Structured Tender Entities Extraction from Complex Tables with Few-short Learning

Asim Abbas[1*], Mark Lee[1], Niloofer Shanavas[2], Venelin Kovatchev[1], Mubashir Ali[1]
[1]School of Computer Science, University of Birmingham, Edgbaston, Birmingham, UK
[2]School of Computer Science, University of Birmingham, Dubai Campus, UAE
Correspondence: axa2233@student.bham.ac.uk

Abstract

Extracting structured text from complex tables in PDF tender documents remains a challenging task due to the loss of structural and positional information during the extraction process. AI-based models often require extensive training data, making development from scratch both tedious and time-consuming. Our research focuses on identifying tender entities in complex table formats within PDF documents. To address this, we propose a novel approach utilizing few-shot learning with large language models (LLMs) to restore the structure of extracted text. Additionally, handcrafted rules and regular expressions are employed for precise entity classification. To evaluate the robustness of LLMs with few-shot learning, we employ data-shuffling techniques. Our experiments show that current text extraction tools fail to deliver satisfactory results for complex table structures. However, the few-shot learning approach significantly enhances the structural integrity of extracted data and improves the accuracy of tender entity identification.

1 Introduction

Tenders are formal requests for proposals or bids, typically issued by a company, organization, or government agency seeking goods, services, or works to be provided (Siciliani et al., 2023b). In addition tender documents are the detailed specifications, terms, and conditions accompanying such requests, outlining the requirements and expectations for potential bidders. These documents ensure transparency, fairness, and accountability in the procurement process and are vital for decision in project management (Toikka et al., 2021).

The tender documents contains meaningful information that must be identified and extracted automatically to convert it into actionable knowledge to improve business decisions. Generally, this information is available in an unstructured or semi-structured format (Siciliani et al., 2023a), which is understandable by humans but difficult to understand by machines because of a lack of documents or text structure, contextual understanding, ambiguity and noise, and limited domain knowledge. Similarly, tender documents are large in size, often consist of over 100 pages each. Manually extracting relevant information from such huge documents requires a lot of energy and time and is a labor-intensive task often prone to errors and inefficiencies. To address these challenges, Natural Language Processing (NLP) based applications and techniques have emerged as a promising solution (Fu et al., 2020).

This study addresses two key contributions crucial for automating tasks in the tender domain: i) Structured text extraction from complex tables, a persistent challenge, is essential for tender documents as tables hold organized information vital for accurate analysis. Losing the structure during extraction can result in misinterpretation, affecting decision-making in the tendering process (Milosevic et al., 2019). ii) Tender Named Entities (TNE) recognition and classification, including addresses, project details, dates, and personnel, are critical for retrieving relevant information, generating recommendations, and automating systems like chatbots and IR systems (Ji et al., 2019; Siciliani et al., 2023b; Ji et al., 2019)

To address these challenges, we have introduced a novel approach that combines the capabilities of few-shot learning with large language models (LLMs) (Brown, 2020). Our approach aims to reconstruct the text structure after extraction, thereby facilitating the accurate identification of tender elements. Initially, we leverage existing pdf text extraction tools like pdfminor (PDFMinersix, 2024) to extract raw text from tender documents, focusing on entities within tables. Subsequently we identify the common terms that assist in document segmentation into header, body and footer. We discarded the body text to reduce the data dimensionality and

Proceedings of the 31st International Conference on Computational Linguistics, pages 59–67
January 19–24, 2025.

improve entities categorization. Further, header and footer text is concatenated, and few-shot learning with LLM is leveraged to restructure it. After restructuring text, we employed hand crafted rules and regular expressions to automatically classify the entities into explicit categories. Consequently, we achieve high accuracy towards tender entities extraction and classification through few-short learning approach compared to without few-short learning.

Furthermore, our study included comprehensive evaluation, comparison and limitation of existing tools utilized for structure text extraction from tender PDF tables. As a result we found that not a single tool provided desire performance towards structure text extraction from tender PDF tables because of the unstructured and dynamic structured of the tables. Similarly, every tool have their own strength geared specific task. In the same way, in our study, we experimented and combined these tools in one place, utilized their explicit features, and developed our own algorithm to automatically extract the tender elements using rules and regular expressions. Consequently, we achieved state-of-the-art accuracy by integrating a few-shot learning approach.

