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. Researches face a lot of problem while documenting their Manuscripts. In absence of proper formatting, the research paper may get rejected, even if it has quality content. The aim of proposed work is to create an application to streamline the process of transforming documents into various recognized academic or professional styles, ensuring consistency and precision. The proposed application is useful for a wide range of use cases, from Research papers to Academic Thesis, which effortlessly manages titles, headings, text, tables, figures, equations and also presents the content in an organized and professional manner. Additionally, it automates the handling of Citations and References, significantly enhancing the document preparation efficiently across various formatting standards. The proposed application is constructed by modifying the nodes of the document tree structure. With this versatile application, valuable time and effort can be saved while ensuring the documents adhere to the highest standards of formatting.

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

INTRODUCTION

In today's academic and professional world, the correct formatting of documents according to research paper standards is crucial for success. Whether you're a student, researcher, or professional, your work often demands adherence to specific guidelines, such as those set by renowned organizations like the Electronic Entertainment Experience (E3S). Achieving impeccable formatting, however, can be a daunting and time-consuming task, leading to frustration and potential errors. To address this challenge, we present the Automated Word Formatter, a web-based platform designed to simplify the process of correctly formatting your documents. At its core, the Automated Word Formatter streamlines the way you format word documents according to research paper standards, such as those outlined by the E3S. By providing users with a seamless and efficient solution, this platform empowers individuals to focus on the substance of their work rather than getting lost in the intricacies of formatting. The Automated Word Formatter offers an intuitive and user-friendly interface, making it accessible to a wide range of users, from students and researchers to professionals across various fields. The key objective is to eliminate the hassle and complexities associated with formatting, ultimately saving time and reducing the risk of errors.

This platform operates on a simple premise: a user uploads a word document, and the Automated Word Formatter takes care of the rest. By leveraging advanced algorithms and pre-defined formatting styles, the system automatically refines the document's layout, ensuring it adheres to the exacting standards expected in the world of research papers. Whether it's managing fonts, margins, headings, or reference lists, the Automated Word Formatter excels in producing professional and standardized documents. The automated formatting tool caters to a variety of requirements and guidelines, with a primary focus on the stringent standards set forth by the E3S. This ensures that your work aligns with the expectations of prestigious institutions and publications. We understand that conformity to these standards is essential not only for readability but also for fostering credibility and trust in your research. Furthermore, the Automated Word Formatter is underpinned by a robust back-end system that securely processes your documents. Your data is treated with the utmost care and confidentiality, allowing you to trust our platform for your sensitive research work. In a nutshell, the Automated Word Formatter is your ultimate partner in document formatting, simplifying what was once a cumbersome and time-intensive task. It allows you to maintain your focus on the core of your work, all while ensuring that your research paper adheres to the highest standards. Whether you're new to academic writing or a seasoned professional, our platform

welcomes you to experience the future of document formatting. Say goodbye to the stress of document formatting; say hello to the Automated Word Formatter, your gateway to impeccable and effortless research paper formatting.

Literature Survey

Nail Nasyrov and his team have introduced an innovative approach [1] for document formatting verification. In their work, they employ a decision tree algorithm, leveraging the efficiency of CatBoost. This algorithm meticulously classifies each element within documents as either syntactically correct or incorrect. The classification is based on a multitude of attributes, including but not limited to font size, alignment, line spacing, and more. The approach offers a systematic way to enhance the accuracy of document formatting verification, showcasing the team's commitment to advancing the field.

[2] Dr. Venkatesan, along with a collaborative team, introduces a template matching technique designed for comparing the layouts of two documents. This innovative method efficiently identifies differences between documents and subsequently adjusts them to align with the original document. The authors leverage the Python docx module for document manipulation, employ the difflib library to compare documents and pinpoint differences, and use regex to identify and handle specific patterns. This collaborative effort, led by Dr. Venkatesan V, showcases a systematic approach to document layout comparison and adjustment, highlighting the team's collective expertise in document analysis and manipulation.

