Web-Oriented Application for Student Attendance Accounting with a Module for Automatic Parsing of Class Schedules*

Yurii Huk[1,†] , Libor Dostalek[1,†] , Jan Owedyk[3,†] , Hamlet Harutyunyan[4,†] and Andriy Yushko[1,†]

[1] *West Ukrainian National University, 11 Lvivska Str., Ternopil, 46009, Ukraine*
[2] *Department of Computer Systems, Czech Technical University in Prague., Prague, Czech Republic*
[3] *Department of Informatics Kujawy and Pomorze University in Bydgoszcz, Bydgoszcz, Poland*
[4] *Department of Fundamental Disciplines Yerevan Educational and Scientific Institute,, Yerevan, Republic of Armenia*

Abstract

The automation of academic schedule parsing and student attendance tracking is crucial for modern educational institutions. Traditional methods relying on rule-based Excel parsing are prone to errors and lack adaptability to unstructured formats. This study presents a comparative analysis of different methods for parsing academic schedules, focusing on the development of an automated system that utilizes AI-driven approaches for intelligent data extraction. A comparative analysis of these methodologies provides insights into their practical applicability and suggests optimal strategies for integrating AI into education.

Keywords

Artificial intelligence, machine learning, schedule parsing, deep learning, data extraction, natural language processing

1. Introduction

The rapid development of artificial intelligence (AI) and machine learning (ML) has led to their widespread integration into various domains, including education. The integration of artificial intelligence (AI) into educational institutions has become increasingly essential, as it enhances administrative efficiency and minimizes human error in routine processes. One of the most critical aspects of academic management is student attendance tracking, which requires accurate and timely extraction of class schedule data. Traditional methods rely heavily on manual input or rule-based algorithms for parsing structured Excel tables, which often fail when dealing with unstructured or scanned schedule formats. As educational institutions continue to digitize their workflows, there is a growing need for intelligent systems capable of automating schedule parsing and attendance tracking with high accuracy and adaptability.

This study explores three distinct methods for extracting and structuring class schedules: (1) a conventional rule-based Excel parsing approach, (2) a computer vision-driven solution using TensorFlow for image recognition, and (3) an AI-powered natural language processing (NLP) technique that processes textual data. The primary objective is to compare the effectiveness of these approaches in handling different schedule formats, optimizing automation, and reducing the dependency on manual corrections. The research focuses on developing a web-based system that leverages AI to transform schedule data into a structured format suitable for automated attendance tracking.

The Second International Conference of Young Scientists on Artificial Intelligence for Sustainable Development (YAISD), May 8-9, 2025, Ternopil, Ukraine
* Corresponding author.
† These authors contributed equally.
yuriiguk529@gmail.com (Y. Huk); libor.dostalek@fit.cvut.cz (L. Dostalek); j.owedyk@kpsw.edu.pl(Jan Owedyk); h.harutyunyan@wunu.edu.ua (Hamlet Harutyunyan); a.yushko@wunu.edu.ua (A. Yushko)
0009-0001-7483-1726 (Y. Huk); 0000-0002-1613-2644 (L. Dostalek); 0000-0001-6071-3983 (Jan Owedyk); 0000-0002-1676-7949(Hamlet Harutyunyan); 0009-0003-6431-3479 (A. Yushko)

AI-driven solutions offer the potential to significantly improve data processing reliability and efficiency. The application of deep learning models trained on schedule images can enhance text recognition, making it possible to extract meaningful information even from scanned or poorly formatted documents. Additionally, NLP techniques allow for intelligent interpretation of textual data, enabling systems to understand variations in schedule formats and structure them into standardized outputs. This study presents a comparative analysis of these methods, highlighting their strengths, limitations, and practical implications for real-world implementation.

2. Methods of data mining

The process of extracting and structuring academic schedules is a critical component of student attendance tracking systems. Traditional approaches rely on predefined rules for parsing structured Excel tables, which can be effective when dealing with well-formatted data but often fail when schedules are unstructured, scanned, or contain inconsistencies [11]. To overcome these limitations, AI-driven techniques such as computer vision and natural language processing (NLP) offer more adaptable and intelligent solutions, allowing for the automated recognition and interpretation of schedule data in various formats [10].