Further, this study is structured as Related Studies are presented in Section 2. Continuing this, we presented Proposed Approach in section 3. Additionally, in section 4 we presented Experiments, Evaluation and Results of the proposed solution. Finally, we conclude the study with limitation and future work in Section 5.

2 Related Studies

Over the years, a range of methodologies has been explored to enhance the accuracy of entity extraction from complex tender documents, including rule-based systems, machine learning (ML) approaches, and more recently, deep learning(DL) models. While these methods have demonstrated varying degrees of success, the heterogeneity and complexity of tender documents often lead to issues with text structure and format retention, making the task of extracting accurate and relevant entities even more challenging.

In the study (Mehrbod and Grilo, 2018), a rules and self-learning approach using a Conditional Random Field (CRF) model has been introduced to automatically create and update the dictionary over time for recognizing the product entities in tender documents. Moreover, an ontology-based approach for information extraction from construction tender documents has been introduced to convert human-readable document structures into machine-readable formats (Mohemad et al., 2011). However, It is a challenging task to develop universal rules for the entire system, capture intricate semantic links between words, and manage named entities, especially in dynamic specialized domains. Similarly, an incomplete dictionary can lead to a low recall while making a complete dictionary manually is tedious and time-consuming.

To address the challenges with Rule-based systems, state-of-the-art machine learning approaches have solved these challenges extensively in the Tender domain for task such as Named Entities Recognition (Hastie et al., 2009). In the study (Siciliani et al., 2023a) an open Information Extraction for Public Administration (OIE4PA) system was introduced for tender information retrieval from large databases. It extracts information based on triples (Subject, Predicate, Object) found in documents, using tools like UDPipe[1] and WikiOIE (Siciliani et al., 2021). In this process, two domain experts manually labeled these triples, which served as the training data for machine learning models such as Support Vector Machines (SVM), XGBoost, and logistic regression (LR). However, machine learning models may not always reach peak performance on labeled data due to potential issues of over-fitting or under-fitting. This sometimes requires the use of intricate feature engineering approaches.

Conspicuously, DL is differs from the classical ML approaches by diminishing the demand for manually designing features such as bag-of-words or n-grams (Wu et al., 2020; Medsker and Jain, 2001; Wolf et al., 2020). As discussed previously, this study (Chalkidis et al., 2017) is further extended by implementing deep learning models such as Bi-LSTM, LSTM, and Conditional Random Field (CRF) to extract the contract entities automatically (Chalkidis and Androutsopoulos, 2017). Moreover, a DL-based approach is introduced (Ji et al., 2019) to enhance the automatic identification of tender entities. The proposed architecture incorporates five main layers, for instance embedding layer using BERT, BiLSTM input BERT embedding as feature vector, feature fusion layer, attention layer, and CRF layer. Similarly, RNN-based architecture models have several constraints and lim-

[1] https://lindat.mff.cuni.cz/services/udpipe/

itations. These include the vanishing gradient problem during back-propagation through time, which makes it challenging for the model to learn long-range dependencies in sequential data. Furthermore, RNNs are sensitive to hyper-parameters such as learning rate, batch size, and sequence length.
To address the limitations of existing methodologies, we introduce a Large Language Model (LLM)-based approach that incorporates few-shot learning for tender entity extraction and classification. By leveraging LLMs, we eliminate the need to build AI models from scratch, which typically requires vast training datasets and the involvement of domain experts is costly and time-consuming process. Furthermore, we combine this AI-driven approach with rule-based methods to enhance accuracy in tender entity extraction and classification, while eliminating the need for ongoing lexicon and rule maintenance across the entire system.

3 Proposed Approach

This section outlines the methodology for structured tender text extraction and entity classification. We first used tools such as PDFMiner (PDFMiner-six, 2024) to extract text from complex PDF tables. A text analyzer was then designed to perform preprocessing, keyword identification, and document segmentation into header, body, and footer. After discarding the body, we concatenated the header and footer. Due to the loss of semantics, context, and sequence in extracted text, we employed a few-shot learning approach to restructure it. Finally, regular expressions and rules were applied to classify tender entities into defined categories.

