**Automated Word Formatter**

Krishna Kalyan N 1, a), Abhinav Aka 2, b), Samuel Sajit Raya 3, c),and*Dr* M Kiran Kumar 4, d)

*1,2,3 UG Student, Department of AIMLE, GRIET, Hyderabad, Telangana, India.*

*4**Associate**Professor, Department of AIMLE, GRIET, Hyderabad, Telangana, India*

a) Corresponding authors: nyshadhamkrishnakalyan2@gmail.com

b) abhinav7835@gmail.com

c) rayasamuel0@gmail.com

d) kirankumar1610@grietcollege.com

**ABSTRACT.** Researchers often struggle when documenting their manuscripts. Without proper formatting, a research paper might face rejection, even with good content. The goal of our project is to develop an app that simplifies the task of converting documents into different academic or professional styles, ensuring consistency and accuracy. Our work serves a wide range of needs from research papers to theses. It smoothly handles titles, headings, text, tables, figures, equations, and presents content in an organized and polished way. Moreover, it automates citation and reference management, greatly improving document preparation efficiency across various formatting guidelines.

**Keywords**: Word Formatter, Layout, Text Segmentation, Document Image Extraction, DOCS, Tabular Data Extraction.

**INTRODUCTION**

In today's academic and professional world, getting your documents formatted right is super important for success. if you're a student, researcher, or pro, you got to follow some specific rules, like those from fancy organizations such as the Electronic Entertainment Experience (E3S). but nailing that perfect format can be tough and eat up a lot of time, causing stress and maybe mistakes to tackle this issue, check out the Automated Word Formatter! it's a cool online tool made to help you easily format your docs correctly. basically, the Automated Word Formatter makes it easier for you to follow research paper guidelines, like the ones from E3S. by giving users a smooth and effective solution, this platform lets folks focus on their work's core ideas instead of stressing over formatting details.

This software works on a straightforward principle: a user uploads a word document, and the Automated Word Formatter does the rest. Through advanced algorithms and predefined formatting styles, the system refines the document layout automatically, ensuring it meets the high standards expected in the academic world. Handling fonts, margins, headings, and references proficiently, the Automated Word Formatter excels at creating professional and standardized documents. The automated tool caters to various requirements and guidelines, focusing primarily on the rigorous standards set by E3S. This guarantees that your work aligns with the expectations of esteemed institutions and publications. Adhering to these standards is crucial not only for readability but also for establishing credibility and trust in your research. Moreover, supported by a robust back-end system, the Automated Word Formatter processes your documents securely. Your data is handled with utmost care and confidentiality, providing you with peace of mind for your sensitive research work. In essence, this tool is your ultimate companion in document formatting; making a previously arduous and time-consuming task simpler.

**Literature Survey**

Nail Nasyrov & his team innovatively tackle document formatting verification. Their method [1] utilizes a decision tree algorithm, maximizing CatBoost's efficiency. This algorithm precisely categorizes document elements as either correct or incorrect based on various attributes like font size, alignment, and line spacing. This systematic approach improves the accuracy of document formatting verification, demonstrating the team's dedication to advancing the field.

[2] Dr. Venkatesan and the team bring forth a novel template matching technique meant for comparing document layouts. This method efficiently spots disparities between documents and then aligns them with the original one. The Python docx module is used for manipulating documents, the difflib library compares documents to find differences, and regex is applied to detect and handle specific patterns. Dr. Venkatesan V and the team demonstrate a methodical approach to comparing and adjusting document layouts, showcasing their combined expertise in analyzing and manipulating documents.

In his work [3], Kirill Chuvilin presents a novel method for processing LaTeX documents, using parse trees to depict document structures. By training a machine learning algorithm on a meticulous dataset of LaTeX documents corrected by experts to include errors like typos, grammatical issues, and formatting errors, the algorithm can effectively pinpoint and fix these errors based on the parse tree representation. This systematic approach driven by machine learning enhances the accuracy and quality of LaTeX documents. Such work makes a substantial contribution to document processing and error correction in the LaTeX framework.

In [4], Oliveira introduces an innovative method that utilizes algorithms to automate the layout of document pages. The initial algorithm works based on the notion that pre-defined rectangular items should be positioned freely within the document. It computes the smallest enclosing box for each item to simplify their arrangement. The second algorithm tackles the arrangement of irregularly shaped items on pages that are partitioned into columns. This is accomplished by determining the most suitable number of columns for the page layout. Oliveira's method offers a structured and systematic approach to automating document page layout, which contributes significantly to advancements in document design and formatting.

Nathan Hurst presents a comprehensive survey in [5], The research explores various methods for automating page layout. It covers different layout modes such as Coordinate, Flow, Grid, VH-Box, Guillotine, and Box. These modes determine how elements are positioned in documents. Additionally, the study examines techniques like Column-driven layout, Cell-driven layout, and Minimal configurations. These techniques are essential in choosing suitable page layouts to address typography issues on both small and large scales. Findings offer valuable insights into the array of ways to automate page layout. This contributes to a better understanding of document design and formatting overall.