In his work [3], Kirill Chuvilin introduces an innovative approach to LaTeX document processing, leveraging parse trees to represent document structures. The methodology involves training a machine learning algorithm on a carefully curated dataset of LaTeX documents, manually corrected by experts to include various errors such as typos, grammatical issues, and formatting errors. The algorithm, informed by the parse tree representation, adeptly identifies and rectifies these errors, offering a systematic and machine learning-driven strategy to enhance the precision and quality of LaTeX documents. This work contributes significantly to the field of document processing and error correction within the LaTeX framework.

In [4], Oliveira presents an approach employing two algorithms to automate document page layout. The first algorithm operates under the assumption that previously defined rectangular items are to be placed freely on the document. It calculates the minimum bounding box for each item, facilitating their placement. The second algorithm addresses the placement of free-form items on pages divided into columns. This is achieved by determining the optimal number of columns for the page. Oliveira's approach offers a systematic and algorithmic solution to automate document page layout, contributing to advancements in document design and formatting.

Nathan Hurst presents a comprehensive survey in [5], exploring various existing approaches for automating page layout. The survey delves into different layout modes, including Coordinate, Flow, Grid, VH-Box, Guillotine, and Box, each of which governs how elements are positioned within documents. Additionally, the survey discusses approaches like Column-driven layout, Cell-driven layout, and Minimal configurations, which play crucial roles in selecting appropriate page layouts to tackle issues related to both micro and macro typography. Hurst's work provides valuable insights into the diverse methods available for automating page layout, contributing to the broader understanding of document design and formatting.

Nasser introduces a novel approach in [6] leveraging the Django framework, this method involves submitting a document as input. The document's contents are stored in a MySQL database and later processed to generate a new document with accurate formatting. Nasser's work stands out for its innovative use of web-based automation, incorporating Django and database management to streamline the word formatting process effectively. Nasser presents an innovative approach that leverages the Django framework to automate word formatting tasks. This method involves submitting a document as input through an interactive web interface. The contents of the document are then stored in a MySQL database and processed to generate a new document with accurate formatting. Nasser's work is notable for its efficient use of web-based automation, utilizing Django for web development and MySQL for database management to streamline the word formatting process effectively. Nasser's approach is implemented using the Django framework and MySQL database, allowing users to interact with the system through a user-friendly web interface. While this method currently provides templates tailored for specific student needs, its use of web technologies makes it convenient for users to input documents and receive formatted outputs seamlessly. By integrating web-based automation with database management, Nasser's approach demonstrates a novel way to automate word formatting tasks, enhancing efficiency and usability for document processing applications.

Gange delves into the optimization of table layouts in [7], offering three distinctive approaches for determining the optimal minimum height layout. These methodologies include the AI-based A* algorithm, Constraint Programming Approach 1, and Constraint Programming Approach 2 with Lazy Clause Generation. The choice of the most effective method is contingent upon the unique characteristics of the tables in question, culminating in the adoption of a hybrid CP/SAT approach. Gange's contribution provides valuable insights and a diverse set of methodologies for enhancing table formatting. Gange explores the optimization of table layouts by presenting three distinct methodologies to determine the optimal minimum height layout for tables. These methodologies include the AI-based A* algorithm, Constraint Programming Approach 1, and Constraint

Programming Approach 2 with Lazy Clause Generation. The choice of the most effective method depends on the specific characteristics of the tables being analyzed, leading to the adoption of a hybrid Constraint Programming (CP) and Satisfiability (SAT) approach for optimal table formatting. Gange's contribution offers valuable insights into diverse methodologies aimed at enhancing table layout optimization. Gange's work employs the A* algorithm and Constraint Programming approaches to optimize table layouts, focusing on minimizing the height of table structures. While these methodologies provide effective solutions, the paper acknowledges limitations such as not addressing nested tables, which are tables embedded inside other tables, leading to more complex layouts. Additionally, Gange's approach introduces a heuristic based on the area of cell content, which improves accuracy compared to previous heuristics used in similar contexts. Overall, Gange's research contributes to advancing table formatting techniques and offers practical methodologies for optimizing table layouts in various document processing applications.

