed ground truth covering content, order and layout of text snippets	Model Queries, Code Generation	Content of a text snippet, content of each line of a text snippet	Object Constraint Language(ACL), VisualBasic, StarBasic

Table 3: Mapping of requirements to model dataset generation framework

C3PO. We thus define that 5% of all Paragraphs in a document will be implemented as a Text Box.

PSMs are translated into macro code which runs on the specified platforms and generates documents. Depending on the target platform, we generate either Visual Basic code that controls MS Word or StarBasic macro that controls LibreOffice. This means that the documents are indeed generated natively using the entire stack of specific code implemented within these environments.

The framework generates two sets of ground truth elements per document. First, metadata about the generated document, including the software and operating system used to create the document, the file format the document is stored in, the number of tables in the document and the number of paragraphs that are implemented as text boxes. The purpose is to enable rich data analysis of the benchmark results. Second, data about the text content of each generated text snippet: (1) raw text of the text snippet with the preserved order of words and (2) structural layout of the text snippet. The latter contains text separated into lines where each line reflects the visual layout of the text snippet when rendered.

Ground truth is extracted in two ways. First, ACL queries are used to extract specific metadata from PIMs and PSMs. Since ACL is a declarative language, it offers an efficient way to articulate the metadata to be generated. Second, by generating macro code that in turn generates the ground truth during the data set generation process. This is used, and necessary, for those ground truth elements for which we can not know the exact value before the content is rendered. This covers metadata such as the number of pages in a document and rendering information of certain text snippets such as the number of lines of text for a paragraph. Both MS Word and LibreOffice expose a rich API which enables obtaining this rendering information.

5 SOFTWARE EVALUATION PROCESS

5.1 General workflow

The evaluation workflow is divided into two stages as shown in Figure 1. The input to the evaluation step is the generated dataset, where each file is accompanied by a ground truth file.

In the first stage of the evaluation, each tool is executed on the dataset and produces a text output for each file it knows how to process. While it is required that a tool produces raw text output, some tools additionally include various formatting elements. This can range from a case where all formatting is removed to cases where a tool tries to mark certain document elements such as paragraphs and tables and preserve their layout as much as possible. We can identify two extreme cases : 1) all the text is returned in one line and 2) each word is returned on a separate line. The second case is common for the uses cases where only words are important and all other formatting can be neglected.

To avoid bias and support flexible interpretation, the evaluation method separates the content of text (words) from structure.

Ground truth data (covering information about each text snippet and stored in a XML format) is turned into a list of elements where each element represents one text snippet and contains the raw text of the text snippet and the text of each line when the text snippet is displayed. The matching algorithm matches each text snippet to a part of extracted text. As the procedure is not uniquely defined, we provide details of our approach in Section 5.2. Once matching is completed, the second stage of the evaluation workflow calculates performance measures, as discussed in Section 5.3.

5.2 Finding a text snippet in extracted text

To match a text snippet to a part of the extracted text, the algorithm must identify its starting and ending words. Among several approaches on how to identify those positions, we base our approach on the edit distance. For a given snippet, the algorithm determines the portion in the extracted text with the smallest edit distance. The function is based on the Levenshtein algorithm [16] but operates on a word level instead of the character level of two strings. It enables us to compute the edit distance of two strings in terms of the number of correct words C , number of inserted words I , number of deleted words D , and number of substituted words S . To identify the portion in the extracted text where a text snippet is found, we extend the algorithm as proposed by Sellers [26] to identify the text segment with the smallest edit distance. Once start and end positions are computed, it is straight forward to calculate the starting and ending