Nasser in [6] Utilizing the Django framework, this technique entails submitting a document as input. The contents of the document get stored in a MySQL database and then processed to produce a new document with precise formatting. Nasser's work is remarkable for its creative utilization of web-based automation, integrating Django and database management to effectively streamline word formatting. Nasser introduces an inventive strategy that exploits the Django framework for automating word formatting duties. By submitting a document as input via an interactive web interface, this process unfolds. The information from the document is subsequently saved in a MySQL database and manipulated to create a new document with accurate formatting. Nasser's efforts are distinguished by the efficient utilization of web-based automation, employing Django for web development and MySQL for database management to effectively streamline the word formatting procedure. Nasser's strategy is executed through the Django framework and MySQL database, enabling users to engage with the system via a user-friendly web interface. Although currently catering templates specifically designed for individual student requirements, this method's integration of web technologies makes it effortless for users to input documents and seamlessly receive formatted outputs. Through the amalgamation of web-based automation and database management, Nasser's approach showcases a fresh method of automating word formatting tasks, ameliorating efficiency and user-friendliness for document processing applications.

Gange demonstrated the optimization of table layouts in [7], Gange's study delves into three distinct strategies for determining the best minimum height layout. These methods encompass the AI-driven A\* algorithm, Constraint Programming Approach 1, and Constraint Programming Approach 2 featuring Lazy Clause Generation. The selection of the most efficient technique hinges on the individual traits of the tables in consideration, leading to the embrace of a merged CP/SAT strategy. Gange's work not only furnishes valuable perspectives but also a varied array of tactics for refining table arrangement. By putting forth three diverse methodologies for ascertaining the optimal minimum height layout for tables, Gange sheds light on enhancing table formatting efficiently. These strategies comprise the AI-based A\* algorithm, Constraint Programming Approach 1, and Constraint Programming Approach 2 with Lazy Clause Generation. The decision on which method proves most effective relies on the distinct attributes of the tables under scrutiny, prompting the adoption of a blended Constraint Programming (CP) and Satisfiability (SAT) approach for impeccable table organization. Gange's study offers invaluable viewpoints concerning an assortment of methodologies aimed at ameliorating table layout optimization. In Gange's exploration, both the A\* algorithm and Constraint Programming approaches are employed to refine table layouts, concentrating on reducing table structure height substantially. While these methodologies provide effective resolutions, certain limitations are acknowledged in the paper such as neglecting nested tables - those tables incorporated within other tables resulting in intricate arrangements. Additionally, Gange's strategy introduces a heuristic founded on cell content area, enhancing precision compared to preceding heuristics applied in similar scenarios. Overall, Gange's research serves to propel advancements in table formatting techniques and presents pragmatic methodologies for streamlining table layouts across various document processing applications.

Bilauca proposes distinct approaches in [8] Utilizing mixed-integer programming (MIP) and constraint programming (CP) offers a robust solution. The MIP model, although versatile and powerful, can present computational hurdles with extensive datasets. On the other hand, the CP model, despite its limited scope, often excels in efficiently tackling practical table layout challenges.

In [9], John Bateman delves into Rhetorical Structure Theory (RST), a framework that aims to explain how text is organized and the connections between its parts. RST works on the idea that texts consist of two main units: elementary discourse units (EDUs) and rhetorical relations. EDUs are the basic building blocks within RST, helping us grasp how texts are structured. At the same, the theory looks at defining rhetorical relations, which describe the complex connections between these EDUs. By clarifying how different parts of a text interact, Bateman's research adds significantly to our knowledge of how textual content fits together within the Rhetorical Structure Theory framework.

In [10], Bolshakov presents a novel method using a mathematical cohesion comparator function to assess each line in a document. The resulting cohesion scores are compared systematically, determining paragraph segmentation based on a predetermined threshold. Paragraph splitting depends on evaluating the difference in cohesion scores between two lines against the set threshold This threshold is carefully calculated through a thorough analysis of a database containing collocations and semantic rules. Bolshakov's methodology, based on mathematical cohesion analysis and semantic rules, provides an analytical and data-driven approach to improve paragraph segmentation, contributing to the advancement of document organization and readability.

TableNet, developed by Shubham Singh [11], A new deep learning model is being introduced for advanced table detection & structured data extraction from scanned documents. This innovative model utilizes a two-stream encoder-decoder design to extract features from input images and generate essential output masks for table regions and individual columns. TableNet tackles the challenges of unstructured document images with tables by not only detecting tables but also extracting information from rows and columns within those tables. The model's comprehensive approach prioritizes accurate table identification and precise structure recognition to meet the growing need for efficient information extraction from various documents scanned through mobile devices and scanners.

In [12], Maud Ehrmann delves into the complex task of identifying and categorizing named entities in historical records. This challenge is made even more difficult due to the nature of documents, varying quality, and limited labeled data availability. Strategies for addressing these obstacles range from rule-based methods to machine learning techniques, including recent advancements in deep learning and specialized systems tailored for specific domains. The digital transformation of historical records has revolutionized accessibility while bringing about new hurdles in effectively extracting information from this extensive archive. Widiastuti emphasizes the importance of developing tools to search, retrieve, and analyze data within this vast reservoir of historical knowledge. The emphasis on named entity recognition (NER) systems in the study corresponds with the needs of scholars in the humanities, underscoring the difficulties posed by a wide range of historical documents with varying degrees of accuracy. The research not only catalogs existing tools and reviews past strategies but also sets forth critical priorities for future innovations in this essential field of historical document interpretation.