Bilauca explores distinct approaches in [8] employing mixed-integer programming (MIP) and constraint programming (CP). The MIP model, while a more general and potent approach, may pose computational challenges, especially for large tables. In contrast, the CP model, while less general, often proves more efficient in solving real-world table layout problems. Through comprehensive evaluations on various real-world tasks, the authors demonstrate the system's capability to generate output comparable to human-generated results. Bilauca's work underscores the effectiveness of combining MIP and CP approaches for achieving optimal table layouts.

In [9], John Bateman provides a comprehensive exploration of Rhetorical Structure Theory (RST), a framework designed to elucidate the organization of text and the relationships existing between its constituent parts. RST operates on the fundamental premise that texts comprise two essential units: elementary discourse units (EDUs) and rhetorical relations. EDUs, representing the smallest analysable units of text within the RST framework, serve as the building blocks for understanding textual structure. Concurrently, the theory focuses on delineating rhetorical relations, which encapsulate the intricate relationships that bind these EDUs. By elucidating the nuanced interplay between text elements, Bateman's work significantly contributes to our understanding of how textual content is organized and interconnected within the framework of Rhetorical Structure Theory.

In [10], Bolshakov introduces an inventive approach that employs a mathematical cohesion comparator function to evaluate each line within a document. The ensuing cohesion scores are systematically compared, and paragraph segmentation is determined based on a precomputed threshold. The decision to split paragraphs hinges on assessing the disparity in cohesion scores between two lines against the established threshold. This threshold is meticulously derived through an in-depth analysis of a database comprising collocations and semantic rules. Bolshakov's methodology, grounded in mathematical cohesion analysis and semantic rules, offers an analytical and data-driven approach to enhance paragraph segmentation, contributing to the evolution of document organization and readability.

TableNet, developed by Shubham Singh [11], introduces an innovative deep learning model tailored for comprehensive table detection and structured data extraction from scanned document images. Utilizing a two-stream encoder-decoder architecture, the model's encoder extracts features from input images, while the decoder generates two vital output masks—one for the table region and another for individual columns within the table. Addressing the challenges posed by unstructured document images containing tables, TableNet goes beyond traditional table detection. It intricately tackles the task of extracting information from rows and columns within the detected tables. The model's holistic approach, emphasizing both accurate table identification and nuanced structure recognition, responds to the increasing demand for efficient information extraction from diverse documents captured through mobile phones and scanners.

In [12], Widiastuti addresses the formidable task of named entity recognition and classification in historical documents, a challenge amplified by document variety, quality variations, and the scarcity of labelled data. Both rule-based and machine learning approaches have been proposed to tackle these challenges, with recent strides in deep learning and domain-specific systems offering promising avenues for advancement. The digitization of historical documents has ushered in a new era of accessibility, but it also presents challenges in efficiently mining information from this vast repository. In the abstract, Widiastuti underscores the significance of developing technologies to search, retrieve, and explore information within this "big data of the past." The focus on named entity recognition (NER) systems in the survey aligns with the demands of humanities scholars, highlighting the challenges posed by diverse, historical, and noisy inputs. The survey not only inventories existing resources and outlines past approaches but also outlines key priorities for future developments in this crucial area of historical document analysis.

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. Unlike previous NLP-focused pre-training models that predominantly emphasize text-level manipulation, LayoutLM uniquely incorporates layout and style information crucial for document image understanding. This innovative approach allows LayoutLM to excel in real-world tasks, including information extraction from scanned documents. Leveraging image features to integrate visual information into the model, LayoutLM represents a breakthrough as the first framework to jointly learn text and layout in a unified manner for document-level pre-training. The versatility of LayoutLM is demonstrated through state-of-the-art results in diverse downstream tasks, such as form understanding, receipt understanding, and document image classification. The code and pre-trained models are made publicly accessible, enhancing accessibility and fostering further advancements in the field.

