Compiling a Corpus of Technical Documents for Dialogue System Development in the Industrial Sector

Laura García-Sardiña,[1], **Eneko Ruiz,**[2] **Cristina Aceta,**[3]
Izaskun Fernández,[3] **María Inés Torres,**[2] **Arantza del Pozo,**[1]
[1]Vicomtech Foundation, Basque Research and Technology Alliance (BRTA)
[2]University of the Basque Country UPV/EHU, Speech Interactive Research Group
[3]Tekniker Foundation, Basque Research and Technology Alliance (BRTA)

Abstract

This work deals with the compilation of a machine-readable corpus of technical documents ready to be processed for the development of data-driven dialogue system applications in the industrial sector. To this end, we propose a pipeline to convert technical *PDF* documents into a set of *JSON* files that allow machine processing of their information. This procedure is able to extract and organise a variety of content types such as text, images, and tables in order to obtain a corpus of structured information, which additionally allows easy conversion into other formats for further visualization or processing purposes. A qualitative analysis of the proposed procedure by expert technical operators has resulted in a positive validation of the proposal. The compiled sample includes Question Answering annotations and instances of the dialogue ontology related to industrial procedures that shall allow the development of voice user interfaces to assist technical operators dealing with industrial tasks.

1 Introduction

The emergence of virtual assistants, mainly in the leisure and domestic scopes, proves that interacting through natural language with different devices and applications is a reality that keeps improving. In this context, the demand for such capacities in industry is increasing, since natural communication with industrial systems leads to productivity and security improvements which reduce operation time and costs (González-Docasal et al., 2021), providing operators 4.0 with powerful and intuitive interaction mechanisms to perform their tasks successfully (Romero et al., 2020).

In fact, the concept of human-centric manufacturing where a collaborative intelligence assists operators in their needs (Lu et al., 2022) has recently been introduced, replacing more traditional system-centric factories. In order to reduce the cost of developing such systems, machine learning based methods are increasingly being considered (Torres, 2013; Zamora et al., 2017), as they are easier to develop and maintain. However, these systems require considerable amounts of data for development (Wang et al., 2018), and one of the main caveats in industrial settings is the lack of corpora for model training (Serras et al., 2020; Vázquez et al., 2023; Justo et al., 2010). A common practice to obtain data for under-resourced scenarios is to exploit available documentary records from the domain at hand (Tiedemann, 2014). In industrial scenarios, common relevant sources are technical documents such as user manuals and manufacturing and assembly dossiers, which are usually available in PDF format.

Automatic information extraction and structuring from PDF documents is a difficult unsolved task (Bast and Korzen, 2017), with scarce previous work, especially in the industrial domain. Documented approaches (Dong et al., 2021) exploit PDF technical reports with relatively simple structures and only consider text extraction, leaving aside the preservation of images and tables, which are especially frequent and relevant in industrial technical documents.

This work proposes a pipeline to convert industrial technical documents in PDF format into a machine-readable JSON structure, that can be used to develop data-driven dialogue system applications. In particular, annotations for Question Answering (QA) and procedure-related instances for the development of ontology-based Dialogue Systems (DS) have been compiled. QA systems allow obtaining information from a collection of unstructured documents (Wang et al., 2021; Xingguang et al., 2022), and are evolving towards conversational interfaces (Reddy et al., 2019), whereas ontology-based DS allow to define domains in detail and reduce ambiguity between agents (Antonelli and Bruno, 2017), leading to structured

Manufacturer	Machines	Tasks
ABB	IRB120	Maintenance
Stäubli	Robotic arm	Maintenance
Fanuc	F11 and F12	CNC
Fagor	8055, 8060	CNC
Fagor	8065, 8070	CNC
Siemens	PG 1106	CNC
Heidenhain	TNC 426	CNC
Heidenhain	TNC 430	CNC
Ikor	RACK 81.51	Assembly

Table 1: Source technical manuals.

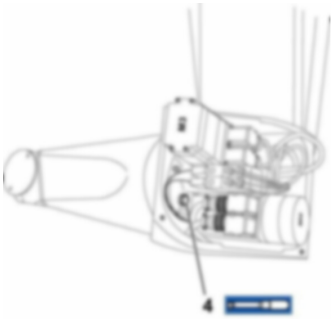

Figure 1: Fragment of a manual showing an image with a pointer to a relevant position for a maintenance task. The image has been intentionally blurred to comply with copyright restrictions.

knowledge representations required in procedural assistance applications. The combination of these technologies shall enable the development of voice-based assistants aimed at guiding operators throughout maintenance tasks, providing answers to technical questions and/or assisting operators in procedures extracted from industrial technical documents.