In [13], Stefan Schweter introduces FLERT, a pioneering method that enriches Named Entity Recognition (NER) models by incorporating document-level features through a Convolutional Neural Network (CNN). FLERT systematically extracts two types of document-level features: global features, encapsulating the overall characteristics of the document, and local features, capturing relationships between named entities within the document. These features are seamlessly integrated into a standard NER model, contributing to enhanced performance. In the broader context of NER approaches traditionally focusing on sentence-level information, FLERT distinguishes itself by leveraging transformer-based models to naturally capture document-level features. The paper conducts a comparative evaluation within the standard NER architectures of "fine-tuning" and "feature-based LSTM-CRF," exploring different hyperparameters for document-level features. Valuable insights from experiments lead to recommendations on effectively modeling document context. The approach is integrated into the Flair framework for reproducibility, presenting new state-of-the-art scores on several CoNLL-03 benchmark datasets and reaffirming the effectiveness of FLERT in advancing document-level feature incorporation for improved NER performance.

In [14], Ming Zhou and their team introduce LayoutLM, a pioneering pre-training framework for Document Image Understanding (DIU) that adeptly predicts both text and layout information in scanned document images. Through extensive pre-training on a large-scale dataset encompassing various tasks such as text recognition, layout analysis, and mask prediction, LayoutLM achieves a holistic understanding of scanned documents.

In reference [15], Widiastuti offers a modular Document Image Extraction System crafted to transform document images into editable text, facilitating effective storage and retrieval of textual content. This system proves especially beneficial for converting paper documents into digital formats, enhancing accessibility to their contents. The primary advancement of this system resides in its modular structure, which combines different text extraction methods to tackle the inherent challenges of processing document images. The flexibility and adaptability of the modular design enable the system to integrate a variety of techniques tailored to specific document attributes and quality.

**Table 1** *Summary of The Existing Approaches*

|  |  |  |  |
| --- | --- | --- | --- |
| **Ref. No** | **Methodology** | **Drawbacks** | **Advantages** |
| [1] | This technique classifies document elements using a decision tree algorithm, which then applies a specified style to the document. | It only checks the errors in documents, but it doesn’t edit the document automatically.​ | Utilizes machine learning algorithms, specifically gradient boosting on decision trees improving performance. |
| [2] | This method utilizes Python's docx, regex, and difflib libraries to classify each element as correct or incorrect. | Lack of template customization options and does not predict mathematical equations. | Provides user friendly  experience and reducing the risk of document rejection and improving overall document quality. |
| [3] | This method uses groups with a tree pattern and group rules to identify potential errors.​ | Can wrongly identify a correct statement to avoid such errors a large number of computational resources are required | It uses Zhang-Shasha algorithm to Correct a wide range of errors including syntax errors, style errors, and formatting errors. |
| [4] | The first approach uses a bounding box method.​  The second approach is a places item in free form​ | The algorithms are highly dependent on the input quality of the document.​The approach doesn’t provide good results on tables and image data. | Recursively divides the document into pages. Places the document items on the pages according to a set of heuristics there by pruning entire traversal |
| [5] | This survey explains about techniques such as Column-driven layout, Cell-driven layout selection methods | It does not use any machine learning approach to automate document formatting it uses the tree structure of the document to edit formatting | It provides accurate formatting as this approach modifies the tree structure of the document. |
| [6] | Implemented using Django framework and MySQL Database. | It only provides template for a specific student template. | Interactive web interface making it convenient for users. |
| [7] | It uses A\* algorithm approach ​  Constraint Programming Approach 1​  Constraint Programming Approach 2 with Lazy Clause Generation | Drawback of the paper is that it does not address the problem of nested tables. Nested tables are tables that are embedded inside other tables which can be used to create complex layouts | This heuristic used in this approach is based on the area of the cell content and is more accurate than the previously used heuristics. |
| [8] | It used two different approaches: MIP (Mixed Integer Programming) and CP (Constraint Programming) | Automatic table layout algorithms may occasionally produce tables that lack accuracy or consistency, particularly when dealing with complex layouts or large amounts of data. | It generates tables in a variety of different formats, such as HTML, PDF, and CSV. |
| [9] | It uses NLP techniques such as chunking parts of speech tagging and semantic role labelling and statistical model Hidden Markov Models (HMM's) | The system is not able to generate diagrams that are interactive and dynamic. | Generates a variety of different types of output including news articles, scientific papers, and technical documentation. |
| [10] | It is achieved in following steps: Quantitative evaluation of text cohesion, Smoothing and Normalizing the cohesion function, Splitting text into paragraphs. | Precision value is low as compared to other experts.​ | The cohesion function is constructed basing on close co-occurring at words pairs contained in a large database collection |
| [11] | TableNet is a two-stream encoder-decoder architecture designed to extract features from an input image, producing two output masks: one for the table region and another for the column region. | TableNet may struggle to accurately detect and extract data from complex tables, especially those with nested headers, merged cells, or irregular structures. | TableNet, being a deep learning model, can learn complex patterns from data. This capability makes it more robust to noise and variations in the appearance of tables in scanned document images. |
| [12] | NERC in historical documents uses both rule-based and machine learning approaches, with recent advances in deep learning and domain-specific systems. | Training NERC models effectively requires a large amount of labeled data, which can be challenging and costly to gather, particularly for historical documents. | NERC models can be trained to achieve high accuracy in identifying and classifying named entities in historical documents. This is important for many applications, such as digital humanities research and historical archiving. |
| [13] | FLERT: Document-Level Features for Named Entity Recognition, FLERT allows NER models to capture document-level information that is not available to traditional NER models.​ | FLERT requires a large amount of labelled data to train effectively. This data can be difficult and expensive to collect, especially for historical documents and low-resource languages. | FLERT has been shown to improve the accuracy of NER models on a variety of datasets. This is because FLERT allows NER models to capture document-level information that is not available to traditional NER models. |
| [14] | LayoutLM is pre-trained on a large-scale dataset of scanned document image.​  Once the model is pre-trained, it can be fine-tuned on a variety of DIU tasks | If the pre-training data is biased, the model will be biased as well. This can lead to errors when the model is used on real-world data. | It is pre-trained on a large-scale dataset of scanned document images. |
| [15] | It Contains 6 components: Document image acquisition, Pre-processing, Text extraction, Feature extraction, Extraction | The system requires a large amount of training data to be trained effectively. This data can be difficult and expensive to collect. | The system is able to extract text from a variety of document types with high accuracy. This is important for applications where the extracted information needs to be reliable. |