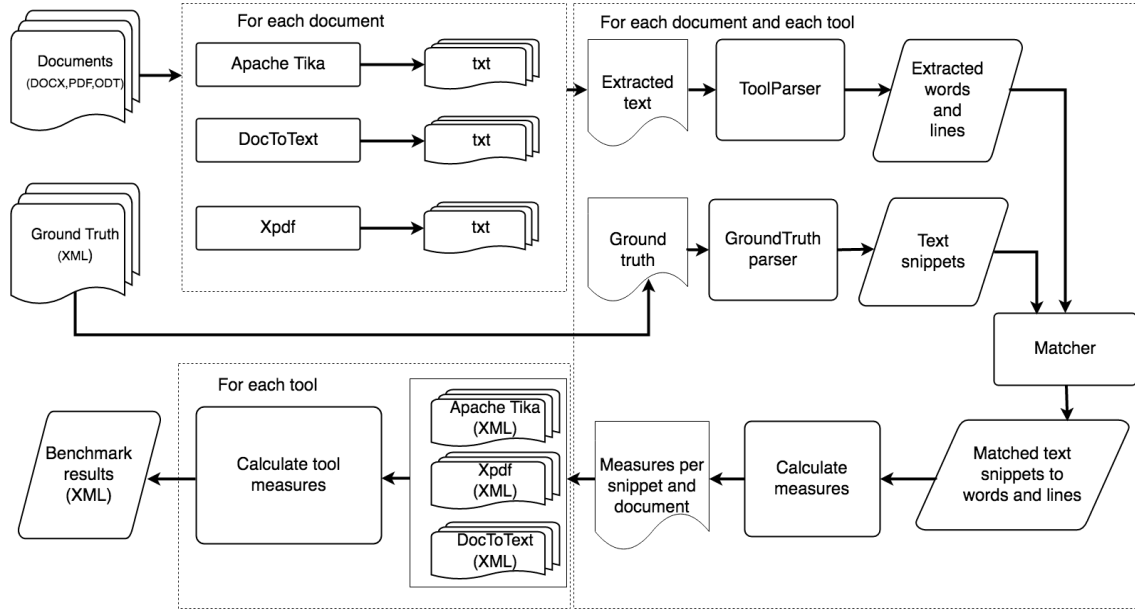


Figure 1: Evaluation process

line as that information is stored during parsing. The approach yields several benefits:

- (1) To identify a text snippet, we are operating on a word level and can ignore any available formatting. Using approaches where additional formatting is used is a biased approach towards tools that are better in preserving the text layout.
- (2) We are independent of the tool output, so it is straight forward to add additional tools to the benchmark.
- (3) It enables matching of text snippets even in cases when text is not 100% accurate, when text was only partially extracted or additional text was inserted.

However, the approach always finds a match, even when a text snippet is not present in the extracted text. Therefore we implemented a threshold value that discards a text snippets with less than 50% of correct words in the matched interval according to the edit distance. Further empirical research is required to find out how threshold settings affect matching results and benchmark scores.

5.3 Calculating performance measures

Once the matching is performed for each text snippet we calculate performance measures as defined by the benchmark specification (Section 3). The C, D, I, S, N values are calculated by rerunning the edit distance algorithm on the text snippet and matched portion of the text. The *layoutPreserved* measures are computed by verifying that the content of each line of a text snippet as defined in the ground truth matches the identified lines in the extracted text. The *orderPreserved* measure on the document level is calculated simply by monitoring the sequence of starting and ending words and verifying that it is strictly increasing.

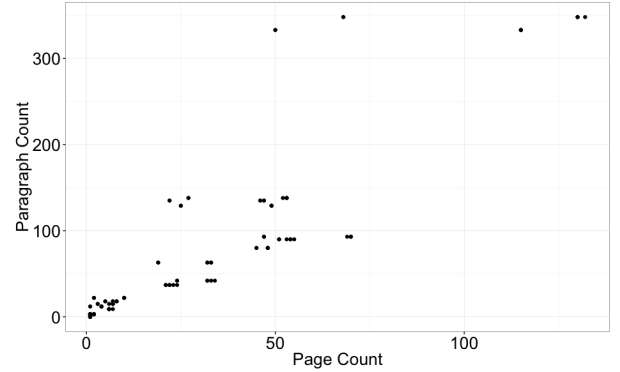


Figure 2: Spread of documents in the generated dataset according to page and paragraph count

6 RESULTS

6.1 Generated dataset and ground truth

Based on the feature distributions acquired by using C3PO on the Govdocs datasets, 20 PIMs were automatically generated. Those PIMs were diversified to 120 PSMs covering the 3 platforms defined in Table 1. From those PSMs, the final 120 documents were generated covering three file formats : DOCX, ODT and PDF. In total there are 40 DOCX, 20 ODT and 60 PDF documents.