This section explores three methodologies for academic schedule parsing: direct rule-based extraction from Excel files, a machine learning-based computer vision approach that processes images of schedules using TensorFlow, and an NLP-powered method that structures textual schedule data into a standardized format. Each approach is analyzed in terms of its algorithmic workflow, implementation challenges, and practical applicability in educational settings. The goal is to evaluate their accuracy, flexibility, and efficiency to determine the most effective solution for automated schedule parsing and attendance tracking.

2.1. Direct Rule-Based Extraction from Excel Files

Traditional schedule parsing methods rely on extracting data from structured Excel files using predefined rules. This method assumes that the schedule format remains constant, with data fields occupying fixed positions within the table. Implementation of this method does not require excessive writing of program code listing and voluminous backend architecture. The easiest method provides a common format of tables, their clear structure, which should be followed, namely (Figure 1): clear naming by headers of each column and corresponding values under them in the same format.

```
# Function to convert Excel to JSON
def convert_excel_to_json(excel_file, output_file):

    # Reading an Excel file
    df = pd.read_excel(excel_file)
    # Converting to JSON format
    json_data = df.to_json(orient='records', indent=2, force_ascii=False)

    # Writing data to a file
    with open(output_file, 'w', encoding='utf-8') as f:
        f.write(json_data)

convert_excel_to_json(excel_file, output_file)
```

Listing 1: Simple direct Excel data conversion

This Python script (Listing 1) converts a simple Excel file into a JSON file as follows:

1. Reads the Excel file:
    - the pd.read_excel(excel_file) function loads the Excel file into a Pandas DataFrame (df), automatically interpreting its structure (columns and rows).
2. Converts it to JSON:
    - df.to_json(orient='records', indent=2, force_ascii=False) converts the DataFrame into JSON format.
    - The orient='records' parameter ensures the JSON output is a list of dictionaries (each row becomes a dictionary where column names are keys).

- indent=2 makes the JSON output human-readable.
- force_ascii=False keeps non-ASCII characters (e.g., special or non-English characters).

3. Writes the JSON file:
   - the script opens a file in write mode ('w') with UTF-8 encoding;
   - the JSON data is written to this file.

It relies on the DataFrame structure from the Excel file (Figure 1) to successfully build a structured JSON file: column names become dictionary keys, rows become JSON objects (dictionaries) and the orient='records' parameter ensures each row is converted into a dictionary within a list.

Another typical approach involves using programming languages such as Python to read an Excel file, navigate to specific cells, and extract relevant information based on predefined coordinates. This technique is widely used in simple applications where the schedule format does not change over time.

The process of direct rule-based parsing generally follows these steps:

1. Loading the Excel file – The program opens the file and accesses the specified sheet containing the schedule data;
2. Navigating to fixed cell positions – Since the table structure is static, the program reads data from predefined rows and columns;
3. Extracting relevant information – Course names, dates, times, and classroom assignments are retrieved based on known cell references;
4. Storing and structuring the data – The extracted data is stored in a structured format such as a JSON object or a Python dictionary for further processing;

The following Python script demonstrates how to extract a simple class schedule from an Excel file using openpyxl, a lightweight library for handling Excel files:

```
from openpyxl import load_workbook

# Load the Excel workbook and select the active sheet
wb = load_workbook("schedule.xlsx")
sheet = wb.active

# Define fixed positions for schedule fields (assuming known structure)
schedule_data = []
for row in range(2, sheet.max_row + 1):  # Skipping header row
    group = sheet.cell(row=1).value
    weektype = sheet.cell(row=row, column=1).value
    subject = sheet.cell(row=row, column=2).value
    teacher = sheet.cell(row=row, column=3).value
    day = sheet.cell(row=row, column=4).value
    time = sheet.cell(row=row, column=5).value
    room = sheet.cell(row=row, column=6).value
    zoom = sheet.cell(row=row, column=7).value

    schedule_data.append({
        "Group": group,
        "Weektype": weektype,
        "Subject": subject,
        "Teacher": teacher,
        "Day": day,
        "Time": time,
        "Room": room,
        "Zoom": zoom
    })

# Convert the extracted data into a JSON format
schedule_json = json.dumps(schedule_data, indent=4, ensure_ascii=False)
print(schedule_json)
```