3.1 Tender Table Structure and Entities

The tender entities of interest are found in the first section of each document, and formatted as complex and dynamic tables. Each tender typically begins with a preamble containing the Tendree (Buyer) title (see Figure 1), followed by the Tenderer (Supplier) information, including company name, address, date, tender number, and project name. The body of the tender includes the project description, supplied items, and terms and conditions. At the end, Tendree information, such as name, address, signatory details, and phone number, is listed.
These entities (Tendree and Tenderer) are located in specific table zones but within a dynamic structure. The width and height of table cells change depending on the length of the tender entities. For example, a longer Tenderer name or address will expand the table to cover more rows in a cell, but the information is always located in the start zone. Similarly, Tendree information is consistently found in the footer zone.
The table structure (Figure 1) is highly complex due to frequent merging and splitting of rows and columns, causing alterations during data extraction. Names, addresses, dates, job numbers, and other entities are scattered across multiple columns and rows, requiring careful extraction.

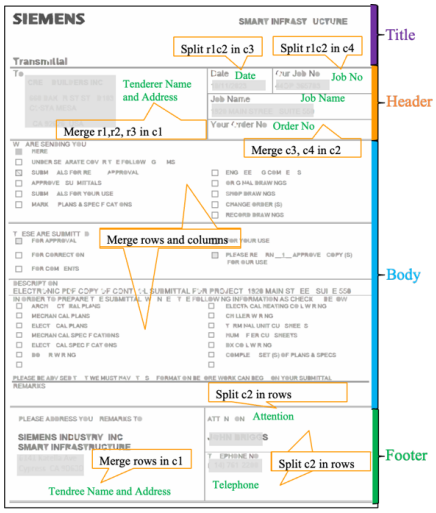

Figure 1: Typical structure of a table in tender document, with tender entities highlighted

3.2 Text Extraction Complexity

At the top of the table there is a title of Tendree, which appears on both the left and right corners in bold (Figure 1). If any part of the title, like "SIEMENS" on the left or "SMART INFRASTRUCTURE" on the right, is missed, existing tools often fail to recognize it. Similarly, the table header contains five tender entities: the tender name and address share a cell, while the tender date, number, and project name are in separate cells. Structurally, the table header has two columns with four cells, though row numbers vary based on the length of elements specifically tender name, address, and project name. In the same way, the footer section has two columns and three cells, with the tender address cell expanding due to longer addresses. Additionally, tender elements within cells may not align consistently, causing gaps between attribute and value horizontally or vertically. This inconsistency leads to the loss of text sequence, as existing tools read PDFs row-wise.

3.3 Text Analyzer

Subsequently, text extraction from tables using tools such as pdfminer (PDFMinersix, 2024) revealed a loss of text sequence, semantics, and context, making it difficult for both humans and machines to interpret. This also complicates applying rules and regular expressions for tender element identification. To address this, we introduced a text analyzer module, which incorporates text preprocessing, keyword identification, and keyword-based segmentation of tender information.

3.3.1 Text Processing

After extracting text from tender PDF tables, we observed that punctuation marks were misplaced, which caused issues with the regular expressions specifically designed for identifying tender elements. For example tender number in table is appear in *"44OP-123456"*. After extraction a punctuation dot(.) has added in the start or end of tender number such as *".44OP-123456"* or *"44OP-123456."*, that become challenging to maintain the regular expression. Similarly, after text extraction, extra spaces between words both horizontally and vertically were found, making the text more challenging to process, reducing readability, and causing inconsistencies during parsing.

3.3.2 Keywords Identification

After extracting text from the table, a lack of predefined format was observed. Identifying keywords helps recognize patterns and structure. Additionally, Keywords help in extracting relevant information, making it easier to categorize and segment the text. For instance to extract the header information the keyword *"PROJECT NAME"* help to extract all the text before this. Similarly extracting body information, the keywords “FOLLOWING ITEMS", and *"REMARKS TO"* helps to extract text between these words. Finally the keyword *"SIEMENS INDUSTRY"* etc help to extract the footer information.

3.3.3 Keywords based Document Segmentation

To accurately segment unstructured text into header, body, and footer, we introduced a keyword-based document segmentation approach to enhance algorithm efficiency and performance in extracting Tender Named Entities (TNE) from tender tables. Unnecessary text increases complexity, so identifying key phrases helps reduce this. As shown in Figure 1, TNEs are located before the phrase *"WE ARE SENDING YOU"* and extracted by recognizing the table attribute *"JoB Name"*. The header entities are extracted from text preceding this phrase, while the body is between *"WE ARE SENDING YOU"* and *"REMARKS TO"*. Similarly, footer TNEs are located between *"SIEMENS INDUSTRY"* and *"TELEPHONE NO"*. By discarding the body section and combining header and footer data, we reduce dimensionality, making the data more manageable, lowering storage needs, and speeding up processing.