In reference [15], Widiastuti presents a modular Document Image Extraction System designed to convert document images into editable text, enabling efficient storage and retrieval of textual content. This system is particularly valuable for digitizing paper documents and making their contents accessible in digital formats. The key innovation of this system lies in its modular architecture, which integrates diverse text extraction methods to overcome inherent challenges associated with document image processing. The modular design allows for flexibility and adaptability, enabling the system to incorporate various techniques based on specific document characteristics and quality.

Table 1 *Summary of The Existing Approaches*

Ref. No	Methodology	Drawbacks	Advantages
[1]	It is a classification technique which uses a decision tree algorithm to classify elements of document there by applying the predefined 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 approach is based on the usage of docx, regex and difflib library's present in python which are used to classify every element as either correct or wrong.	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 wi th 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 can sometimes generate tables that are not accurate or consistent. This is especially true for tables with complex layouts or tables that contain a lot 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 that extracts features from the input image and generates two output masks, one for the table region and one for the column region.	TableNet may not be able to accurately detect and extract tabular data from complex tables, such as tables with nested headers, merged cells, or irregular structures.	TableNet is a deep learning model, which means that it is able to learn complex patterns from the data. This 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.	NERC models require a large amount of labelled data to train effectively. This data can be difficult and expensive to collect, especially 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	FLERT requires a large amount of labelled data to	FLERT has been shown to improve the accuracy of

	Recognition, FLERT allows NER models to capture document-level information that is not available to traditional NER models.	train effectively. This data can be difficult and expensive to collect, especially for historical documents and low-resource languages.	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

In the fast-paced world of academia and research, scholars often grapple with the intricacies of document formatting. Whether it's adhering to specific style guidelines, ensuring uniformity across sections, or grappling with the nuances of various formatting styles, the process can be time-consuming and error-prone. The existing tools and platforms fall short in providing a seamless solution to this persistent challenge. Manual formatting consumes valuable time that could be better utilized for research and content creation. Inconsistencies in formatting may lead to a lack of professionalism in academic and research documents, affecting the overall quality and impact of scholarly work.

The need for an efficient, automated document formatting tool is evident. This tool must not only align with the rigorous standards of academic formatting but also be intuitive, user-friendly, and adaptable to diverse formatting requirements. Addressing this need will empower researchers, students, and academics to focus more on the substance of their work rather than getting bogged down by formatting complexities. In the realm of academia and research, scholars face a persistent challenge when it comes to document formatting. The meticulous adherence to specific style guidelines, achieving uniformity across different sections, and navigating through the nuances of diverse formatting styles can be both time-consuming and prone to errors. Current tools and platforms available often fall short of providing a seamless solution to this dilemma. Manual formatting processes consume valuable time that could otherwise be allocated to research and content creation.

