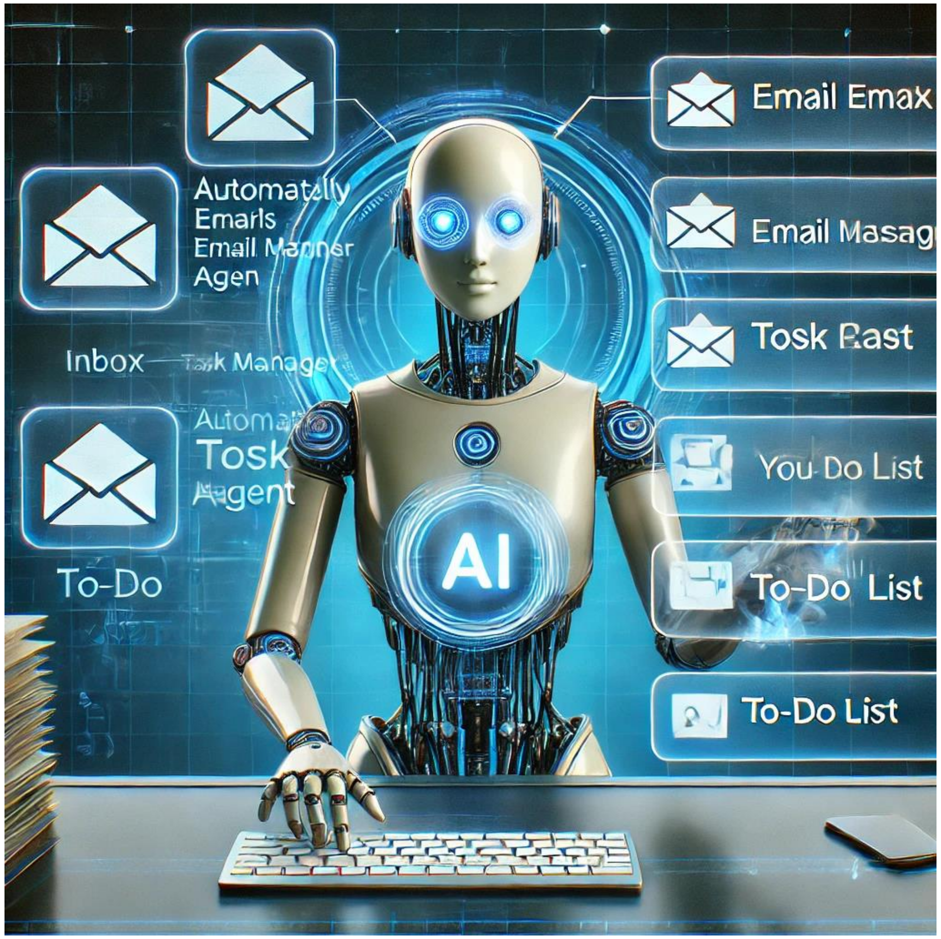
Email Task Manager Agent



Problem Statement

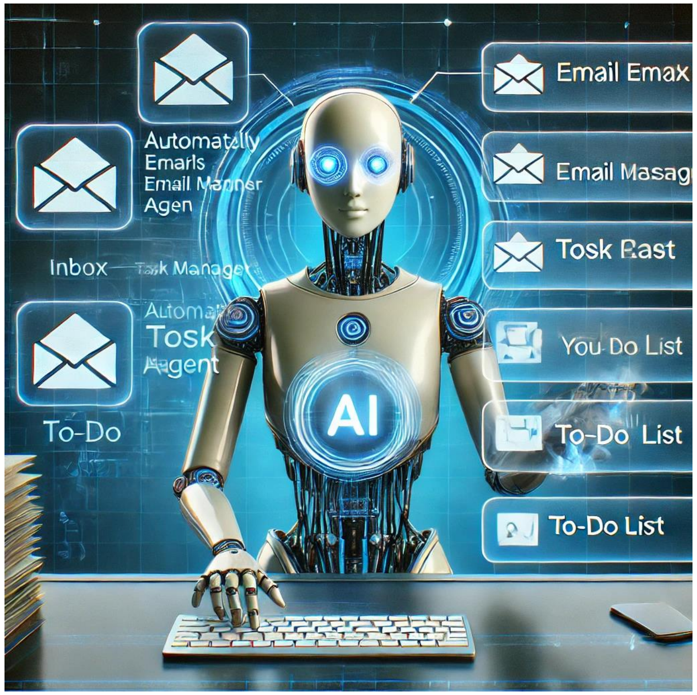
- High email volume
- Lack of prioritization
- Manual effort