3.4 Structured text generation employing few-Shot learning

When header and footer text is concatenated, the resulting text become unstructured with sequence, semantic and context lost, posing significant challenges for further processing. To address this, we proposed a few-shot learning approach leveraging LLM effectively. To enable the LLM to handle this unstructured data, a set of manually prepared structured example samples is created. These examples consists of pairs of unstructured data and their corresponding structured formats, showcasing the desired outcome. These example samples are then used to guide the LLM through prompt engineering, teaching it to understand and convert unstructured data into a structured format. Using in-context learning, new unstructured data can be processed by the model to generate structured outputs efficiently. This approach ensures that the LLM can effectively maintain the correct sequence and organization of data, transforming unstructured text into a well-structured format suitable for further use.

3.4.1 Case Study

The Figure 2 illustrates a case study on few-shot learning prompting using a ChatGPT-3.5 (LLM) for structure tender text generation.The diagram is splited into two main section, each demonstrating the conversion of unstructured input text into a structured format using example-based learning. The top section outlines a few-shot learning approach where example pairs of unstructured (Input) and structured (Output) texts are provided to the LLM with the specific instruction known as instructed prompt. The input data, such as *"TRANSMITTAL SMART INFRASTRUCTURE To: FANDERSON RAVE & BULCKLEY INC..."*, is processed to generate a corresponding structured output, showcasing the model’s ability to learn from

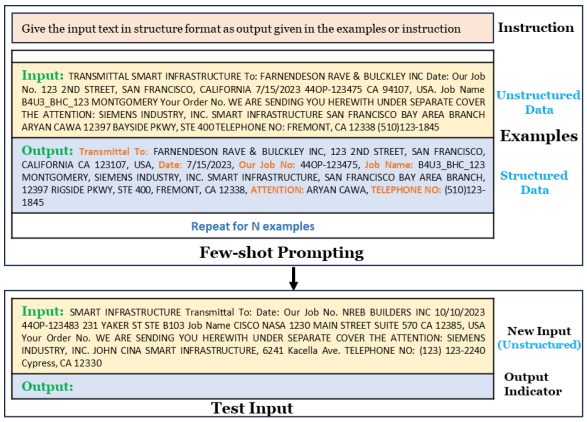

Figure 2: An example case study for understanding few-shot Learning approach towards structured data generation

minimal examples. Below this, the diagram repeats the process for multiple examples to reinforce the model's learning. The bottom section shows a new unstructured input(Test Input) being transformed into a structured output based on the learned examples. The prompt provided at the top guides the model to apply the learned structure to new inputs, demonstrating the efficacy of few-shot learning in generating accurate structured data from unstructured text with minimal examples. Ultimately, structure tender text output, heuristic approach is applied to identify and classify the tender elements into explicit categories.

3.5 Tender Entities Recognition and Classification

After concatenation of header and footer text, we acquired a text paragraph incorporated Tender Named Entities(TNE). We have designed a set of hand crafted rules and regular expressions for explicit entities recognition and classification. As shown in Table 1, the same regular expression is used for identifying the names and addresses of both the Tendree and Tendrer, as their naming patterns are similar. However, they have distinct content: the names of the Tendree and Tendrer typically end with "$INC|COMPANY$", while addresses start with a house or building number and end with a postal code followed by the country. Similarly, Tender date appeared in various format for example "$dd/mm/YYYY$" or "$dd-mm-YYYY$". Further, the Tender Number is always started with the two digit followed by two character and any six digit value after hypen. Moreover, the Tender Name is always located between *"Job Name"* and *"Your Order No*". However, sometimes the Tender Name may resemble the Tenderer's address, which can lead to incorrect classification as the Tenderer's address. Similarly, Tenderee Personal entity is challenging to accurately identify because every tender have variant personal name but it usually available in the text before *"TELEPHONE NO:"* pattern. Finally in footer Tenderee Telephone number is available in various format such as *"xxx-xxx-xxxx"* or *"(xxx)xxx-xxx"* or *"xxx.xxx.xxx"* etc. Our algorithm first attempts to extract entities using regular expressions. If any entities are missed, we have developed rules that work in conjunction with regular expressions to identify and extract the tender entities.