- 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

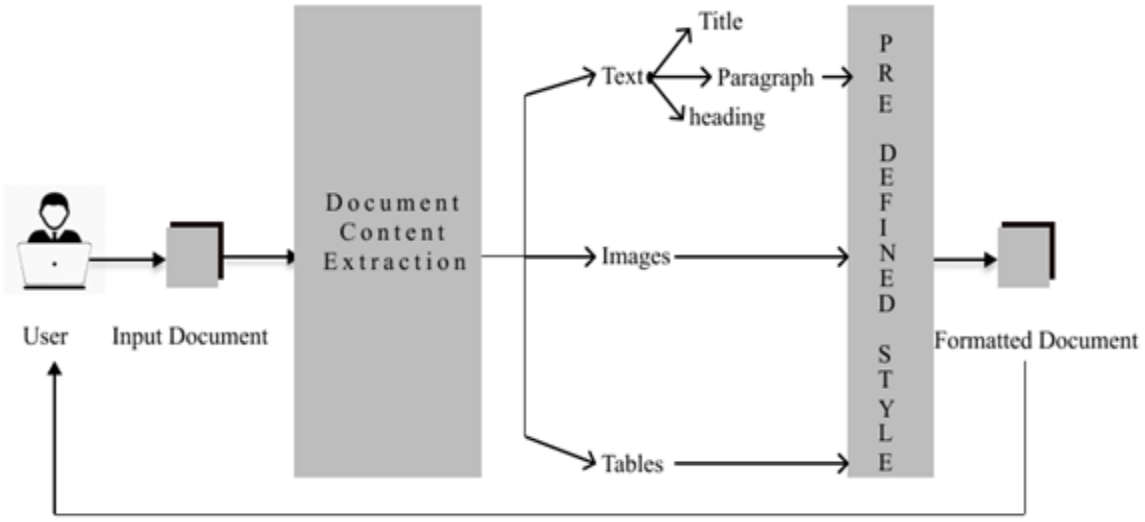

Figure 1 *Architecture Diagram*

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.

Following the data is extracted Machine learning comes into play with the application of an XG Boost classifier for element classification which involves classifying the elements as either headers, non-headers, tables or images. This dynamic approach ensures the differentiation of document components, paving the way for the implementation of predefined formatting styles. Each element is formatted with precision in accordance with established research paper standards. Then each element is now analysed using the regex module to capture its style once these style attributes are captured, the system automatically updates the style of each element accordingly. This ensures a precise adjustment of the document's appearance, aligning it with the predefined formatting standards. The use of regex adds a sophisticated layer to the formatting process, making it adaptive to diverse document styles and enhancing the overall accuracy of the proposed work.

The correctly formatted document is then seamlessly reintegrated into the MongoDB database. This harmonious cycle not only ensures the user's ability to access the formatted document at their convenience but also sets the stage for the generation of a personalized download link. The provision of a download link is a key feature, directly connecting users to their formatted documents stored in the MongoDB database. This link acts as a bridge, enabling users to effortlessly retrieve their documents with a simple click. The end result is an efficient, user-centric process that aligns with the platform's commitment to delivering correct and aesthetically pleasing 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

Once a document is retrieved from MongoDB, the system employs the python-docx module for data extraction. This step involves parsing both textual content and structural elements embedded within the document, preserving its layout and formatting. Subsequently, an element identification phase distinguishes between headings and non-headings, a crucial step for accurate formatting.

Machine Learning-driven Element Classification

The system leverages Machine Learning with an XGBoost classifier for element classification. This dynamic approach categorizes document elements into headers, non-headers, tables, or images, enabling precise differentiation of document components. This classification lays the foundation for applying predefined formatting styles tailored to each element.

Adaptive Formatting with Regex

Each document element undergoes analysis using the regex module to capture its style attributes. This information is then used to automatically update the style of each element, ensuring precise adjustments to the document's appearance according to established formatting standards. The use of regex adds sophistication to the formatting process, making it adaptable to diverse document styles and enhancing overall accuracy.

Seamless Integration and Personalized Document Retrieval

The correctly formatted document seamlessly reintegrates into the MongoDB database, ensuring users can access their formatted documents conveniently. The system generates personalized download links that directly connect users to their documents stored in MongoDB. This feature enables effortless document retrieval with a simple click, enhancing user experience and accessibility.

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.

Scalability and Performance Optimization

The architecture prioritizes scalability and performance optimization to accommodate varying user demands and document processing requirements. Implementations such as load balancing, distributed computing, and caching strategies enhance system responsiveness and scalability. The system is designed to handle large volumes of document uploads and formatting tasks efficiently, ensuring consistent performance under varying workloads.

User Feedback and Analytics Integration

To continuously improve the document formatting experience, the system integrates user feedback mechanisms and analytics tools. User feedback on formatting quality and preferences informs iterative improvements to the system's algorithms and user interface. Analytics data provides insights into usage patterns, document processing trends, and performance metrics, enabling data-driven enhancements and optimizations over time.