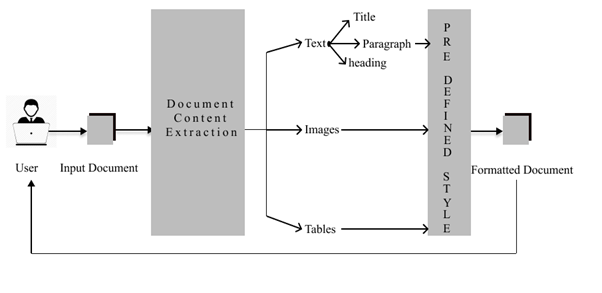
**PROBLEM STATEMENT & OBJECTIVES OF THE PROPOSED WORK**

Within the realm of academia and research, scholars often face challenges when it comes to document formatting. Adhering to specific style guidelines, maintaining uniformity across sections, and navigating the nuances of different formatting styles can be quite a task. The tools currently available do not fully address this ongoing issue. Manual formatting takes up precious time that could be more effectively spent on research and creating content. Inconsistent formatting could compromise the professionalism of academic and research papers, impacting the quality and influence of scholarly work.

It is clear that there is a demand for an efficient, automated tool for formatting documents. This tool should meet the high standards of academic formatting also being easy to use, intuitive, and flexible enough to handle various formatting needs. Meeting this demand will enable researchers, students, and academics to concentrate more on the content of their work rather than being overwhelmed by formatting intricacies. In academia and research, scholars often struggle with document formatting. Following specific style guidelines meticulously, ensuring consistency across different sections, and navigating through various formatting styles can be time-consuming and prone to errors. Existing tools and platforms frequently fail to offer a smooth solution to this problem. Manual formatting procedures consume precious time that could otherwise be spent on research and creating content.

* To Implement a python-based word formatting tool which automates manual editing to save time and effort. ​
* The approach used is to classify the elements of the document this process is carried out by analyzing the xml structure of the document.​
* To implement approaches for formatting text, images, and tables and configuring the type of text and its style by using a XG Boost classifier model.

**PROPOSED METHOD**

**Figure 1** *Architecture Diagram*

As illustrate in figure1, the architectural design is a more detail view of the Automated Word Formatter, it organises a smooth interaction between users and the underlying system. Users initiate the process by uploading Microsoft Word documents through the platform's intuitive interface. These documents find their repository in a NoSQL MongoDB database, as a document is unstructured type of data making it a perfect choice to select a NoSQL database, additionally its adaptability and scalability, ensuring efficient document storage and retrieval are the key features provided by MongoDB. Once the document is retrieved from MongoDB, the system engages in a data extraction process facilitated by the python-docx module. This step involves parsing and capturing both textual content and structural elements embedded within the document. Subsequently, an element identification phase takes place to discern between headings and non-headings, a pivotal aspect for accurate formatting.

Once data is extracted, machine learning steps in using an XG Boost classifier to classify elements such as headers, non-headers, tables, or images. This method allows for precise differentiation of document components, setting the stage for applying specific formatting styles. Each element receives meticulous formatting following established research paper norms. Subsequently, each element's style is analyzed using the regex module to capture its attributes. Once these attributes are obtained, the system automatically adjusts each element's style accordingly. This meticulous process ensures accurate alignment with formatting standards, enhancing the document's appearance. Utilizing regex elevates the formatting procedure, making it adaptable to various document styles and improving overall accuracy.

The properly arranged document is then smoothly merged back into the MongoDB database. This seamless cycle not only guarantees the user's access to the arranged document at their convenience but also paves the way for creating a personalized download link. Offering a download link is a crucial function, directly linking users to their formatted documents stored in the MongoDB database. This linkage serves as a connection, allowing users to easily retrieve their documents with just a simple click. The final outcome is an efficient, user-focused process that aligns with the platform's dedication to providing precise and visually appealing document formatting.

Efficient Document Upload and Storage. The process begins with users initiating document formatting by uploading Microsoft Word files through the platform's intuitive interface. These uploaded documents are securely stored in a NoSQL MongoDB database, chosen for its adaptability and scalability in handling unstructured data like documents. MongoDB efficiently manages document storage and retrieval, ensuring that users can access their documents easily and quickly.

Intelligent Data Extraction and Element Identification

After fetching a document from MongoDB, the system utilizes the python-docx module to extract data. This involves analyzing both text content and structural elements within the document while maintaining layout and formatting. Following this, a phase of identifying elements differentiates between headings and non-headings, a critical step for precise formatting.

Element Classification Driven by Machine Learning

The system employs Machine Learning with an XGBoost classifier for element categorization. This method categorizes document elements as headers, non-headers, tables, or images, allowing for accurate differentiation of document parts. This classification sets the groundwork for applying predefined formatting styles specific to each element.