4 Experiments, Evaluation and Results

We have utilized 30 commercial confidential tender documents, that can not be released. Each document contains over 100 pages of information intended for architects, engineers, or project owners, submitted for approval by the contractor. The required tender entities are available in the form of complex table, which is difficult to extract accurately. In this section, we present the results of our Pre-fewshot and Post-fewshot learning approaches, which are combined with rules and regular expressions for structured text extraction from complex tables and the classification of tender entities.

4.1 Structured Text Extraction From Complex Table

To evaluate the structure text extraction utilizing existing tools by integrating few-shot learning approach, we explored two empirical approaches: a) Average Relative Distance (ARD) and b) BLEU score. Similarly, we computed the results in two different way: i) structured text extraction Pre-few-shot learning and ii) structured text extraction Post-few-shot learning. Similarly, we incorporated data shuffling technique to deterministically shuffle data 50% and 100% to asses the strength of LLMs in a few-shot learning environment towards structured data generation.

4.1.1 Average Relative Distance (ARD)

The Average Relative Distance (ARD) is a metric used to quantify the average movement of words from their positions in the original text to their positions in the extracted text. The resulting ARD value provides an indication of how much, on average, the words have shifted from their original locations

Table 1: Tender Named Entities Recognition(TNER) Regular Expressions

Named Entities	Regular Expression
Tenderee/Tenderer Name	.*INC$|. $* COMPANY.$
Tenderee/Tenderer Address	\b\d+\s.*?\d+\b
Tender Date	(\d{1,2}\/\d{1,2}\/\d{4})|(\d{1,2} − \d{1,2} − \d{4})
Tender Number	(\d{2}[A − Z]{2} − d{6})
Tender Name	Job Name\s*(.*?)\s*Your Order No
Tenderee Personal	∧(.*?)\bTELEPHONE\s?NO :?\b
Tenderee Telephone	\b(?:\+ \d{1,2}?)?\(?\d{3}\)?[-.\s]?\d3[-.\s]?\d{4}\b

in the extracted text. The Average Relative Distance (ARD) emperical formula can be expressed as shown Eq.1.

$$\text{ARD} = \frac{1}{N}\sum_{i=1}^{N}|p_i - q_i| \tag{1}$$

where N is the number of words in the original text, p_i is the position of the i-th word in the original text, and q_i is the position of the i-th word in the extracted text. ARD score lower (↓) is better and score higher(↑) is worst. The Figure 3 demonstrates the effectiveness of applying few-shot learning approach to improve structured text extraction from complex tables in tender PDF documents by computing ARD. Before few-shot learning, all tools struggled to maintain the structure and sequence of the extracted text, as indicated by relatively high ARD values. Among the tools, PyMUPDF performed the best initially with an ARD of 2.88, while PyPDF2 had the worst performance with an ARD of 7.22. As the data was shuffled, the ARD values increased across all tools, with the highest distortion occurring at 100% shuffling, where PyMUPDF and PyPDF2 reached ARD values of 10.99 and 14.18, respectively. This indicated severe structure loss with increasing shuffling.

After applying few-shot learning using ChatGPT-3.5, there was a remarkable reduction in ARD across all tools, showing the approach's effectiveness in restructuring the text. UnStructured.io and PyMUPDF showed the most significant improvements, with post-few-shot ARD values dropping as low as 0.37 and 0.49, respectively, in non-shuffled conditions. Even with 100% shuffled data, these tools maintained relatively low ARD values of 3.41 for PyMUPDF and 3.84 for UnStructured.io. PDF Minor and PyPDF2, while benefiting from few-shot learning, still had slightly higher Post-few-shot ARD values (around 4.27 and 3.04, respectively) when dealing with fully shuffled data.

Overall, few-shot learning drastically improved text extraction performance, particularly for UnStructured.io and PyMUPDF, which consistently achieved the lowest ARD values across all conditions. The results clearly demonstrate that applying few-shot learning to these tools can effectively restore text structure, even when the data is shuffled, although some tools still exhibit minor distortions in highly shuffled scenarios.