These documents cover various combinations of document elements and their properties. Figure 2 shows a scatter plot of page and paragraph count in the whole dataset. We detect a radial spread where there is a bigger coverage of combinations with smaller number of pages and paragraph and a rather sparse coverage of larger documents with more pages and paragraphs.


```

<element>
  <ID>da80bdfc-9c50-4606-82c0-ffc22131876</ID>
  <text> nonjuristic charlatan 86 91 68 61 93 93 </text>
  <lines>
    <line num="1">nonjuristic charlatan</line>
    <line num="2">86 91</line>
    <line num="3">68 61</line>
    <line num="4">93 93</line>
  </lines>
</element>
<element>
  <ID>79eac1a1-32f9-484d-bbab-c3c5dfef3809</ID>
  <text> societism consentiently cyanean grimp ungagged skato ...
  <lines>
    <line num="1">societism consentiently cyanean grimp ungagged...
    <line num="2">cinquefoil defrock caviya dote polymasty Hallo...
    <line num="3">measurelessly noctambulation scribledom Glis ...
    <line num="4">rhyssimeter Seltzer remonstrance underbraced ...
    <line num="5">consummativeness theopantism decomposite sup...
    <line num="6">zadruga terebratuline Wilsonian relime corta ...
  </lines>
</element>

```

Listing 1: Part of the ground truth data showing raw text and layout information for two text elements

Each generated document is accompanied by two files containing ground truth data. The first contains all the metadata and the second data about text snippets. Listing 1 shows a portion of text ground truth for one document. It shows two text snippets (elements) where the first has 8 words which are divided to 4 lines. The second snippet has more text, displayed on 6 lines.

6.2 Benchmark results

We provide here, due to the space limit, only results on the tool level. As some tools like Xpdf are extracting data only from PDF files, we do not analyse it in terms of the other two file formats (DOCX, ODT). Other tools such as Apache Tika and DocToText are capable of extracting text from all three file formats.

Figure 3 provides an overview of the results for Apache Tika and DocToText on the whole dataset (a) and an overview of the results of Tika, DocToText and Xpdf only on the pdf portion of the dataset (b). Three performance measures are reported for each tool: *avgCorrect*, *avgOrder*, and *avgLayout*. For Apache Tika we included several versions (v1.1, v1.2 and v1.13) to compare the tool quality across versions.

Both DocToText and Apache Tika achieved a high degree of correctness when extracting text from all three file formats. This ranges well above 90%, and Apache Tika v1.13 achieved the best results (99.3%). Comparing versions of Apache Tika shows that the newest version (v1.13) achieved the best result. The slightly worse results (97.6%) in older versions are mainly due to the specific implementations of paragraphs that our benchmark supports (TextBox and Content Control Box in MS Word). Apache Tika v1.1 and v1.2 both fail at extracting Text Boxes from documents in DOCX file format created by MS Word 2010, but are successful on those created by MS Word 2007. Content Control Boxes are not extracted by those two versions for both cases (MS Word 2007 and MS Word 2010). This behaviour in fact was identified by the two reported bugs⁹ resolved after version 1.2. The small difference in results is due to the low probability of appearance of the TextBoxes and Content Control Boxes in our documents.

⁹<https://issues.apache.org/jira/browse/TIKA-1005> and <https://issues.apache.org/jira/browse/TIKA-1130>

The performance on the order of text snippets (*avgOrder*) shows that DocToText is significantly better than Apache Tika, which demonstrates equal performance across all versions. While DocToText preserved the order of text snippets in 100% of the documents, Apache Tika did so only in 70%. This has significant implications for its use in data processing workflows that rely on the integrity of overall text content.

All tools exhibit weaker performance in terms of the layout aspects (*avgLayout*). Again DocToText achieves the highest performance with an average of 54.7% of text snippets with correct layout, but Tika comes close. The reason for the weaker results is the fact that neither tool is good at identifying structure when extracting text from DOCX or ODT. An interesting exception here is the embedded excel table.

Figure 3 (b) provides an overview of the success of Apache Tika v1.13, DocToText and Xpdf when operating only on PDF documents. According to correctness, Tika is showing the best score (99.1%) but is comparable to DocToText (98.9%). Xpdf demonstrates slightly worse result (95%). In terms of the order, DocToText shows again the best performance and Xpdf is second. Apache Tika is showing a significantly worse performance than the other two tools (58%). It can be concluded that the PDF proportion of the dataset hurts the performance of Apache Tika on the whole dataset in terms of the order (70%). In terms of layout, the score is much higher when focusing on PDF. DocToText again has the highest score, followed closely by Tika and Xpdf.