Adaptive Formatting Using Regex

Each document element is analyzed using the regex module to capture its style attributes. This data is then utilized to automatically adjust the style of each element, ensuring exact modifications to the document's appearance based on established formatting standards. The inclusion of regex enhances the sophistication of the formatting process, making it adaptable to various document styles and improving overall precision.

Smooth Integration and Personalized Document Access

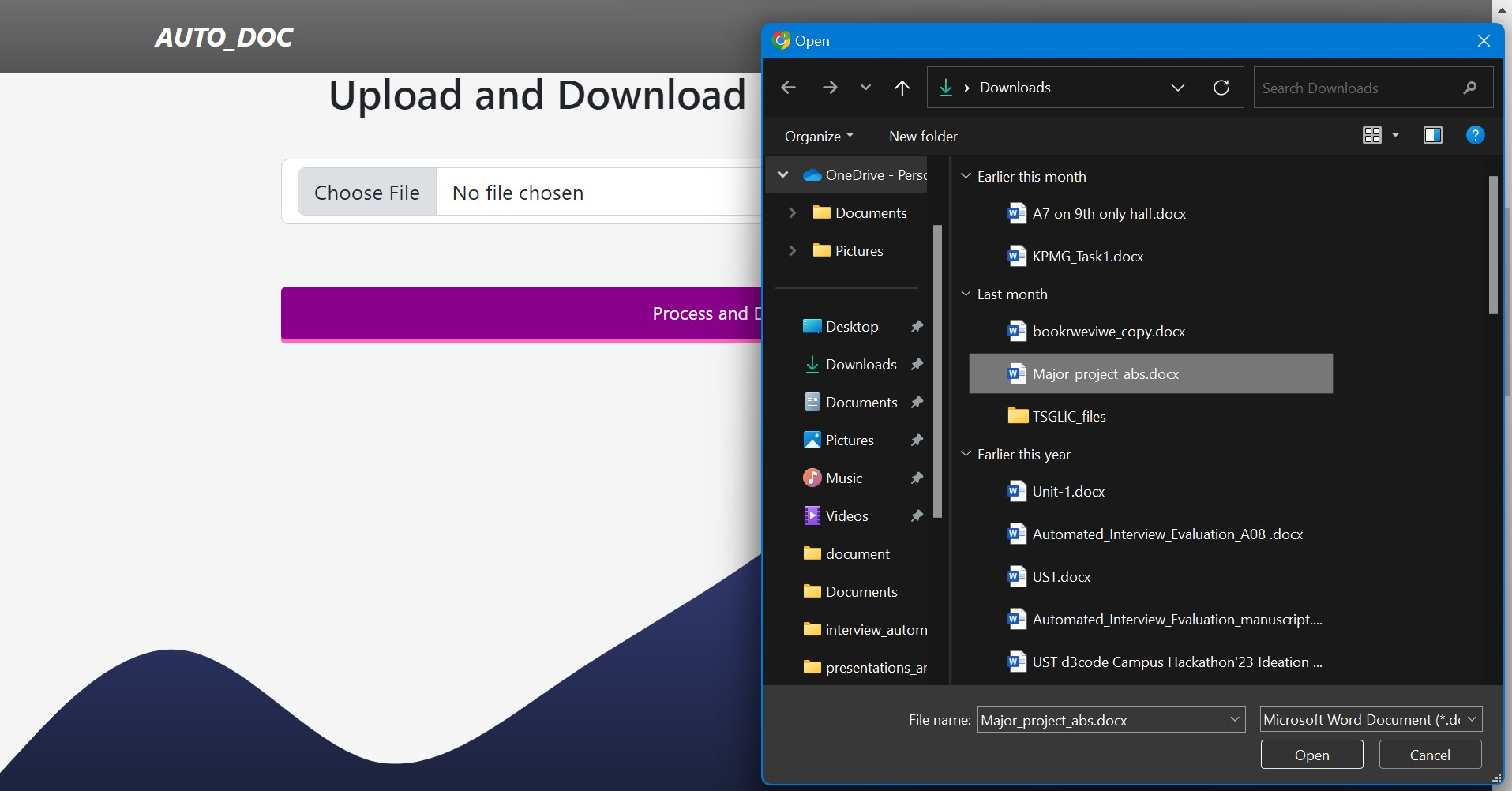
The properly formatted document seamlessly integrates back into the MongoDB database, enabling users to conveniently retrieve their formatted documents. The system creates customized download links that directly link users to their documents stored in MongoDB. This functionality allows for easy document retrieval with a simple click, enhancing user accessibility and experience.

Automated Document Formatting Pipeline The system orchestrates an automated document formatting pipeline that streamlines the entire process from upload to retrieval. This pipeline encompasses sequential stages of document processing, including extraction, identification, classification, formatting, and storage. Each stage is interconnected to ensure a cohesive and efficient workflow for transforming raw document data into formatted outputs.

The system focuses heavily on scalability and optimizing performance to meet user needs and document processing demands effectively. Strategies like load balancing, distributed computing, and caching are in place to improve system responsiveness and scalability. It's built to efficiently handle a large influx of document uploads and formatting tasks while maintaining consistent performance under various workloads. In order to enhance the document formatting experience continuously, the system incorporates user feedback mechanisms and analytics tools. User input on formatting quality and preferences drives iterative improvements in the system's algorithms and user interface. Analytics data offers valuable insights into usage patterns, document processing trends, and performance metrics, guiding data-driven enhancements and optimizations over time. Security is paramount within the architecture to protect user data and documents. Encryption protocols secure data transmission and storage, access control mechanisms manage user permissions, and auditing features monitor system activities. Adhering to data protection regulations guarantees the confidentiality and integrity of user information throughout the entire document formatting process.