The rest of the paper is structured as follows. Section 2 introduces the main characteristics of industrial technical documents and a machine-readable corpus structure for them. Section 3 describes the proposed pipeline to convert technical documents into the that format, while the annotation procedure for relevant question-answer pairs is presented in Section 4. Next, Section 5 provides details regarding the corpus compiled using the proposed pipeline, as well as some evaluation metrics. Finally, conclusions are drawn in Section 6.

2 Industrial Technical Documents

Industrial documents of technical nature cover a wide range of topics. In this work, we have selected a set of 9 documents, focusing on three relevant industrial tasks in which operators used to frequently consult technical documents when interacting with industrial production machines and processes, and where the development of QA systems as well as procedural assistants, along with spoken interfaces, can have considerable productivity impact: (a) maintenance, (b) machine programming, and (c) manufacturing and assembly tasks.

Table 1 details the selected documents from different manufacturers, spanning from robot maintenance manuals, or computer numerical control (CNC) programming manuals to manufacturing and assembly dossiers, all in PDF format.

Despite the structures of the manuals fluctuate across manufacturers, all documents include headers and footers, chapter and sub-chapter based levels, as well as a considerable amount of tables and images embedded in the text. In addition, tables and images usually include key information that cannot be overlooked for QA and DS applications for industrial procedure assistance, e.g. they often describe processes or show parts of the machine where actions are required. For instance, Figure 1 shows a PDF fragment including an image indicating the concrete positioning of a particular element, to be considered during mechanical failure reparations or maintenance activities.

So, relevant information for QA and dialogue for assistance can correspond to components of various sizes from an entire section or subsection, to a short span of text, or even be a table or a part of it, or an image. Thus, it is of key importance that converted documents are structured in a way that preserves all this information in a machine-readable format.

In order to enable dialogue applications to exploit the structural information of the source industrial technical documents in PDF format, the following machine-readable corpus structure has been defined:

- **Manual**. Each manual gives rise to a dialogue use case, so there is a folder for each manual.
- **Chapters**. The contents of the manuals are divided into chapters, so there is a folder for each chapter.
- **Sections**. Chapters are often divided into sections, so a JSON file is created for each section, comprising its contents in a structured way. If the chapter has no sections, a single file is created for the chapter's contents. Each

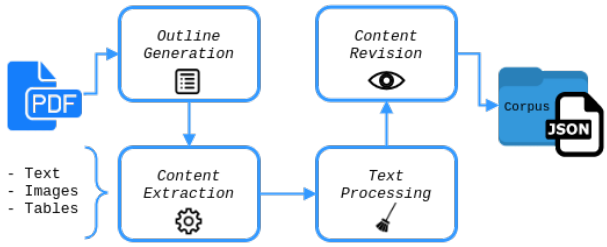

Figure 2: Main steps of the corpus compilation pipeline.

chapter folder also contains an *images* folder where the images extracted are stored.

- **Subsections**. Sections can be further divided into subsections, when they appear as part of the numbered structure (e.g., 4.1.1 is a subsection of section 4.1 in chapter 4), or what we called ***titles***, which are distinguished parts of the contents with their own name but which do not follow the numbered schema. We also consider as *title* the level below *subsection* when it exists, numbered or not.

- **Contents** in the deepest level. When the deepest level of a manual's structure is reached for each element of the nested listings (e.g., a subsection with no title elements as children), its contents are incorporated as a dictionary structure including: the raw text; the processed (clean) text; the images found for the excerpt, as a list of the names with which they have been stored; the list of tables that appear in the excerpt; and the final text after manual revision and corrections.

3 PDF Document Conversion Pipeline

The pipeline proposed to automatically transform the source industrial technical PDF documents into the machine-readable corpus structure introduced in Section 2 and depicted in Figure 2, consists of four main steps:

1. **Outline generation.** This outline is a structured representation in JSON format of the different levels of contents conforming the document.

2. **Content extraction.** This step consists of extracting the contents in the original manuals, using the outlines produced in step one to correctly locate fragments, and saving them in a structured form. This step includes the extraction of three types of contents of interest: text, images, and tables.