Security and Access Control Measures

Security measures are integrated into the architecture to safeguard user data and documents. This includes encryption protocols for data transmission and storage, access control mechanisms to manage user permissions, and auditing features to monitor system activities. Compliance with data protection regulations ensures the confidentiality and integrity of user information throughout the 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.

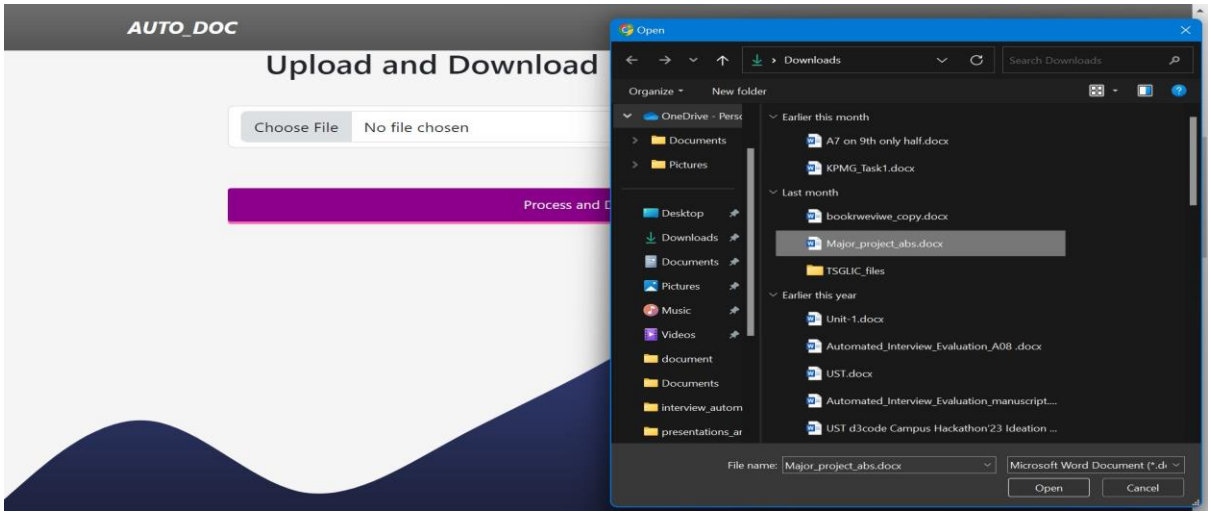

Figure 2 *Uploading Document*

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.

```
PS C:\Users\nysha\OneDrive\Desktop\major project\document> python -u "c:\Users\nysha\OneDrive\Desktop\major project\tests\new_t
ests.py"
                                    para_text table_id            style
0                    Automated Interview Evaluation  Novalue            Title
1   Ch. Sri Latha 1, N. Krishna Kalyan 2, J. Abhil...  Novalue  Author Last Name
2   1Assistant Professor, Department of AIMLE, GRI...  Novalue      Affiliation
3   2UG Student, Department of AIMLE, GRIET, Hyder...  Novalue      Affiliation
4                                                      Novalue      Affiliation
..                                                ...      ...              ...
81  Ren, Y., & Qin, T. (2019). Fast and High-Quali...  Novalue   References Body
82  Zhong, Zhen, et al. "ALBERT: A Lite BERT for S...  Novalue   References Body
83  Zena, H., & Tokuda, K. (2012). Statistical par...  Novalue   References Body
84    “Getting Started with RNN” [Online]. Available:  Novalue   References Body
85  “https://www.pluralsight.com/guides/getting-st...  Novalue   References Body

[86 rows x 3 columns]
['Title' 'Author Last Name' 'Affiliation' 'Normal' 'Body Text'
 'Paragraph_first' 'Novalue' 'List Paragraph' 'Paragraph' 'Table content'
 'References Body']
```

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.