Listing 2: Direct rule-based extraction from Excel file

	A	B	C	D	E	F	G	H	I
1	№	Group	Weektype	Subject	Teacher	Day	Time	Room	Zoom
2	1	CS-31	Odd	Artificial Intelligence	Dr. Smith	Monday	10:00	6401	603 232 4543
3	2	SE-42	Even	Machine Learning	Prof. Jones	Wednesday	14:00	6102	604 232 4543

Figure 1: Example of a table with the class schedule

```
[
  {
    "№": "1",
    "Group": "CS-31",
    "Weektype": "Odd",
    "Subject": "Artificial Intelligence",
    "Teacher": "Dr. Smith",
    "Day": "Monday",
    "Time": "10:00",
    "Room": "6401",
    "Zoom": "603 232 4543"
  },
  {
    "№": "2",
    "Group": "SE-42",
    "Weektype": "Even",
    "Subject": "Machine Learning",
    "Teacher": "Prof. Jones",
    "Day": "Wednesday",
    "Time": "14:00",
    "Room": "6102",
    "Zoom": "604 232 4543"
  }
]
```

Listing 3: The output (result) of the script in JSON file

While this approach is straightforward and computationally efficient, it suffers from major limitations. Any changes to the file's structure, such as column rearrangements or merged cells, can lead to parsing errors. Moreover, this method does not handle unstructured or scanned schedules, making it unsuitable for real-world educational environments where schedules frequently change. Studies on AI-driven scheduling confirm that rigid rule-based approaches lack the flexibility needed for automated schedule extraction across diverse formats [1].

2.2. Computer Vision-Based Recognition Using TensorFlow

Computer Vision encompasses techniques for the automated extraction, interpretation, and analysis of meaningful information from individual images or sequences of images [9]. To overcome the limitations inherent in rule-based extraction methods, computer vision techniques have been employed to interpret schedule data from images or scanned documents. Utilizing TensorFlow, a prominent deep learning framework, models can be trained to recognize patterns and extract pertinent information from visual data [6]. This approach involves creating a dataset of labeled schedule images and training a convolutional neural network (CNN) to identify and extract relevant details, such as course names, times, and locations. The advantage of this method lies in its ability to handle unstructured data and variations in formatting, providing a more flexible solution for schedule parsing. Research has demonstrated that deep learning techniques significantly improve the accuracy of text recognition from complex tabular images, making them highly applicable in educational data processing [2,3]. Furthermore, CNN models trained on large datasets have been proven to outperform traditional optical character recognition (OCR) techniques, particularly when handling varying fonts, alignments, and noise levels in scanned documents [4].

The implementation of a computer vision-based schedule parsing system encompasses several key stages:

1. Dataset Preparation: collect a comprehensive dataset of schedule images, ensuring diversity in formats, fonts, and layouts. Annotate these images to label the regions containing relevant information, such as course names, times, and locations;
2. Image Preprocessing: enhance image readability by applying grayscale conversion [13], contrast adjustments, and noise reduction. Techniques such as adaptive thresholding and edge detection help refine input data for better recognition;
3. Neural Network Design: construct a CNN model that can efficiently process image-based schedule data. The architecture should include multiple convolutional layers to capture spatial features, along with pooling layers to reduce dimensionality [9,12,13];
4. Model Training: use supervised learning techniques to train the CNN model on annotated schedules. The network learns to distinguish patterns and extract relevant text-based information, improving accuracy with each training iteration;
5. Post-Processing: implement post-processing steps to convert the model's output into a structured format, such as JSON. This may involve mapping the detected information to predefined categories and organizing it for further use;

The following Python example demonstrates how to define and train a CNN model using TensorFlow to recognize structured data from schedule images:

```
import tensorflow as tf
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense

# Define the CNN model structure
cnn_model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(128, 128, 1)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Conv2D(128, (3, 3), activation='relu'),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(5, activation='softmax')  # Adjust output neurons based on label categories
])

# Configure the model with a suitable loss function and optimization algorithm
cnn_model.compile(optimizer='adam',
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])

# Train the CNN using preprocessed schedule images and labels
cnn_model.fit(train_data, train_labels, epochs=25, validation_data=(test_data, test_labels))
```