4.1.2 Average BLEU Score

BLEU (BiLingual Evaluation Understudy), is a metric used in NLP and machine translation to evaluate the quality of candidate text against one or more high quality reference text . It measures how similar a Tools extracted text is to one or more manually structured text.

The BLEU score is based on precision of n-grams (up to a certain length) between the candidate sentence and reference sentences. Here's the mathematical formula for BLEU score:

$$\text{BLEU}_{\text{avg}} = \frac{BP \cdot \exp\left(\sum_{n=1}^{N} w_n \cdot \log(p_n)\right)}{T} \tag{2}$$

Where, BP is the brevity penalty, when the candidate text length less than/equal ($\leq$) to the reference text length, w_n are the weights for each n-gram precision, p_n is the precision for n-grams, N is the maximum n-gram length considered and T is divide the sum of BLEU scores by the total number of sentence pairs. Notably, Average BLEU Score(ABS) higher (↑) is better and score lower (↓) is worst.

The Figure 4 shows the ABS for structured text extraction from complex tables in tender PDFs using five tools before and after applying few-shot learning with ChatGPT-3.5.

Before few-shot learning, BLEU scores dropped as shuffling increased, indicating a loss of text struc-

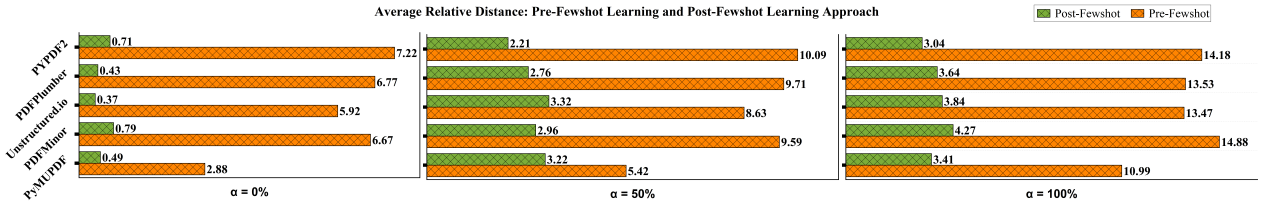

Figure 3: Performance of Structured Tender Text Extraction from Complex Tables: Pre- and Post-few-Shot Learning computed Average Relative Distance utilizing Text Extraction Tools and LLMs

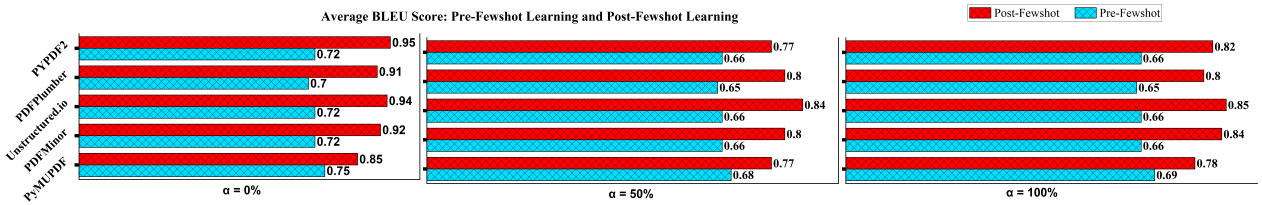

Figure 4: Performance of Structured Tender Text Extraction from Complex Tables: Pre- and Post-few-Shot Learning computed Average BLEU score utilizing Text Extraction Tools and LLMs

ture. PyMUPDF, for example, had a BLEU score of 0.75 for unshuffled data, which fell to 0.68 and 0.69 with partial and full shuffling, respectively. Similar patterns were observed with the other tools, where Pre-few-shot BLEU scores ranged from 0.65 to 0.72. After few-shot learning, all tools saw significant improvements in BLEU scores. PyMUPDF's score increased to 0.85 (unshuffled), 0.77 (50% shuffled), and 0.78 (100% shuffled), reflecting better text structure restoration. Likewise, UnStructured.io and PyPDF2 performed the best, with Post-few-shot scores of up to 0.95 and 0.94, respectively, in unshuffled and shuffled conditions. PDF Minor and PDFPlumber also improved but trailed slightly behind.

Inclusively, few-shot learning greatly improved structure preservation across all tools, with UnStructured.io and PyPDF2 emerging as the top performers, particularly in shuffled data scenarios.