7 DISCUSSION

The results clearly differentiate the quality of benchmarked tools and show that text extraction components still need to be improved. This is especially true for the order and layout aspects which can be a challenge even in the most simple cases.

The data set uncovers specific issues in tools and distinguishes effectively according to the identified aspects. This demonstrates that the method is reasonable and effective for dataset generation in scenarios where the presence of fine-granular and reliable ground truth is crucial. We discuss key aspects below.

- **Scalability of the data generation method:** We have demonstrated our approach on 3 platforms. Adding additional platforms to include options such as LibreOffice on Windows and different versions of software provides new data sets with the entire stack of ground truth, data generation and granularity merely through extending the configuration set. The spread in terms of the platforms can include operating systems, software used to create documents and other settings such as installed fonts and libraries. This can be easily achieved by adding new virtual machine configurations. Similarly, the framework can be adapted to extract other ground truth elements or introduce other features and feature distributions simply by adapting the models.
- **Effectiveness in finding faulty behaviour:** The experiments demonstrate that the approach is effective in finding faulty behaviour. This is confirmed by confirming several known bugs, but also finding new cases.
- **Analysis support:** The provided ground truth enables a deep analysis of benchmark results. The method generates not just

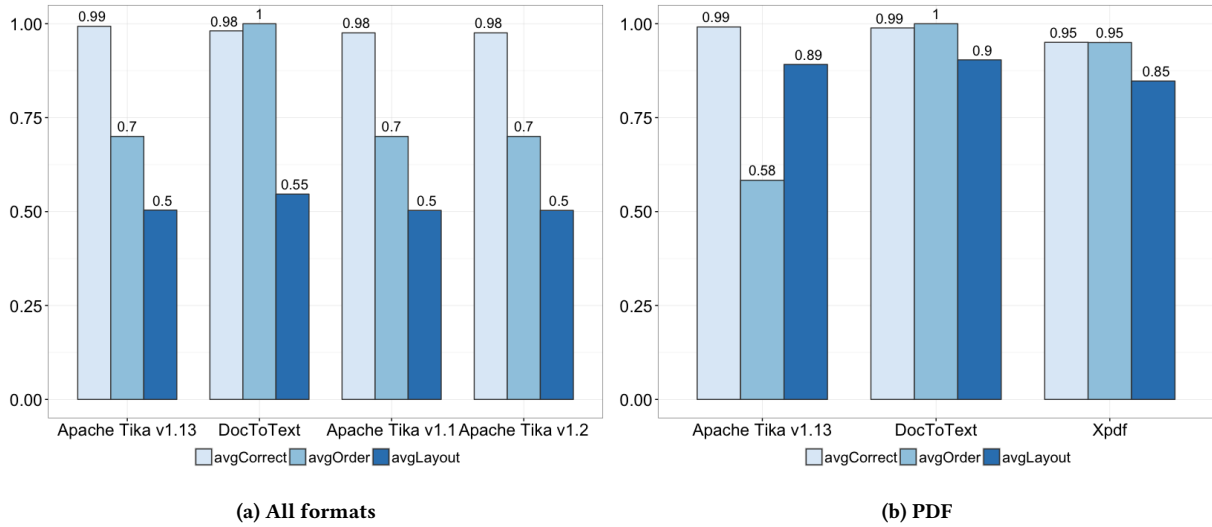


Figure 3: Tools performance measures

text ground truth, but also documents a list of features (meta-data) that can be further used to better understand faulty behaviour. Such additional data can become important when an error needs to be analyzed to identify and reproduce faults and can help to reveal patterns in faulty behaviour.

- **Benchmark results:** A key quality for benchmark results is the external validity of results. This requires that the used dataset is representative of a real world dataset. Representativeness should be achieved by following the feature distributions of real world datasets. We are doing this by using the collection profiling and sampling provided by C3PO. The second and equally important aspect of representativeness is the fact that the dataset is born right within the same environment (MS Word, LibreOffice) as real documents. That means that generated documents, even though simple in their structure at this point, still resemble a lot of the implementation details of real-world documents. We are less interested in generating interesting exotic examples (e.g. a PDF document which is also a valid ZIP file) because those example are unlikely to appear in real life. If they do on the other hand, the approach can be used to model and generate them.
- **Expertise** Human involvement is not completely removed through this approach, but shifted: Manual verification on data sets of realistic size would not be effective, but instead of the repetitive task of annotation, what the approach requires is expertise in format specifications and modelling.
- **Data sharing.** The data sets can be freely released and shared, a crucial aspect considering the need to advance solid evidence bases in digital preservation and other fields [21]. The data sets and results from the experiment described above are available at [8].