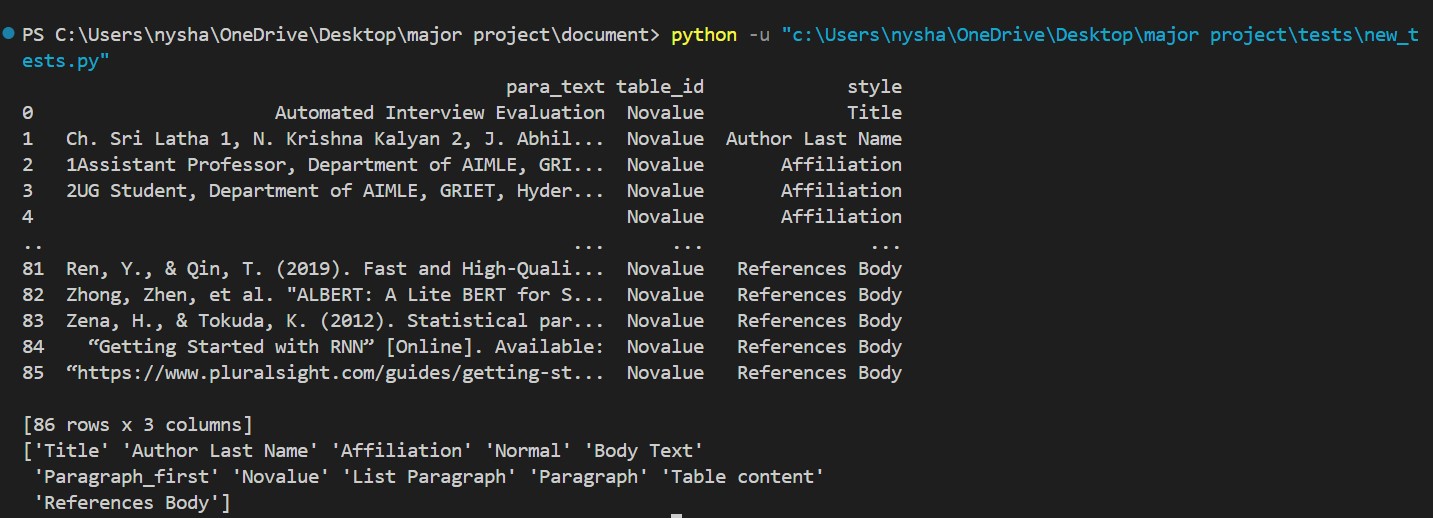
**RESULTS AND DISCUSSIONS**

The first step is to upload a document in our website which can be done by selecting the required file.



**Figure 2** *Uploading Document*

Illustrated in Figure 2, the user engagement process starts with a clear and user-friendly interface. Users begin the document upload process by interacting with the "Choose File" button, which prompts a window to select the desired document from the file manager. After selecting the file, users have the option to reselect the file or proceed by clicking the "Process and Download" button provided on the interface. This straightforward workflow ensures ease of use and efficiency for users interacting with the system.



**Figure 3** *Classifying of Style*

Once the user uploads a document, the system performs automatic classification of document elements based on their styles, distinguishing between images, paragraphs, and tables, and identifying the specific style of each text component. For instance, Figure 3 illustrates how the paragraph text field "Automated Interview Evaluation" was accurately mapped to the style of "Title". This systematic approach ensures that all text contents, along with other elements, are precisely categorized and assigned their respective styles for comprehensive analysis and processing within the document.

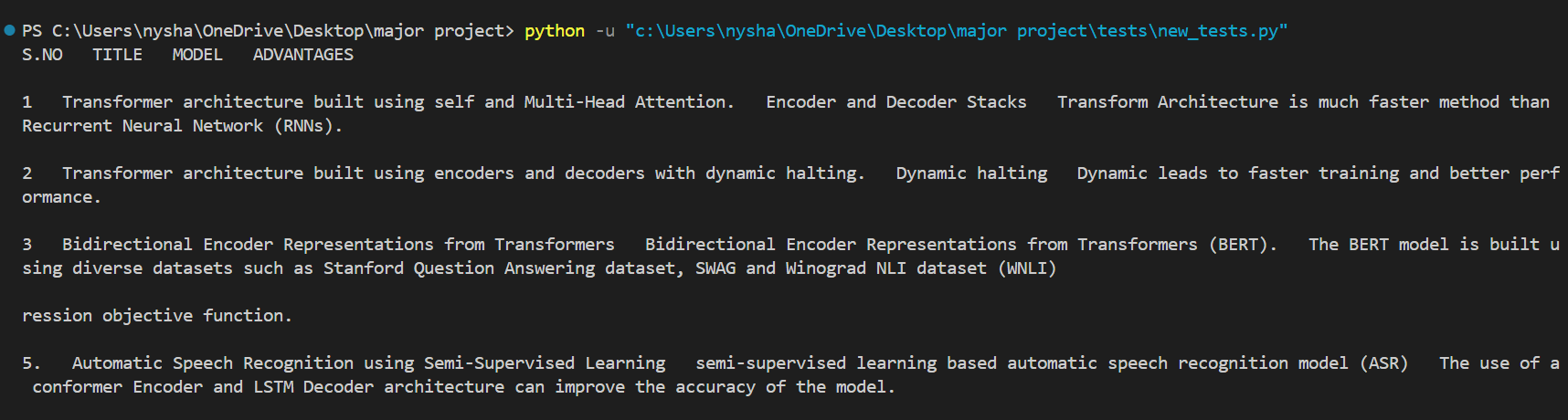
The style is parsed using XML, the paragraph styles undergo re-verification and validation using the XG Boost classification model. This model distinguishes paragraphs as headings or non-headings based on assessments of size, letter count, and word count within the extracted text. Achieving an impressive accuracy of 92% on a randomly generated test dataset, the XG Boost model demonstrates robust performance in accurately classifying paragraph styles, enhancing the reliability and effectiveness of the overall document analysis process.