4.2 Tender Entities Classification

We have developed a scoring-based method to evaluate the Named Entity Recognition and Classification of tenders, comparing the results obtained before and after applying a few-shot learning approach. The point scoring criteria is set as Full Match (F_i): 2 points, Partially Match(P_i): 1 points, Not Match (N_i): -1 point (small penalty for not matching) and Wrong Match (W_i): -2 points (higher penalty for incorrect matching) as shown in Equ. 3.

In Equ. 4, for a document with n categories, where each category can score a maximum of 2 points (for a Full Match) and minimum of -2 point (Wrong Match), So the maximum possible score is $S_{max} = 2{*}n$ and minimum possible score is $S_{min} = -2{*}n$. Similarly, we plan to evaluate Pre-few-shot learning and Post-few-shot learning approach at entities level and documents level.

$$S_{actual} = \sum_{i=1}^{n} S_i = \sum_{i=1}^{n} (2F_i + P_i - N_i - 2W_i), \tag{3}$$

Where S_i is calculated score of each category, and n is the number of categories.

$$Accuracy(\%) = \left(\frac{S_{actual} - S_{min}}{S_{max} - S_{min}} \right) \times 100 \tag{4}$$

The Table 2 presents a comprehensive analysis of accuracy improvements for nine tender entities before (Pre-Fewshot) only rules and regular expression and after (Post-Fewshot) applying the few-shot learning approach integrating rules and regular expression. These results are evaluated across five different text extraction tools: PDFMiner, PyMUPDF, Unstructured.io, PDFPlumber, and PyPDF2.

Overall, the results clearly indicate that the Post-few-shot learning approach significantly improves the accuracy of tender entity extraction across all tools. For instance, the entity "To Company" achieves 100% accuracy Post-few-shot learning across all tools, with notable improvements in tools like Unstructured.io (from 82.67% to 100%) and PDFPlumber (from 85.33% to 100%). Similarly, "To Address" shows considerable improvements,

Table 2: Entity-level tender documents evaluation Pre- and Post-Few-shot Learning Approach

Entities	PDFMiner (%)		PYMUPDF (%)		Unstructured.io (%)		PDFPlumber (%)		PYPDF2 (%)	
	Pre	Post	Pre	Post	Pre	Post	Pre	Post	Pre	Post
To Company	89	100	97.63	100	82.67	100	85.33	100	81.67	100
To Address	93	100	95	100	92	100	70.33	100	63	95.33
Date	100	100	100	100	95	100	100	100	95	100
Tender No	100	100	100	100	100	100	100	100	95	100
Tender Name	95	100	97.3	92	80	100	91.67	100	94.33	100
From Company	100	100	90	92	100	100	89.33	100	83.67	100
From Address	98.67	100	87	92	100	100	87.33	100	82	100
Attention	94.33	100	91.33	92	80.33	100	37.67	100	38	100
Telephone No	100	100	90	100	92.67	100	63.33	100	49.67	100

especially with PyMUPDF (from 95% to 100%) and Unstructured.io (from 92% to 100%). The "Date" and "Tender No" entities maintain 100% accuracy across all tools in both Pre- and Post-few-shot learning phases, indicating these entities are well-recognized due to pattern consistency regardless of the method used. However, other entities such as "Attention" and "Telephone No" benefit considerably from the few-shot learning approach. For example, "Attention" sees an impressive boost in PDFPlumber (from 37.67% to 100%) and PyPDF2 (from 38% to 100%).

The results also highlight the variability in pre-few-shot performance among the tools, with some, such as PyPDF2, initially struggling with lower accuracy rates across several entities, like "From Address" (82% pre, 100% post) and "Telephone No" (49.67% pre, 100% post). However, the application of few-shot learning substantially bridges these gaps across all tools, showing that the model can effectively generalize from a few examples.

Concisely, few-shot learning demonstrates its utility in improving entity extraction, particularly in tools like Unstructured.io, PDFPlumber, and PyPDF2, which showed weaker performance in the pre-few-shot phase. Across all tools and entities, the Post-few-shot learning results consistently approach or reach 100% accuracy, underscoring the approach's effectiveness in entities extraction from complex tender tables.