8 CONCLUSION

This paper developed and implemented a benchmark for text extraction tools comprising the five benchmark components defined

in [7]. Functional correctness is judged in terms of the correctness of the extracted text, order and layout. In contrast to the prevailing approach of manually annotating randomly selected documents, we base our benchmark on an automatically synthesized dataset. We implement each component of the benchmark and describe the approach to synthesizing both data set and ground truth from an initial model based on real-world feature distributions.

We foresee that future work lies in three directions. First, the feature space of the generated dataset must be extended in several directions. On the platform independent level, we intend to model more complex element arrangements such as nested objects (e.g. table in a table). Furthermore we intend to explore ways of incorporating specific genre models that enable the data generation method to focus on specifically defined scenarios such as caption extraction from scientific articles or text extraction from invoices. On the platform specific level, the supported feature space should be expanded. It is expected that once the platform specific models get rich enough, the question of their reusability will need to be addressed.

Second, we are currently basing our data generation target profiles on feature distributions from real world datasets. Where those feature distributions are unknown, we provide our own to the best of our knowledge. Basing the dataset on representativeness is an important aspect for the needs of benchmarking, but from a software testing perspective, this comes with flaws. Instead, in software testing, *feature space coverage* is much more important than representativeness: Rather than approximating a specific data set, the set of possible data sets is sought. Therefore we are interested in extending our approach in ways that address and quantify feature space coverage more generally.

Finally, we foresee the need for extensive experimentation that should explore several directions. The first direction is the effect of particular approaches on benchmark results. A series of experiments should be done in order to evaluate the ability of such data generation methods to generate datasets that produce reliable

benchmark results and compare the synthesized datasets to manually annotated datasets to evaluate whether the results achieved on manually annotated datasets can be confirmed with synthesized data.

In those scenarios where evaluations are already available, we can use them as a baseline for evaluating the synthesis approach. Once both methods of generating a dataset (synthesizing or annotating) are available, it will be important to understand in which scenarios which approach is more effective, and how to combine the approaches to leverage each of their strengths. While annotation is expensive, synthesizing a dataset comes with a price as well in terms of the effort and knowledge required to do the modelling task. Hence, it will be important to explore and experiment with the possibilities of combining the two approaches in order to gain the benefits from both.

ACKNOWLEDGMENTS

Part of this work was supported by the Vienna Science and Technology Fund (WWTF) through the project BenchmarkDP (ICT12-046) and by NSERC through RGPIN-2016-06640.