3. **Text processing.** The raw text extracted from the manuals is processed to produce a clean version containing only the text in the main body, free of undesired segments such as page headers and footers, text fragments that belong inside tables, or contents belonging to other fragments that have not been correctly delimited in the extraction step.

4. **Content Revision.** An optional revision can be carried out manually to ensure that the final contents of the documents are correct.

Detailed information about the tools that were tested and those that were finally integrated into the pipeline as well as on the implementation of the different steps of the pipeline are presented in the next subsections.

3.1 Outline Generation

PDF documents with a valid navigation-enabling outline, preserve information about their internal structure (i.e., chapters, sections, subsections, etc. and their starting pages) in their metadata. Exploiting this information enables to generate the base JSON structure of the outline in an automatic way.

The `pdftohtml` (Glyph & Cog, 2021b) PDF conversion utility of the `xpdf` (Glyph & Cog, 2021d) toolkit was used to extract the PDF's metadata relative to the contents outline in a structured HTML format that preserves the elements' hierarchy, titles and start pages. The result can be parsed to extract the relevant information and format it into the targeted JSON structure. The library used to parse the HTML information was `AdvancedHTMLParser` (Savannah, 2021). The fragments' end page numbers, which are not present in the PDF files outlines, are established by some heuristics and assumptions.

3.2 Content Extraction

There exists a variety of tools aimed at automatically extracting content from PDF files. However, after trying several, none of them seemed to be optimal for extracting all three targeted types of contents, text, images, and tables. For this reason, different methods were tested and selected for the extraction of each content type.

The methodology used for testing and selecting the optimal tools for content extraction involved

the following steps: 1) Applying each tool on a selected manual; 2) Analysing the produced result; 3) If the result was unsatisfactory the tool would be discarded; 4) The tools that performed best would be applied to other manuals to check whether the results were consistent on other documents from different manufacturers, and so deem them as acceptable.

3.2.1 Text Extraction

The following tools available which are oriented to the extraction of text from PDF documents were tested for our pipeline:

- `pdfplumber` (Singer-Vine, 2021). Although this library is more oriented to the extraction of tables from PDF files, its text extraction functionality was also tested. Results were bad, given that text spacing was not interpreted correctly by the extractor and words appeared joined together.
- `tika-python` (Mattmann, 2021) is a library that makes use of the Apache Tika toolkit[1] for extracting text from PDF. Results of applying this library across manuals were generally good, although there were some cases where page numbers appeared joined together with the main text, without spacing.
- `pdftotext` (Glyph & Cog, 2021c) is `xpdf`'s command line utility to extract text from PDF files. As no errors were detected when applying it across the manuals, this tool was selected and incorporated in the pipeline for text extraction.

3.2.2 Images Extraction

We found only a few tools capable of extracting images from PDF files. Some included little to no documentation on how to use this functionality. For example, `pdfplumber` extracts some image representation objects from the PDF's metadata, but it does not provide information on how to obtain the original images from those representations. The image extractors that produced valid results when applied to our manuals are:

- `pdfimages` (Glyph & Cog, 2021a) is a `xpdf`'s command line utility to extract images from PDF files. Images across manuals were correctly extracted and saved into a target folder. Errors were found with some images' formats.
- `PyMuPDF` (Jorj X. McKie and Liu, 2021) along with `Pillow` (Clark, 2021). The combination of both Python libraries allows extracting and saving images from PDF files. Results were comparable to the previous tool, some errors being still encountered with certain images. This method was selected and incorporated to the pipeline for image extraction.

3.2.3 Tables Extraction

As for table extraction, we only found a couple of tools aimed at extracting tables from PDF files. General content extractors such as the already-mentioned `pdftohtml` would not recognise tables. The tools specialised in table extraction that we tested are:

- `pdfplumber` (Singer-Vine, 2021): Despite this library is specifically oriented to extract tables from PDF files, the results of its application were bad. For some tables, only the header row was recognised as a table, and their text contents included spacing errors. Many other tables were simply not recognised as such.
- `camelot` (Mehta, 2021) is also a Python library specifically aimed at extracting tables from PDF documents, and it allows saving the extracted tables as JSON, among other formats. Results of applying this tool across manuals were mostly good, with a few cases of tables not automatically recognised as such. This was the tool selected and incorporated into the pipeline for table extraction.