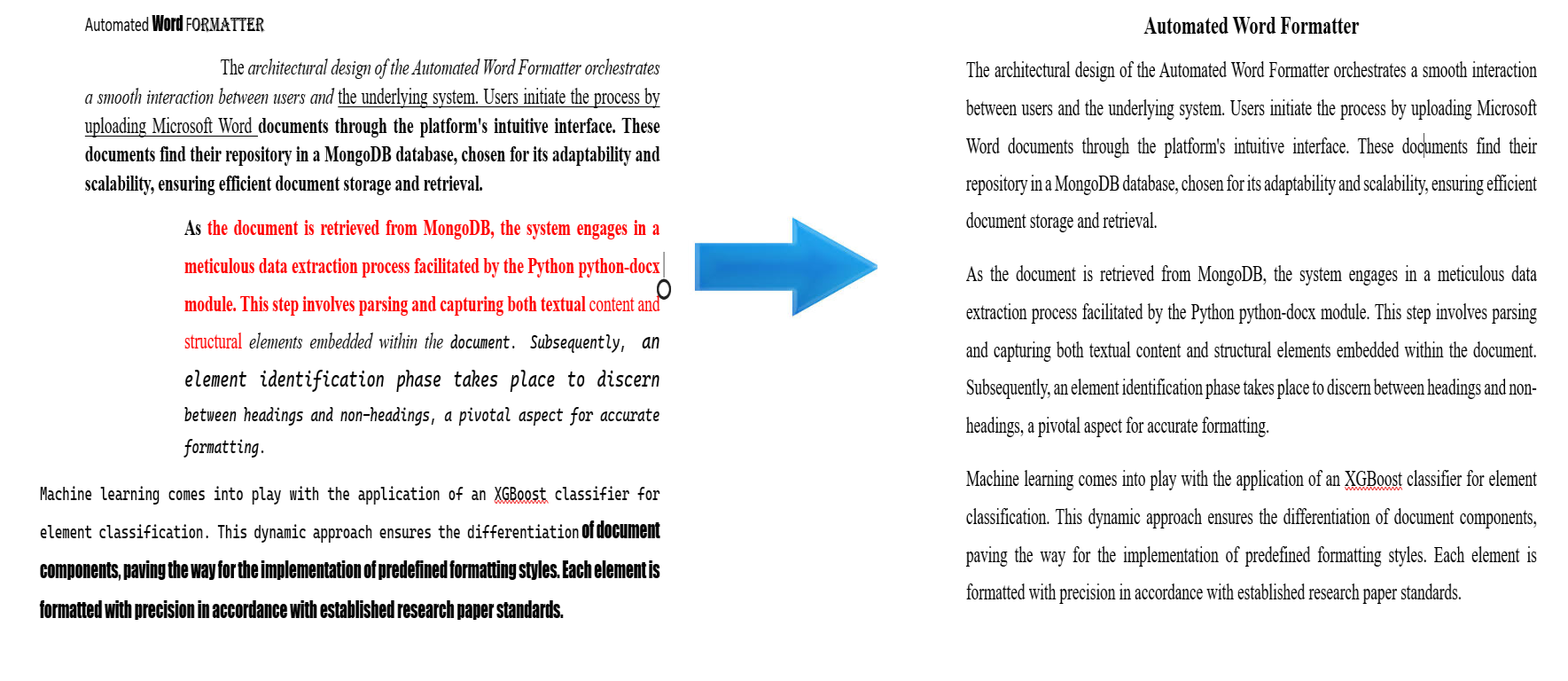
**Figure 4** *Classification Report*

The Figure 4 report shows the performance of a classification model on a binary classification task. The two classes are represented by 1 and 0. The precision, recall, and F1-score are all high for both classes, which means the model is performing well. Precision is a measure of how accurate the model is. A high precision means that most of the time the model predicts class 1, it is actually class 1. In this case, the precision for class 1 is 1.00 and for class 0 it is 0.86.

Recall is a measure of how well the model finds all the actual positives. A high recall means that the model is not missing many actual class 1 examples. In this case, the recall for class 1 is 0.87 and for class 0 it is 1.00.F1-score is a harmonic mean of precision and recall. A high F1-score means that the model is balanced between precision and recall. In this case, the F1-score for class 1 is 0.93 and for class 0 it is 0.92. The accuracy of the model is also high, at 93%. This means that the model correctly classified 93% of the examples in the test set.

**Figure 5** Dealing with Tables

As shown in Figure 5, the process of table extraction involves converting tables into a structured data frame format followed by meticulous formatting based on the style guidelines specified in the research paper. During this formatting phase, careful attention is given to justify each row and column appropriately, ensuring a uniform and visually appealing layout. Specific cell spacing adjustments are also implemented to optimize the table's readability and comprehensibility, facilitating easy interpretation of the data presented within the table for readers and researchers alike. This methodical approach contributes to the overall clarity and effectiveness of the tables within the document analysis workflow.