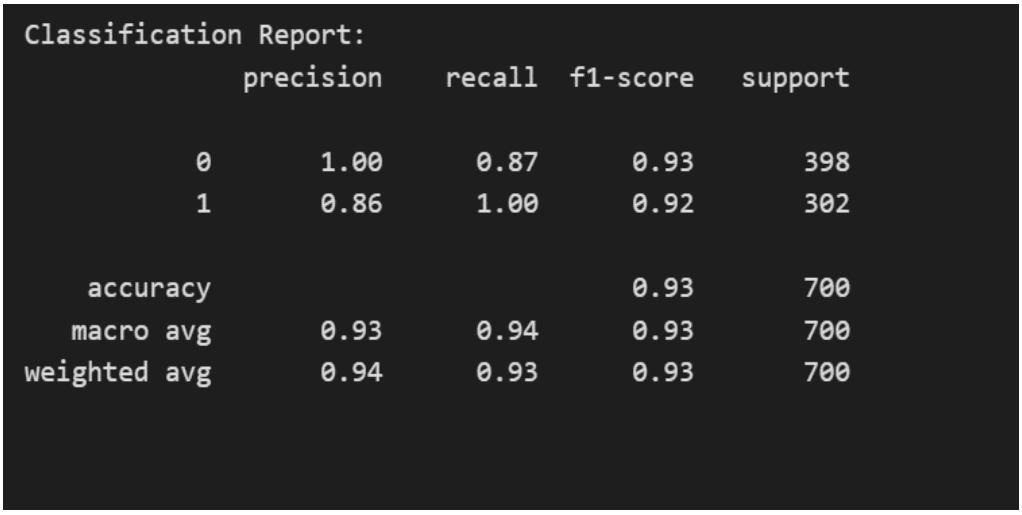

Figure 4 *Classification Report*

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.