Given that table extractors work at page level, a common consequence is that tables that span over several pages are always considered as individual, separate tables. Since in the manuals it is usual for tables to repeat the header row when continuing in a new page, a processing function was created to automatically join split tables when necessary.

3.3 Text Processing

This step of the pipeline aims to delete undesired information. Within text fragments this mainly includes headers and footers, page numbers, text fragments that belong inside tables, and content that has not been correctly delimited and does not belong to the current fragment. In some cases, it also includes expressions that do not provide meaningful information and hinder readability of the

[1] `http://tika.apache.org/`

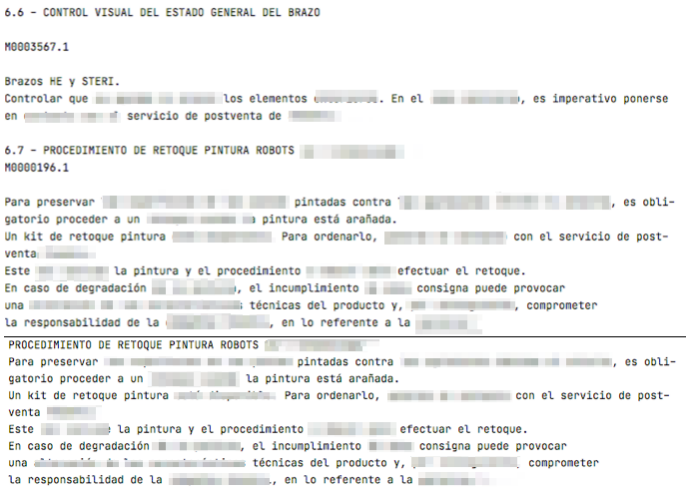

Figure 3: Above, text as extracted in step 2. Below, same text after applying the text processing step. Contents have been intentionally blurred to comply with copyright restrictions.

corresponding fragment (e.g. “Continued on next page" or alphanumeric codes used by the manufacturer, such as “M000..." in Figure 3).

Regular expressions are used to delete headers, footers, page numbers, and undesired information in general. They are fed from a configuration file adapted to each manual format, as it changes across industrial documents. Although some of the employed regular expressions are simple and straightforward, others get more complex to avoid deleting meaningful information inside the text. Figure 3 shows an example of a text fragment in which the footer, page number, and an alphanumeric code that deteriorates readability have been erased.

The process used to automatically delete content not belonging to a particular text fragment consists of the following steps: 1) The level of the current element is identified (section 6.7 in Figure 3); 2) Any text appearing before the title of the current element is deleted (in the example, text belonging to section 6.6 is deleted); 3) The title of the next element is identified and all text from that point on, if included, is erased from the current fragment.

3.4 Quality Revision

The final step of the pipeline involves an optional manual revision process in order to ensure the quality of the final corpus.

In order to make this process easier for human reviewers, each fragment to be reviewed is dumped into a text file following this format: (i) A header including the fragment’s starting and ending page in the PDF, plus the fragment’s title and a separator; (ii) The processed text as obtained from the previous step of the pipeline, and a separator; and (iii) A preview of the tables found for the current fragment, formatted as text using the `tabulate` (Astanin, 2021) library on the JSON table data. The first and third parts are provided as helping references, while the part appearing between separators is the one to be reviewed and possibly modified. This part is then loaded and included in the final JSON after revision.

3.5 Pipeline in use

The presented PDF document conversion pipeline has been applied and qualitatively validated extracting the information contained in the technical manuals listed in Table 1. Generally speaking, structuring and automatic content extraction have been quite satisfactory. This impression was confirmed at the content revision step, where three technical reviewers agreed that the information extracted was correct to a great extent. In addition, four expert technical operators have also provided a very positive opinion of the pipeline output, validating it for annotation and information presentation purposes within the project.