Listing 4: Computer vision-based recognition using TensorFlow snippet

In this example, train_images and train_labels represent the preprocessed training data and corresponding labels, respectively. The CNN consists of convolutional layers interspersed with pooling layers, followed by fully connected layers that output class probabilities [12].

```
Epoch 34/50
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 102ms/step - accuracy: 0.7348 - loss: 0.9082 - val_accuracy: 0.6667 - val_loss: 1.4336
Epoch 35/50
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 94ms/step - accuracy: 0.7348 - loss: 0.8889 - val_accuracy: 0.6667 - val_loss: 1.4331
Epoch 36/50
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 100ms/step - accuracy: 0.6932 - loss: 0.9427 - val_accuracy: 0.6667 - val_loss: 1.4359
Epoch 37/50
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 97ms/step - accuracy: 0.6932 - loss: 0.9304 - val_accuracy: 0.6667 - val_loss: 1.4403
Epoch 38/50
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 97ms/step - accuracy: 0.7348 - loss: 0.8666 - val_accuracy: 0.6667 - val_loss: 1.4435
Epoch 39/50
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 98ms/step - accuracy: 0.6932 - loss: 0.8965 - val_accuracy: 0.6667 - val_loss: 1.4417
Epoch 40/50
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 96ms/step - accuracy: 0.6932 - loss: 0.8828 - val_accuracy: 0.6667 - val_loss: 1.4418
Epoch 41/50
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 96ms/step - accuracy: 0.7765 - loss: 0.8045 - val_accuracy: 0.6667 - val_loss: 1.4401
Epoch 42/50
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 98ms/step - accuracy: 0.7348 - loss: 0.8541 - val_accuracy: 0.6667 - val_loss: 1.4354
Epoch 43/50
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 111ms/step - accuracy: 0.7765 - loss: 0.7653 - val_accuracy: 0.6667 - val_loss: 1.4280
Epoch 44/50
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 106ms/step - accuracy: 0.6932 - loss: 0.8718 - val_accuracy: 0.6667 - val_loss: 1.4183
Epoch 45/50
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 98ms/step - accuracy: 0.7765 - loss: 0.7283 - val_accuracy: 0.6667 - val_loss: 1.4114
```

Figure 2: Model training results

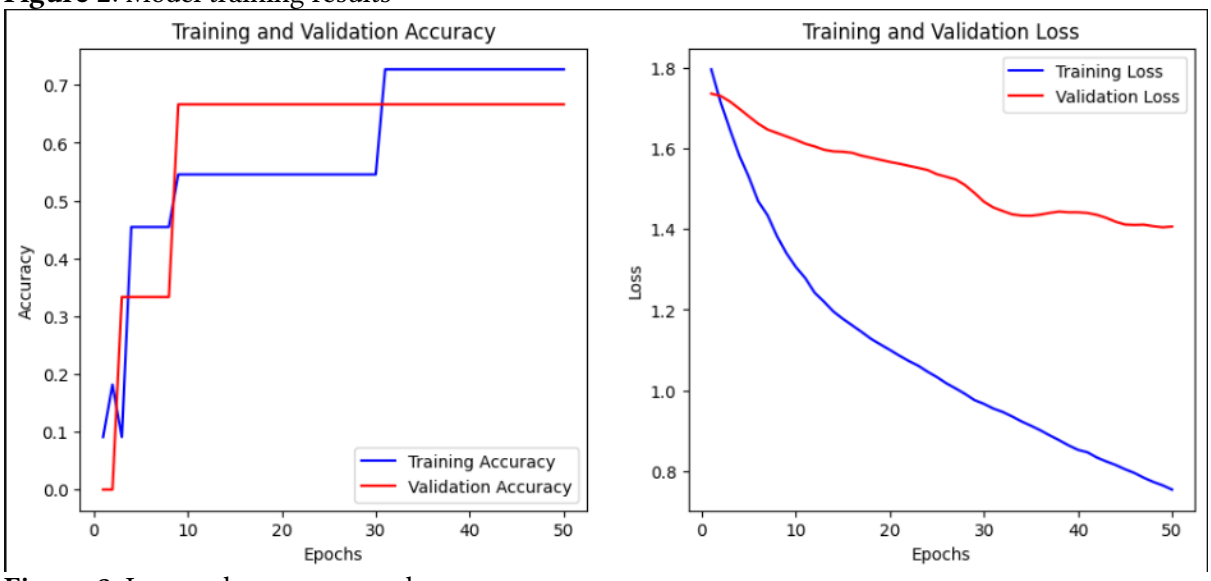

Figure 3: Loss and accuracy graphs

After training the model for data collection, analysis and processing, we obtained the following results (Figure 2,3):