**Figure 6** *Formatted Document*

Upon successful document upload, the formatting process commences, as visually represented in Figure 6. This pivotal stage involves a systematic approach to document enhancement. The system initiates by collecting essential data from the uploaded document, delving into its textual content and structural elements. Subsequently, an intricate element identification process takes place, distinguishing between headings and non-headings. This critical step lays the foundation for the application of predefined styles tailored to meet the specific requirements of the user. The user-centric approach ensures that the formatting aligns seamlessly with their preferences and conforms to established standards, contributing to a refined and polished document. This structured workflow underscores the system's commitment to delivering accurate and customized formatting, enhancing the overall user experience.

**CONCLUSIONS**

The Automated Word Formatter serves as a revolutionary solution, simplifying document formatting effortlessly. Users can enjoy a smooth formatting experience by simply uploading their documents to the platform. This represents a significant change, turning the once burdensome task of formatting documents into a walk in the park. One of the standout features is the platform's dedication to providing a seamless formatting experience. Users can achieve a polished and professional appearance for their documents without the need for intricate manual adjustments. The formatter streamlines the process, allowing users to focus on the content the document rather than getting bogged down in details. Ensuring consistency in style is a key accomplishment of the Automated Word Formatter. Fonts, alignments, and spacing are all standardized, guaranteeing a cohesive and professional look across documents. This level of consistency is invaluable for users looking to maintain a uniform style in their written work. An important advantage of the formatter is its time-saving capacity. Manually adjusting paragraphs and headings becomes a thing of the past. The platform's efficient process enables users to dedicate their time more effectively to crucial aspects of content creation, confident that formatting challenges are expertly addressed. The adaptability of the formatter shines through, making it a versatile solution for various document types. Whether users are working on a research paper or a report, the platform easily adjusts to different needs, offering a flexible and comprehensive formatting solution. A significant accomplishment of the Automated Word Formatter is its ability to ensure consistency in style across all documents. By standardizing fonts, alignments, and spacing, it creates a cohesive and professional appearance, which is vital for users striving for consistency in their written work. Additionally, the platform saves considerable time by eliminating manual adjustments of paragraphs and headings. This efficiency allows users to allocate their time more wisely to critical aspects of content creation, knowing that formatting complexities are expertly managed.

**FUTURE ENHANCEMENTS**

Supporting Various Document Formats: Enhance compatibility by expanding document format support beyond DOCX to include popular options like PDF, ODT, and RTF. This wide range of formats caters to different user preferences, promoting collaboration and satisfaction. By diversifying format support, your platform can attract a broader audience across various industries and workflows.

Enhancing Collaboration with Multiple Research Formats: Incorporating collaborative editing capabilities along with the ability to work with research formats such as LaTeX, Markdown, BibTeX, and HTML elevates the platform's value for researchers. This integration simplifies collaboration among scholars and streamlines the creation, sharing, and publication of academic content. Researchers can now operate efficiently within familiar formats, boosting productivity and knowledge dissemination.

# **REFERENCES**

|  |  |
| --- | --- |
| 1. | N. Nasyrov, "Automated formatting verification technique of paperwork based on the gradient boosting on decision trees," in *Procedia Computer Science*, 2020. |
| 2. | Dr. Venkatesan, "An Implementation Approach Towards The Automation Of Document Formatting Using Python," *International Journal of Aquatic Science ,* vol. 12, no. 2, 2021. |
| 3. | K. Chuvilin, "Machine learning approach to automated correction of LaTeX documents," in Proceedings of the Conference of FRUCT Association, 2016. |
| 4. | Oliveira, "Two algorithms for automatic document page layout," in Proceedings of the Eighth ACM Symposium, 2008. |
| 5. | N. Hurst, "Review of automatic document formatting," in Proceedings of the 9th ACM Symposium, 2009. |
| 6. | I. Nasser, "Web application for generating a standard coordinated documentation for CS students’ graduation project in Gaza universities," International Journal of Engineering and Information Systems (IJEAIS), vol. 1, no. 6, 2017. |
| 7. | G. Gange, "Optimal automatic table layout," in Proceedings of the 11th ACM Symposium, 2011. |
| 8. | M. Bilauca, "Automatic minimal-height table layout," INFORMS Journal on Computing, vol. 27, 2015. |
| 9. | J. Bateman, "Towards constructive text, diagram, and layout generation for information presentation," Computational Linguistics, vol. 27, 2001. |
| 10. | I. A. Bolshakov, "Text segmentation into paragraphs based on local text cohesion," Springer-Verlag, 2001. |
| 11. | "TableNet: Deep learning model for end-to-end table detection and tabular data extraction from scanned document images," in Proceedings of the International Conference on Document Analysis and Recognition (ICDAR), 2020. |
| 12. | Maud Ehrmann, "Named Entity Recognition and Classification in Historical Documents: A Survey," *ACM Computing Surveys,* vol. 56, no. 2, 2023. |
| 13. | Stefan Schweter, "FLERT: Document-Level Features for Named Entity Recognition," in *Computation and Language*, 2021. |
| 14. | Yiheng Xu, "LayoutLM: Pre-training of Text and Layout for Document Image Understanding," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020. |
| 15. | Widiastuti, "Document image extraction system design," in Proceedings of the 3rd International Conference on Informatics, Engineering, Science, and Technology (INCITEST 2020), 2020. |