REFERENCES

- [1] Timothy B. Allison and Paul M. Herceg. 2015. *Methods for Evaluating Text Extraction Toolkits: An Exploratory Investigation*. Technical Report. DTIC Document. <http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA626579>
- [2] Saswat Anand, Edmund K. Burke, Tsong Yueh Chen, John Clark, Myra B. Cohen, Wolfgang Grieskamp, Mark Harman, Mary Jean Harrold, Phil McMinn, Antonia Bertolino, J. Jenny Li, and Hong Zhu. 2013. An orchestrated survey of methodologies for automated software test case generation. *Journal of Systems and Software* 86, 8 (Aug. 2013), 1978–2001. DOI: <http://dx.doi.org/10.1016/j.jss.2013.02.061>
- [3] E.T. Barr, M. Harman, P. McMinn, M. Shahbaz, and S. Yoo. 2015. The Oracle Problem in Software Testing: A Survey. *IEEE Transactions on Software Engineering* 41, 5 (May 2015), 507–525. DOI: <http://dx.doi.org/10.1109/TSE.2014.2372785>
- [4] Christoph Becker and Kresimir Duretec. 2013. Free Benchmark Corpora for Preservation Experiments: Using Model-driven Engineering to Generate Data Sets. In *Proceedings of the 13th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL '13)*. ACM, New York, NY, USA, 349–358. DOI: <http://dx.doi.org/10.1145/2467696.2467719>
- [5] Christoph Becker, Kresimir Duretec, and Andreas Rauber. 2017. The Challenge of Test Data Quality in Data Processing. *J. Data and Information Quality* 8, 2 (Jan. 2017), 7:1–7:4. DOI: <http://dx.doi.org/10.1145/3012004>
- [6] Christoph Becker, Kresimir Duretec, Artur Kulmukhametov, and Andreas Rauber. 2016. The Benchmarking Forum at IPRES 2015. *D-Lib Magazine* 22, 1/2 (2016), 2015. DOI: <http://dx.doi.org/10.1045/january2016-becker>
- [7] Kresimir Duretec, Artur Kulmukhametov, Andreas Rauber, and Christoph Becker. 2015. Benchmarks for Digital Preservation tools. In *Proceedings of the 12th International Conference on Digital Preservation (IPRES)*. <https://hdl.handle.net/11353/10.429524>
- [8] Kresimir Duretec, Andreas Rauber, and Christoph Becker. 2017. BenchmarkDP - Text extraction from general documents benchmark dataset. (2017). DOI: <http://dx.doi.org/10.6084/m9.figshare.c.3683332>
- [9] Nicola Ferro. 2016. Proposal for an Evaluation Framework for Compliance Checkers for Long-term Digital Preservation. In *12th Italian Research Conf. on Digital Libraries (IRCDL)*. DOI: http://dx.doi.org/10.1007/978-3-319-56300-8_12
- [10] Andrew Fetherston and Tim Gollins. 2012. Towards the Development of a Test Corpus of Digital Objects for the Evaluation of File Format Identification Tools and Signatures. *International Journal of Digital Curation* 7, 1 (March 2012), 16–26. DOI: <http://dx.doi.org/10.2218/ijdc.v7i1.211>
- [11] Simson Garfinkel, Paul Farrell, Vassil Roussev, and George Dinolt. 2009. Bringing science to digital forensics with standardized forensic corpora. *Digital Investigation* 6 (Sept. 2009), S2–S11. DOI: <http://dx.doi.org/10.1016/j.diin.2009.06.016>
- [12] Yunhua Hu, Hang Li, Yunbo Cao, Dmitriy Meyerzon, and Qinghua Zheng. 2005. Automatic Extraction of Titles from General Documents Using Machine Learning. In *Proceedings of the 5th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL '05)*. ACM, New York, NY, USA, 145–154. DOI: <http://dx.doi.org/10.1145/1065385.1065418>
- [13] Yunhua Hu, Hang Li, Yunbo Cao, Li Teng, Dmitriy Meyerzon, and Qinghua Zheng. 2006. Automatic extraction of titles from general documents using machine learning. *Information Processing & Management* 42, 5 (Sept. 2006), 1276–1293. DOI: <http://dx.doi.org/10.1016/j.ipm.2005.12.001>
- [14] Artur Kulmukhametov and Christoph Becker. 2014. Content Profiling for Preservation: Improving Scale, Depth and Quality. In *The Emergence of Digital Libraries: Research and Practices*. Number 8839. Springer International Publishing, 1–11. DOI: http://dx.doi.org/10.1007/978-3-319-12823-8_1
- [15] Matthew Lease and Emine Yilmaz. 2013. Crowdsourcing for information retrieval: introduction to the special issue. *Information Retrieval* 16, 2 (April 2013), 91–100. DOI: <http://dx.doi.org/10.1007/s10791-013-9222-7>
- [16] Vladimir I Levenshtein. 1966. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady* 10 (February 1966), 707.
- [17] Ying Liu, Kun Bai, Prasenjit Mitra, and C. Lee Giles. 2007. TableSeer: Automatic Table Metadata Extraction and Searching in Digital Libraries. In *Proceedings of the 7th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL '07)*. ACM, New York, NY, USA, 91–100. DOI: <http://dx.doi.org/10.1145/1255175.1255193>