1. On the left graph "Training and validation accuracy" (Figure 3):
   - accuracy steadily increases for both training (blue) and validation (red);
   - however, validation accuracy seems to plateau after around 30 epochs, suggesting that additional training may not significantly improve validation performance;
   - the training accuracy surpasses 0.7, while validation accuracy remains slightly lower;

   This indicates some potential overfitting, but not severe.
2. On the right graph "Training and validation loss" (Figure 3):
   - training loss (blue) consistently decreases, which is expected as the model learns;
   - validation loss (red) decreases initially, but then flattens or slightly increases after a certain point. This suggests the model may start to overfit to the training data rather than generalizing well.

By employing a computer vision-based approach, the system can effectively parse schedules from images with varying formats and structures. The trained CNN can generalize across different layouts, recognizing and extracting relevant information despite variations in design [6,7,9]. This adaptability addresses the rigidity of rule-based methods, offering a robust solution for real-world applications where schedule formats may not be consistent.

Computer vision-based recognition using TensorFlow [6] provides a significant advancement over traditional rule-based methods for schedule parsing. By leveraging the capabilities of CNNs to learn complex patterns in visual data, this approach accommodates unstructured and diverse schedule formats. The flexibility and robustness of deep learning models make them well-suited for educational data processing tasks, enhancing the accuracy and reliability of information extraction from schedule images.

2.3. Natural Language Processing-Driven Text Processing

In the realm of automated schedule parsing, Natural Language Processing (NLP) techniques have emerged as powerful tools for extracting structured information from unstructured textual data [14]. NLP leverages computational methods to process and analyze human language, enabling systems to interpret and organize textual information effectively [8,15]. This approach is particularly beneficial when dealing with schedules presented in free-form text, where traditional rule-based methods may falter due to variability in language and formatting. Studies suggest that NLP-based models outperform rule-based approaches in terms of flexibility and adaptability, as they can generalize across different formatting styles without requiring predefined rules [5]. Additionally, advancements

in NLP models, such as transformers and sequence-to-sequence architectures, have shown promising results in extracting structured data from semi-structured academic documents [3].

The process begins with data collection, where a substantial corpus of textual schedules is gathered to serve as the training dataset. This dataset should encompass a wide variety of schedule formats and linguistic expressions to ensure the model's robustness. Subsequently, data preprocessing is conducted, involving tokenization (dividing text into words or phrases), part-of-speech tagging (identifying grammatical categories) [15], and named entity recognition (detecting entities like dates, times, and course names). These preprocessing steps transform raw text into a structured format suitable for machine learning algorithms.

Following preprocessing, the core of the NLP approach involves training machine learning models to recognize patterns and relationships within the text. Advanced models, such as Transformer-based architectures (e.g., BERT, GPT, and T5) [8], have demonstrated remarkable proficiency in understanding context and semantics in natural language, making them well-suited for this task [6]. These models are trained to identify and extract pertinent information, such as course titles, timings, and locations, from the textual data. The extracted information is then organized into a structured format, such as a JSON object, facilitating seamless integration with other systems and applications.

For instance, consider a segment of text from a schedule: *"The Introduction to Biology class meets every Monday and Wednesday at 09:35 AM in Room 6204 with Prof. Jones only on even week."* An NLP model can process this sentence to extract the course name ("Introduction to Biology"), the days of the week ("Monday and Wednesday"), the time ("09:35 AM"), the location ("Room 6204"), the teacher (“Prof. Jones”) and the type of the week (“even”). The extracted data can then be structured into JSON format as follows in Listing 3.

The following Python code demonstrates how an NLP pipeline can extract structured schedule data using the spaCy library:

```
import spacy
import json

# Load a pre-trained NLP model
nlp = spacy.load("en_core_web_sm")

# Example schedule text
text = " The Introduction to Biology class meets every Monday and Wednesday at 09:35 AM in Room 6204 with Prof.  Jones only on even week."

# Process text
doc = nlp(text)

# Extract relevant entities (simplified approach)
schedule_data = {
   "course_name": "Introduction to Biology",
   "days": ["Monday", "Wednesday"],
   "time": "09:35",
   "location": "6204",
   "teacher": "Prof. Jones",
   "week_type": "even"
}

# Convert to JSON
schedule_json = json.dumps(schedule_data, indent=4)
print(schedule_json)
```

Listing 5: Example Code for an NLP-Based Schedule Parser

The flexibility of NLP-based methods allows for the handling of diverse schedule representations, making them adaptable to various textual formats without the need for rigid predefined rules. This adaptability is particularly advantageous in educational settings where schedule formats may vary significantly across institutions or departments. Moreover, NLP techniques can manage ambiguities and variations in natural language, enhancing the robustness of the schedule parsing system.