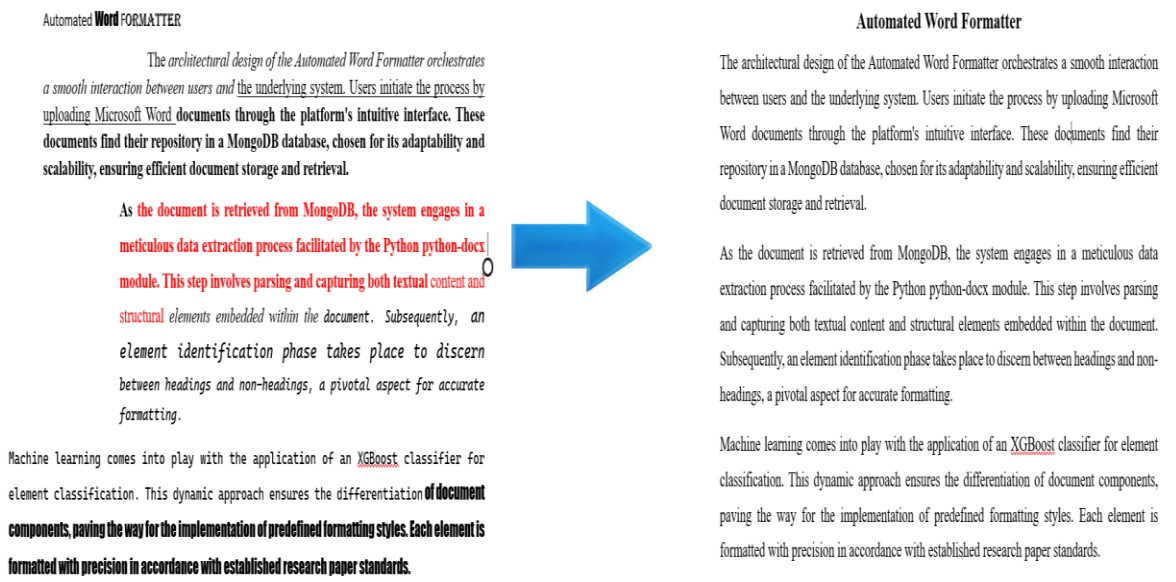

Figure 6 *Formatted Document*

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 stands as a transformative solution, rendering document formatting a hassle-free. Users experience seamless formatting magic by effortlessly uploading their documents to the platform. This marks a significant shift, making the once tedious task of formatting documents a breeze. One of the standout features is the platform's commitment to delivering an effortless formatting experience. Users can achieve a polished and professional look for their documents without the need for intricate manual adjustments. The formatter streamlines the process, enabling users to channel their focus on the document's content rather than getting entangled in formatting intricacies. Consistency in style is a paramount achievement facilitated by the Automated Word Formatter. Fonts, alignments, and spacing are standardized, ensuring a cohesive and professional appearance across documents. This level of consistency is valuable for users aiming to maintain a unified style in their written work. A notable advantage of the formatter is its capacity to save time. The manual adjustment of paragraphs and headings becomes a thing of the past. The platform's streamlined process allows users to efficiently allocate their time to more critical aspects of content creation, knowing that formatting complexities are expertly handled. The formatter's adaptability shines through, making it a universal solution for diverse document types. Whether users are crafting a research paper or report the platform readily adapts to varying needs, providing a versatile and comprehensive formatting solution. A key achievement of the Automated Word Formatter is its ability to ensure consistency in style across documents. Standardizing fonts, alignments, and spacing contributes to a cohesive and professional appearance, which is crucial for users aiming to maintain a unified style in their written work. Moreover, the platform saves significant time by eliminating the need for manual adjustments of paragraphs and headings. This efficiency allows users to allocate their time more effectively to critical aspects of content creation, knowing that formatting complexities are expertly managed.

FUTURE ENHANCEMENTS

Additional Document Formats: Extend support beyond DOCX to include other popular document formats, ensuring a broader user base. Diverse formats like PDF, ODT, and RTF, ensures compatibility with various user preferences. This broad support fosters seamless collaboration and enhances user satisfaction, ultimately expanding your platform's reach and appeal across different industries and workflows.

Collaborative Editing and support for multiple research formats: Implementing collaborative editing features alongside support for multiple research formats such as LaTeX, Markdown, BibTeX, and HTML enriches the platform's utility for researchers. This facilitates seamless collaboration among scholars and streamlines the process of creating, sharing, and publishing academic content. With these capabilities, researchers can work efficiently within familiar formats, fostering productivity and knowledge dissemination.

REFERENCES

1. K. Chuvilin, "Machine learning approach to automated correction of LaTeX documents," in Proceedings of the Conference of FRUCT Association, 2016.

2. Oliveira, "Two algorithms for automatic document page layout," in Proceedings of the Eighth ACM Symposium, 2008.

3. N. Hurst, "Review of automatic document formatting," in Proceedings of the 9th ACM Symposium, 2009.

4. 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.

5. G. Gange, "Optimal automatic table layout," in Proceedings of the 11th ACM Symposium, 2011.

6. M. Bilauca, "Automatic minimal-height table layout," INFORMS Journal on Computing, vol. 27, 2015.

7. J. Bateman, "Towards constructive text, diagram, and layout generation for information presentation," Computational Linguistics, vol. 27, 2001.

8. I. A. Bolshakov, "Text segmentation into paragraphs based on local text cohesion," Springer-Verlag, 2001.

9. Widiastuti, "Document image extraction system design," in Proceedings of the 3rd International Conference on Informatics, Engineering, Science, and Technology (INCITEST 2020), 2020.

10. "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.

11. Maud Ehrmann, "Named Entity Recognition and Classification in Historical Documents: A Survey," *ACM Computing Surveys,* vol. 56, no. 2, 2023.

12. Stefan Schweter, "FLERT: Document-Level Features for Named Entity Recognition," in *Computation and Language*, 2021.

13. 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.

14. N. Nasyrov, "Automated formatting verification technique of paperwork based on the gradient boosting on decision trees," in *Procedia Computer Science*, 2020.

15. Dr. Venkatesan, "An Implementation Approach Towards The Automation Of Document Formatting Using Python," *International Journal of Aquatic Science ,* vol. 12, no. 2, 2021.