The outline generation module was capable of extracting an accurate content structure from the PDF files’ metadata. The selected content extraction tools were capable of extracting raw text, images, and tables without major problems across the varied set of analysed industrial technical documents. And the implemented text processing module allowed to adequately filter out undesired (e.g., headers and footers), irrelevant (e.g. “Continued on next page"), and duplicated information auto-

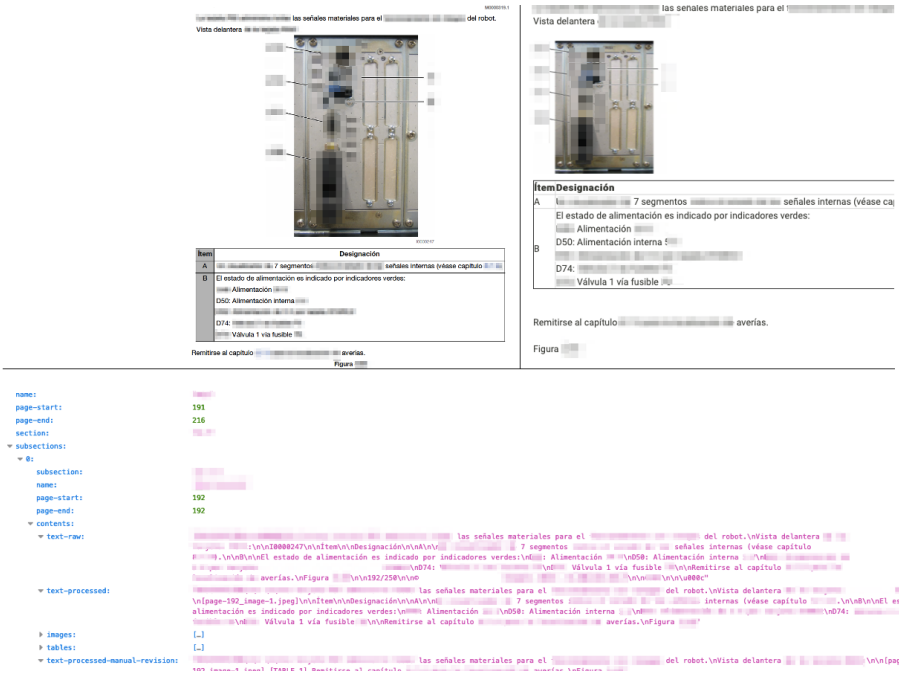

Figure 4: A sample fragment of a manual in its original PDF format (top left), the structured JSON produced by our pipeline (bottom), and recomposed for visualisation on an annotation tool (top right). Contents have been intentionally blurred to comply with copyright restrictions.

matically.

In a few occasions, some content extraction anomalies were detected, such as *missing text content*, which was mostly caused by the missing text being wrongly considered part of a table, and *additional text content*, the origin of which was mostly related with some subsections and their parent sections' names being equal, causing content delimiters to fail. Furthermore, additional content was also generated when tables were incorrectly extracted as text or text from images was considered as part of the main body of text.

Tables, although generally well-extracted, presented more problems than text content. The most common issues were related to *content* (empty, missing, or incomplete tables and multiple tables that were incorrectly joined) and *format* (misplaced cell content or moved content when cells had itemised or numbered lists).

All in all, the conversion pipeline is able to produce a comprehensive version of the original contents in PDF to a machine-exploitable structured JSON format.

4 Annotation Procedure

Once converted, the industrial technical documents were annotated by four skilled operators, in order to identify relevant question-answer pairs and procedures to develop both question-answering and dialogue systems for procedure assistance, respectively. The employed annotation tool (Justo et al., 2016) requires the input to be formatted as Markdown text, which allows inline HTML code. Knowing this, processed text was easily converted from the original JSON files and displayed by the tool with minimal changes (e.g. adapting line breaks). Nevertheless, references to images and tables still had to be converted from JSON to their HTML equivalents to be visualized.

Figure 4 depicts a fragment from a PDF manual, including text, an image, and a table, at three different stages: On the top left, the fragment is shown in its original PDF format; The bottom image illustrates the results of applying the PDF conversion pipeline, where contents are presented as a structured JSON; Finally, the image on the top right shows how the same contents are displayed by the annotation tool.

Two types of question-answer pairs were identified by annotators: Generic and Specific. Generic questions usually describe a procedure (e.g., disassembling part of a robot) and corresponded to whole subsections or, depending on the manual, sub-subsections. On the other hand, answers to specific questions were extracted from either subsections or sub-subsections and generally referred

Manual	# Chapters	# Running Words	Vocabulary size	Norm. Lev. distance
IRB120	3	3167	807	0.66
Controller	2	8368	1883	0.78
Robotic Arm	2	2329	780	0.93
F11	8	23302	3650	0.93
F12	5	37974	4862	0.70
CNC 8060/8065	15	61730	4975	0.91
PG 1106	11	46430	6794	0.81
TNC 426/TNC 430	12	65410	6270	0.86
RACK AA.81.51.2001	9	5019	1525	0.91

Table 2: Basic information about the number of chapters and words per manual as well as the normalized Levenshtein between the manually corrected text and the automatically corrected text.

to particular queries (e.g., what tools should I use to disassemble the part). In total and during six months, the four operators annotated 1416 generic and 751 specific question-answer pairs.