The application of NLP techniques in schedule parsing offers a sophisticated and flexible approach to extracting structured information from unstructured textual data [15]. By leveraging advanced machine learning models, NLP enables the development of robust systems capable of adapting to

diverse schedule formats and linguistic variations, thereby enhancing the efficiency and accuracy of automated schedule management. NLP-based approaches outperform traditional rule-based methods in their adaptability and efficiency, making them well-suited for modern AI-driven academic administration systems.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

3. Conclusions

This study explored three distinct methods for parsing academic schedules: rule-based extraction from structured Excel files, deep learning-based computer vision recognition, and Natural Language Processing (NLP)-driven text processing. Each method presents unique advantages and limitations, influencing its suitability for different use cases in automated student attendance tracking.

Rule-based Excel parsing is a straightforward approach that efficiently extracts data from well-structured spreadsheets. It is computationally inexpensive and easy to implement but lacks adaptability when schedule formats change. Even minor deviations, such as merged cells or column reordering, can cause errors, making it unsuitable for handling unstructured data.

The computer vision-based approach using TensorFlow offers greater flexibility by recognizing schedule information directly from images. It enables the extraction of data from scanned documents and printed timetables, making it more robust than rule-based methods. However, it requires substantial computational resources for training convolutional neural networks (CNNs) and a large annotated dataset to ensure high accuracy. Despite these challenges, this method excels in environments where schedules are available only in image form.

The NLP-driven method is the most adaptable, capable of handling unstructured text data and generalizing across different formatting styles. By leveraging transformer-based architectures, NLP models can extract and structure schedule information with high accuracy. This approach is especially beneficial when dealing with schedules in free-text format or inconsistent table structures. However, its effectiveness depends on the availability of extensive training datasets and preprocessing techniques.

Table 1
Comparative Analysis of Schedule Parsing Methods

Method	Execution Time	Data Accuracy	Adaptability	Complexity	Suitability
Rule-Based Excel Parsing	Fast	High (for fixed formats)	Low	Low	Best for structured spreadsheets with a static format
Computer Vision (CNN)	Moderate	High (for printed schedules)	Medium	High	Best for scanned or image-based schedules
NLP-Based Parsing	Moderate to High	Very High	Very High	High	Best for unstructured text-based schedules with variable formatting

The choice of a schedule parsing method depends on the specific requirements of an institution. If schedules are consistently formatted Excel spreadsheets, a rule-based method may suffice. If the schedules are often scanned or photographed, a computer vision-based approach would provide greater flexibility. However, if schedules exist in multiple textual formats with varying structures, NLP-based methods offer the most robust and scalable solution.

Future improvements in AI-driven schedule parsing could involve hybrid models that integrate rule-based approaches for structured data, CNNs for image-based recognition, and NLP for text processing. Such a multimodal system would enhance adaptability, ensuring high accuracy regardless of the input format. Additionally, fine-tuning deep learning models with larger and more diverse datasets will further improve their generalization capabilities and efficiency in real-world educational applications.

By integrating AI-driven solutions, educational institutions can automate schedule management with greater precision, reducing manual workload and improving attendance tracking efficiency. As AI continues to evolve, future systems may incorporate real-time schedule adjustments and predictive analytics to optimize resource allocation and enhance overall academic administration.