5 Conclusion, Limitation and Future Work

Structured text extraction from the complex table is an ongoing research challenge despite various AI tools and techniques. This study leverages LLMs in a few-shot learning environment to enhance tender entity classification from complex tables in PDF tender documents. We integrated text extraction tools, rules, and regular expressions with LLMs, and introduced text shuffling (50% and 100%) to assess LLMs capability in structured text extraction. After obtaining structured text, we applied handcrafted rules and regular expressions for precise entity classification. Similarly, we assessed several text extraction tools towards structured data extraction, as a result we found that not a single tool provided desire performance. The experimental results demonstrate that after text extraction from tables employ few-shot learning significantly improves performance and accuracy, addressing the challenge of structured text extraction from complex tender tables.

However this research has several limitations to be addressed in future work. First, the model's performance is heavily dependent on large datasets, especially in the Pre-few-Shot learning phase, where accuracy may decline with smaller datasets. Additionally, the few-Shot learning approach needs to be tested on datasets from other domains for a more robust evaluation. Finally, exploring zero-shot and one-shot learning approaches would further validate our findings.

In future, we aim to address the limitations of our study as discussed by expanding the dataset size and diversity to improve the accuracy and reliability of few-shot learning approach for structured data extraction and tender name entity recognition.

References

Tom B Brown. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Ilias Chalkidis and Ion Androutsopoulos. 2017. A deep learning approach to contract element extraction. In *JURIX*, volume 2017, pages 155–164.

Ilias Chalkidis, Ion Androutsopoulos, and Achilleas Michos. 2017. Extracting contract elements. In *Proceedings of the 16th edition of the International Conference on Articial Intelligence and Law*, pages 19–28.

Sunyang Fu, David Chen, Huan He, Sijia Liu, Sungrim Moon, Kevin J Peterson, Feichen Shen, Liwei Wang, Yanshan Wang, Andrew Wen, et al. 2020. Clinical concept extraction: a methodology review. *Journal of biomedical informatics*, 109:103526.

Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. 2009. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer.

Yunfei Ji, Chao Tong, Jun Liang, Xi Yang, Zheng Zhao, and Xu Wang. 2019. A deep learning method for named entity recognition in bidding document. In *Journal of Physics: Conference Series*, volume 1168, page 032076. IOP Publishing.

Larry R Medsker and LC Jain. 2001. Recurrent neural networks. *Design and Applications*, 5(64-67):2.

Ahmad Mehrbod and António Grilo. 2018. Tender calls search using a procurement product named entity recogniser. *Advanced Engineering Informatics*, 36:216–228.

Nikola Milosevic, Cassie Gregson, Robert Hernandez, and Goran Nenadic. 2019. A framework for information extraction from tables in biomedical literature. *International Journal on Document Analysis and Recognition (IJDAR)*, 22:55–78.

Rosmayati Mohemad, Abdul Razak Hamdan, Zulaiha Ali Othman, and Noor Maizura Mohamad Noor. 2011. Ontological-based information extraction of construction tender documents. In *Advances in Intelligent Web Mastering–3: Proceedings of the 7th Atlantic Web Intelligence Conference, AWIC 2011, Fribourg, Switzerland, January, 2011*, pages 153–162. Springer.

PDFMinersix. 2024. Pdfminersix. https://pdfminersix.readthedocs.io/en/latest/. [Online; accessed 02-June-2024].

Lucia Siciliani, Pierluigi Cassotti, Pierpaolo Basile, Marco de Gemmis, Pasquale Lops, Giovanni Semeraro, and Aldo Moro. 2021. Extracting relations from italian wikipedia using self-training. In *CLiC-it*.

Lucia Siciliani, Eleonora Ghizzota, Pierpaolo Basile, and Pasquale Lops. 2023a. Oie4pa: open information extraction for the public administration. *Journal of Intelligent Information Systems*, pages 1–22.

Lucia Siciliani, Vincenzo Taccardi, Pierpaolo Basile, Marco Di Ciano, and Pasquale Lops. 2023b. Ai-based decision support system for public procurement. *Information Systems*, 119:102284.

Esa Toikka et al. 2021. Information extraction from procurement contracts. Master's thesis.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45.

Stephen Wu, Kirk Roberts, Surabhi Datta, Jingcheng Du, Zongcheng Ji, Yuqi Si, Sarvesh Soni, Qiong Wang, Qiang Wei, Yang Xiang, et al. 2020. Deep learning in clinical natural language processing: a methodical review. *Journal of the American Medical Informatics Association*, 27(3):457–470.