5 Corpus Description and Evaluation

Table 2 provides a general description of the obtained corpus in terms of number of chapters, number of running words and vocabulary size. In order to gain a better insight into the performance of the conversion pipeline we compare the manually corrected text with the text extracted by the system. To this end we considered the Levenshtein distance, which calculates the minimum number of changes, i.e. adding, deleting, or replacing a single character, that are needed to transform one string into the other.

In this way, we can evaluate how well the automatic text extractor and text processor based on regular expressions work. However, it heavily depends on the length of both strings to be compared and does not provide any meaningful information in our case, where the length varies significantly from one manual to the other. To solve this problem, we have adapted it to the normalized Levenshtein distance, defined in Equation 1

$$Lev_{norm} = 1 - \frac{Lev(s1, s2)}{max(len_{s1}, len_{s2})} \quad (1)$$

where $Lev(s1, s2)$ is the Levenshtein distance between the strings $s1$ and $s2$ and len_{s1} and len_{s1} are the length of the strings. Thus, the distance does not longer depends on the length of both strings. The value of the normalized Levenshtein varies between 1.0, where both strings are equal, and 0.0, where they are completely dissimilar. Table 2 incldues the normalised Levenshtein distance between the correct text with the one extracted through the proposed PDF conversion pipeline. This table shows a very good performance of the extraction and processing tool for most of the manuals, being higher than 0.78 for almost all of them. However, the improper operation of the extraction and processing tools lead to worse performance for both the IRB120 and F12 manuals. These manuals have numerous images, then the extraction tool retrieved information embedded in the images.

On the other hand, Table 3 shows the main characteristics of the QA corpus obtained after the annotation procedure described in Section 4. This table provides the number of generic (GQ) and specific (SQ) questions along with the average lengths of each manual, in terms of the number of words. This table shows a high variability in the length of the answers within the same manual, and from one manual to another. This fact, along with that some answers include images, shows the complexity of developing question-answering systems for the industrial environment. This complexity, for example, does not exist in other more popular extractive question-answer datasets such as SQuAD v1.1 (Rajpurkar et al., 2016) and v2.0 (Rajpurkar et al., 2018), where images are not included and the length of the answers is less variable. However, this dataset has been successfully used for a QA system, which has been tested in the framework of the same research project (Ruiz et al., 2023).

Finally, Table 4 provides information about the corpus for ontology-based DS, detailing instances derived from the 6 selected procedures in the corpus, represented according to the ontology described in (Aceta et al., 2022). From the total of 268 instances, 6 correspond to procedures, 8 to methods, 20 to tasks and 69 to steps. In a nut-

Manual	# Generic questions (GQ)	Avg. # words in GQ	% answers with images	# Specific question (SQ)	Avg. # words in SQ
IRB120	419	1342.2	0	159	101.1
Controller	133	2528.3	0	53	112.3
Robotic Arm	55	1285.7	0	36	88.1
F11	192	2452.9	26	62	120.7
F12	186	2030.9	35	144	182.5
CNC 8060/8065	124	2342.8	21	60	209.7
PG 1106	91	5667.9	27	47	194.1
TNC 426/430	164	2071.4	43	63	210.2
RACK AA.81.51.2001	52	1075.8	48	127	94.7

Table 3: Annotated QA dataset, which includes informations of General (GQ) and Specific Questions (SQ)

Class	Instances	Avg. instances per procedure	Avg. instances per upper-class	Additional info		Descriptions	
				Avg.	SD	Avg.	SD
Procedure	6	-	-	3.66	2.71	1	0
Method	8	1.33	1.33 (procedure)	0.75	1.39	1.63	4.6
Task	20	3.33	2.5 (method)	0.5	1.15	0.35	0.49
Step	69	11.5	3.45 (task)	0.52	0.61	0.49	0.68

Table 4: Ontology instantiation overview

shell, each procedure has an average of 1 method, each method an average of 3 tasks and each task an average of 4 steps.