References

[1] Vasileiou, V., & Yeoh, W. (2025). AI tool helps make trustworthy, explainable scheduling decisions. Washington University in St. Louis Engineering News. URL: https://engineering.washu.edu/news/2025/AI-tool-helps-make-trustworthy-explainable-scheduling-decisions.html

[2] Hilbert, M. (2021). Machine learning for the educational sciences. *Review of Education*, 9(3), 691-725. URL: https://doi.org/10.1002/rev3.3310

[3] Lin, Y., Chen, H., Xia, W., Lin, F., Wang, Z., & Liu, Y. (2023). A comprehensive survey on deep learning techniques in educational data mining. *arXiv preprint arXiv:2309.04761*. URL: https://arxiv.org/abs/2309.04761

[4] Mahakud, B., Parida, B., Panda, I., Maity, S., Sahoo, A., & Sharma, R. (2022). A machine learning system to monitor student progress in educational institutes. *arXiv preprint arXiv:2211.05829*. URL: https://arxiv.org/abs/2211.05829

[5] Virtosoftware. (2024). AI tools for school schedules and timetables: Prompts & guide. *Virtosoftware Blog*. URL: https://blog.virtosoftware.com/ai-schedule-maker-for-schools/

[6] TensorFlow Core. Convolutional Neural Network (CNN). URL: https://www.tensorflow.org/tutorials/images/cnn

[7] Sebastian Schreiber, Stefan Agne, Ivo Wolf, Andreas Dengel, and Sheraz Ahmed. DeepDeSRT: Deep Learning for Detection and Structure Recognition of Tables in Document Images. In Proceedings of the International Conference on Document Analysis and Recognition (ICDAR), 2017.

[8] Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*. URL: https://arxiv.org/abs/1810.04805

[9] Nayeem, T. A., Motaharuzzaman, S. M., Hoque, A. T., & Rahman, M. H. (2022, December). Computer vision based object detection and recognition system for image searching. In *2022 12th International Conference on Electrical and Computer Engineering (ICECE)* (pp. 148-151). IEEE.

[10] Medium. AI Techniques for Data Parsing and Structuring. URL: https://medium.com/isomeric/ai-techniques-for-data-parsing-and-structuring-4345c0456032

[11] Medium. Comparing 6 Frameworks for Rule-based PDF parsing. URL: https://levelup.gitconnected.com/comparing-6-frameworks-for-rule-based-pdf-parsing-f9e7ca5b6cc9

[12] Zhao, Wenzhi & Du, Shihong. (2016). Spectral–Spatial Feature Extraction for Hyperspectral Image Classification: A Dimension Reduction and Deep Learning Approach. IEEE Transactions on Geoscience and Remote Sensing. 54. 4544-4554. 10.1109/TGRS.2016.2543748.

[13] Medium. Convolutional Neural Networks. URL: https://medium.com/@erdematbas/convolutional-neural-networks-ff2070fe185d

[14] Docsumo. Harnessing Natural Language Processing (NLP) for Information Extraction. URL: https://www.docsumo.com/blog/nlp-information-extraction

[15] Khurana, D., Koli, A., Khatter, K., & Singh, S. (2023). Natural language processing: state of the art, current trends and challenges. *Multimedia tools and applications*, *82*(3), 3713-3744.

[16] Dyvak, Mykola, Oleksandr Papa, Andrii Melnyk, Andriy Pukas, Nataliya Porplytsya, and Artur Rot. 2020. "Interval Model of the Efficiency of the Functioning of Information Web Resources for Services on Ecological Expertise" Mathematics 8, no. 12: 2116. https://doi.org/10.3390/math8122116

[17] A. Kovbasistyi, A. Melnyk, M. Dyvak, V. Brych and I. Spivak, "Method for detection of non-relevant and wrong information based on content analysis of web resources," 2017 XIIIth International Conference on Perspective Technologies and Methods in MEMS Design (MEMSTECH), Lviv, Ukraine, 2017, pp. 154-156, doi: 10.1109/MEMSTECH.2017.7937555.

[18] M. Dyvak, A. Melnyk, A. Kovbasistyi, R. Shevchuk, O. Huhul and V. Tymchyshyn, "Mathematical Modeling of the Estimation Process of Functioning Efficiency Level of Information Web-Resources," 2020 10th International Conference on Advanced Computer Information Technologies (ACIT), Deggendorf, Germany, 2020, pp. 492-496, doi: 10.1109/ACIT49673.2020.9208846.

[19] M. Dyvak, A. Kovbasistyi, A. Melnyk, I. Shcherbiak and O. Huhul, "Recognition of Relevance of Web Resource Content Based on Analysis of Semantic Components," 2019 9th International Conference on Advanced Computer Information Technologies (ACIT), Ceske Budejovice, Czech Republic, 2019, pp. 297-302, doi: 10.1109/ACITT.2019.8779897.