- [18] Natasa Milic-Frayling. 2010. Digital Object Characterization: Document Conversion and Quality Assurance. In *Dagstuhl Seminar Proceedings*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik. <http://drops.dagstuhl.de/opus/volltexte/2010/2901/>
- [19] D. Di Nardo, N. Alshahwan, L. C. Briand, E. Fournert, T. Nakic-Alfirevic, and V. Masquelier. 2013. Model based test validation and oracles for data acquisition systems. In *2013 IEEE/ACM 28th International Conference on Automated Software Engineering (ASE)*. 540–550. DOI: <http://dx.doi.org/10.1109/ASE.2013.6693111>
- [20] D. Di Nardo, F. Pastore, and L. Briand. 2015. Generating Complex and Faulty Test Data through Model-Based Mutation Analysis. In *2015 IEEE 8th International Conference on Software Testing, Verification and Validation (ICST)*. 1–10. DOI: <http://dx.doi.org/10.1109/ICST.2015.7102589>
- [21] National Digital Stewardship Alliance. 2014. *2015 National Agenda for Digital Stewardship*. Technical Report. <http://www.digitalpreservation.gov/nds/documents/2015NationalAgenda.pdf>
- [22] Robert Neumayer, Hannes Kulovits, Andreas Rauber, Manfred Thaller, Eleonora Nicchiarelli, Michael Day, Hans Hofman, and Seamus Ross. 2007. On the Need for Benchmark Corpora in Digital Preservation. In *2nd DELOS Conference on Digital Libraries*. Pisa, Italy.
- [23] Sung Hee Park. 2013. *Discipline-Independent Text Information Extraction from Heterogeneous Styled References Using Knowledge from the Web*. Ph.D. Dissertation. Virginia Polytechnic Institute and State University. <https://vtechworks.lib.vt.edu/handle/10919/52860>
- [24] Jeff Pasternack and Dan Roth. 2009. Extracting Article Text from the Web with Maximum Subsequence Segmentation. In *Proceedings of the 18th International Conference on World Wide Web (WWW '09)*. ACM, New York, NY, USA, 971–980. DOI: <http://dx.doi.org/10.1145/1526709.1526840>
- [25] Cartic Ramakrishnan, Abhishek Patnia, Eduard Hovy, and Gully APC Burns. 2012. Layout-aware text extraction from full-text PDF of scientific articles. *Source Code for Biology and Medicine* 7, 1 (Dec. 2012), 7. DOI: <http://dx.doi.org/10.1186/1751-0473-7-7>
- [26] Peter H. Sellers. 1980. The theory and computation of evolutionary distances: pattern recognition. *Journal of algorithms* 1, 4 (1980), 359–373. <http://www.sciencedirect.com/science/article/pii/0196677480900164>
- [27] Susan Elliott Sim, Steve Easterbrook, and Richard C. Holt. 2003. Using Benchmarking to Advance Research: A Challenge to Software Engineering. In *Proceedings of the 25th International Conference on Software Engineering (ICSE '03)*. IEEE Computer Society, Washington, DC, USA, 74–83. <http://dl.acm.org/citation.cfm?id=776816.776826>
- [28] Ross Spencer. 2013. Generation of a Skeleton Corpus of Digital Objects for the Validation and Evaluation of Format Identification Tools and Signatures. *International Journal of Digital Curation* 8, 1 (June 2013), 120–130. DOI: <http://dx.doi.org/10.2218/ijdc.v8i1.249>
- [29] Matt Staats, Michael W. Whalen, and Mats P.E. Heimdahl. 2011. Programs, Tests, and Oracles: The Foundations of Testing Revisited. In *Proceedings of the 33rd International Conference on Software Engineering (ICSE '11)*. ACM, New York, NY, USA, 391–400. DOI: <http://dx.doi.org/10.1145/1985793.1985847>
- [30] T. Strecker, J. v. Beusekom, S. Albayrak, and T. M. Breuel. 2009. Automated Ground Truth Data Generation for Newspaper Document Images. In *2009 10th International Conference on Document Analysis and Recognition*. 1275–1279. DOI: <http://dx.doi.org/10.1109/ICDAR.2009.214>
- [31] Jorg Tiedemann. 2014. Improved Text Extraction from PDF Documents for Large-Scale Natural Language Processing. In *Computational Linguistics and Intelligent Text Processing*. Springer, Berlin, Heidelberg, 102–112. DOI: http://dx.doi.org/10.1007/978-3-642-54906-9_9
- [32] Mark Utting, Alexander Pretschner, and Bruno Legeard. 2012. A taxonomy of model-based testing approaches. *Software Testing, Verification and Reliability* 22, 5 (Aug. 2012), 297–312. DOI: <http://dx.doi.org/10.1002/stvr.456>
- [33] J. M. Wing and J. Ockerbloom. 2000. Respectful type converters. *IEEE Transactions on Software Engineering* 26, 7 (July 2000), 579–593. DOI: <http://dx.doi.org/10.1109/32.859529>