The rest of the instances in the dialogue system ontology instantiation include other details such as the necessary materials to perform the procedure given a specific method or the descriptions and additional information for each procedure division class. As for the latter, Table 4 also shows the average numbers of instances covering each type of information per each class and the Standard Deviation (SD) to provide more objective insights on this data. As it can be seen in the table, the number of instances may differ significantly depending on the procedure or method.

6 Conclusions and Future Work

This work has been developed in the framework of a research project whose objective is to facilitate industrial maintenance, programming, manufacturing, and assembly tasks through the use of voice-based interfaces. Unlike classic dialogue systems in which the information to be supplied to the user is naturally structured in a database (e.g., restaurants, types of food, addresses, etc.), human-machine interaction assisting industrial operators' tasks needs to handle information that is usually found in PDF documents.

In this scenario, we have proposed a pipeline aimed at extracting content from technical PDF documents and converting them into a machine-readable format required for the automatic processing of their information. This procedure is able to extract and organise a variety of content types such as text, images, and tables in order to get a corpus of structured information organised in JSON files.

A qualitative analysis of the results of the proposed procedure has been carried out by expert technical operators, who have provided a very positive opinion validating the proposal. Moreover, normalised Levenshtein distance between the manually corrected text and the text generated by the pipeline is quite high showing a very good performance of the pipeline. This way, we are contributing to the scarce literature addressing multimodal content extraction from documents in PDF format, which is still mainly limited to text extraction.

In addition, one of the advantages of the proposed procedure to structure contents into JSON files is that it can be easily converted into other formats, such as HTML, for further visualisation or processing purposes. We have taken advantage of this feature to facilitate an annotation process carried out by expert technical operators. These annotations as well as the derived ontology instances have allowed the compilation of a useful corpus for the development of question answering and

assistance-oriented dialogue system applications in the industrial sector.

Acknowledgements

This work was supported by the Basque Government's Elkartek research and innovation program, through the EKIN KK-2020/00055) and BERREKIN (KK-2022/00102) projects

References

Cristina Aceta, Patricia Casla, Izaskun Fernandez, and Aitor Soroa. 2022. Kide4assistant: an ontology-driven dialogue system adaptation for assistance in maintenance procedures. In Kyritsis D. Sarkar A. Sanfilippo E.M., Karray M., editor, *Proceedings of the 12th International Workshop on Formal Ontologies Meet Industry (FOMI 2022) Co-Located with Workshops About the Industrial Ontology Foundry (IOF) and the European Project OntoCommons (EU H2020 Project). Tarbes, France, September 12-15*, volume 3240. CEUR Workshop Proceedings.

Dario Antonelli and Giulia Bruno. 2017. Human-Robot Collaboration Using Industrial Robots. In *2017 2nd International Conference on Electrical, Automation and Mechanical Engineering (EAME 2017)*, pages 99–102. Atlantis Press.

Sergey Astanin. 2021. python-tabulate. https://pypi.org/project/tabulate/. Accessed: August, 2023.

Hannah Bast and Claudius Korzen. 2017. A benchmark and evaluation for text extraction from pdf. In *2017 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, pages 1–10.

Alex Clark. 2021. Pillow. https://pypi.org/project/Pillow/. Accessed: August, 2023.

Zihui Dong, Shiladitya Paul, Karl Tassenberg, Geoff Melton, and Hongbiao Dong. 2021. Transformation from human-readable documents and archives in arc welding domain to machine-interpretable data. *Computers in Industry*, 128:103439.

Glyph & Cog. 2021a. pdfimages – Portable Document Format (PDF) image extractor. https://www.xpdfreader.com/pdfimages-man.html. Accessed: August, 2023.

Glyph & Cog. 2021b. pdftohtml – Portable Document Format (PDF) to HTML converter. https://www.xpdfreader.com/pdftohtml-man.html. Accessed: August, 2023.

Glyph & Cog. 2021c. pdftotext – Portable Document Format (PDF) to text converter. https://www.xpdfreader.com/pdftotext-man.html. Accessed: August, 2023.

Glyph & Cog. 2021d. Xpdf – PDF viewer and toolkit. https://www.xpdfreader.com/. Accessed: August, 2023.

Ander González-Docasal, Cristina Aceta, Haritz Arzelus, Aitor Álvarez, Izaskun Fernández, and Johan Kildal. 2021. Towards a natural human-robot interaction in an industrial environment. In Luis Fernando D'Haro, Zoraida Callejas, and Satoshi Nakamura, editors, *Conversational Dialogue Systems for the Next Decade*, pages 243–255. Springer Singapore, Singapore.

Jorj X. Jorj X. McKie and Ruikai Liu. 2021. PyMuPDF. https://pypi.org/project/PyMuPDF/. Accessed: August, 2023.

Raquel Justo, J Alcaide, and M Torres. 2016. Crowdzientzia: Crowdsourcing for research and development. *Proceedings of IberSpeech*, pages 403–410.

Raquel Justo, Oscar Saz, Vîctor Guijarrubia, Antonio Miguel, M. Inês Torres, and Eduardo Lleida. 2010. Improving dialogue systems in a home automation environment. ICST.

Yuqian Lu, Hao Zheng, Saahil Chand, Wanqing Xia, Zengkun Liu, Xun Xu, Lihui Wang, Zhaojun Qin, and Jinsong Bao. 2022. Outlook on human-centric manufacturing towards industry 5.0. *Journal of Manufacturing Systems*, 62:612–627.

Chris A. Mattmann. 2021. tika-python. https://github.com/chrismattmann/tika-python. Accessed: August, 2023.

Vinayak Mehta. 2021. Camelot: PDF Table Extraction for Humans. https://camelot-py.readthedocs.io/. Accessed: August, 2023.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Siva Reddy, Danqi Chen, and Christopher D Manning. 2019. Coqa: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266.

D. Romero, J. Stahre, and M. Taisch. 2020. The Operator 4.0: Towards Socially Sustainable Factories of the Future. *Computers & Industrial Engineering*, 139.

Eneko Ruiz, María Inés Torres, and Arantza del Pozo. 2023. Question answering models for human–machine interaction in the manufacturing industry. *Computers in Industry*, 151:103988.

Tim Savannah. 2021. AdvancedHTMLParser. https://pypi.org/project/AdvancedHTMLParser/. Accessed: August, 2023.

Manex Serras, María Inés Torres, and Arantza Del Pozo. 2020. Improving Dialogue Smoothing with A-priori State Pruning. In *ICPRAM*, pages 607–614.

Jeremy Singer-Vine. 2021. pdfplumber. https://github.com/jsvine/pdfplumber. Accessed: August, 2023.

Jörg Tiedemann. 2014. Improved Text Extraction from PDF Documents for Large-Scale Natural Language Processing. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 102–112. Springer.

M. Inés Torres. 2013. Stochastic bi-languages to model dialogs. In *Proceedings of the 11th International Conference on Finite State Methods and Natural Language Processing*, pages 9–17, St Andrews, Scotland. Association for Computational Linguistics.

Alain Vázquez, Asier López Zorrilla, Javier Mikel Olaso, and María Inés Torres. 2023. Dialogue management and language generation for a robust conversational virtual coach: Validation and user study. *Sensors*, 23(3).

Jinjiang Wang, Yulin Ma, Laibin Zhang, Robert X. Gao, and Dazhong Wu. 2018. Deep learning for smart manufacturing: Methods and applications. *Journal of Manufacturing Systems*, 48:144–156. Special Issue on Smart Manufacturing.

Tian Wang, Jiakun Li, Zhaoning Kong, Xin Liu, Hichem Snoussi, and Hongqiang Lv. 2021. Digital twin improved via visual question answering for vision-language interactive mode in human–machine collaboration. *Journal of Manufacturing Systems*, 58:261–269. Digital Twin towards Smart Manufacturing and Industry 4.0.

Liu Xingguang, Cheng Zhenbo, Shen Zhengyuan, Zhang Haoxin, Meng Hangcheng, Xu Xuesong, and Xiao Gang. 2022. Building a question answering system for the manufacturing domain. *IEEE Access*, 10:75816–75824.

Mauricio Zamora, Eldon Caldwell, Jose Garcia-Rodriguez, Jorge Azorin-Lopez, and Miguel Cazorla. 2017. Machine Learning Improves Human-Robot Interaction in Productive Environments: A Review. In *IWANN2017: Advances in Computational Intelligence*, pages 283–293, Cham. Springer International Publishing.