Objective:

Develop an AI-powered agent for structured task management.

Introduction: Email Task Manager Agent

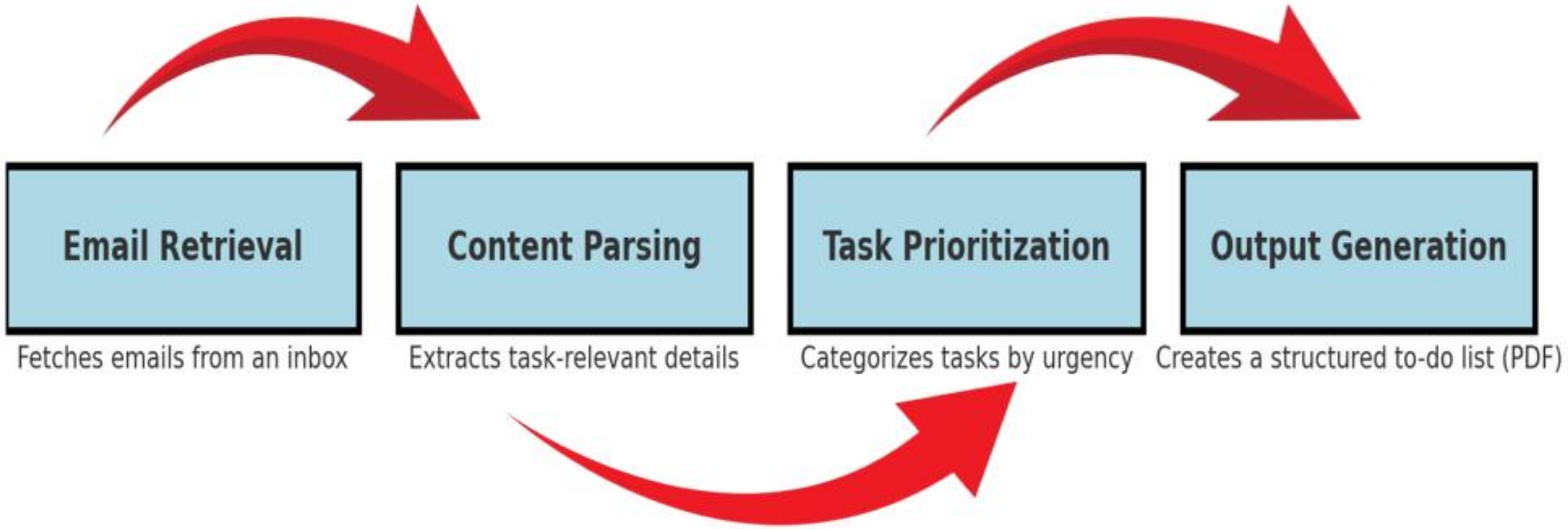
Managing emails efficiently is challenging. This application automates task extraction and prioritization using NLP.



Tools & Techniques

- Python
- NLTK
- Regex
- PDF Generation

System Workflow & Architecture


Email Retrieval
Fetches emails from an inbox
Content Parsing
Extracts task-relevant details
Task Prioritization
Categorizes tasks by urgency
Output Generation
Creates a structured to-do list (PDF)

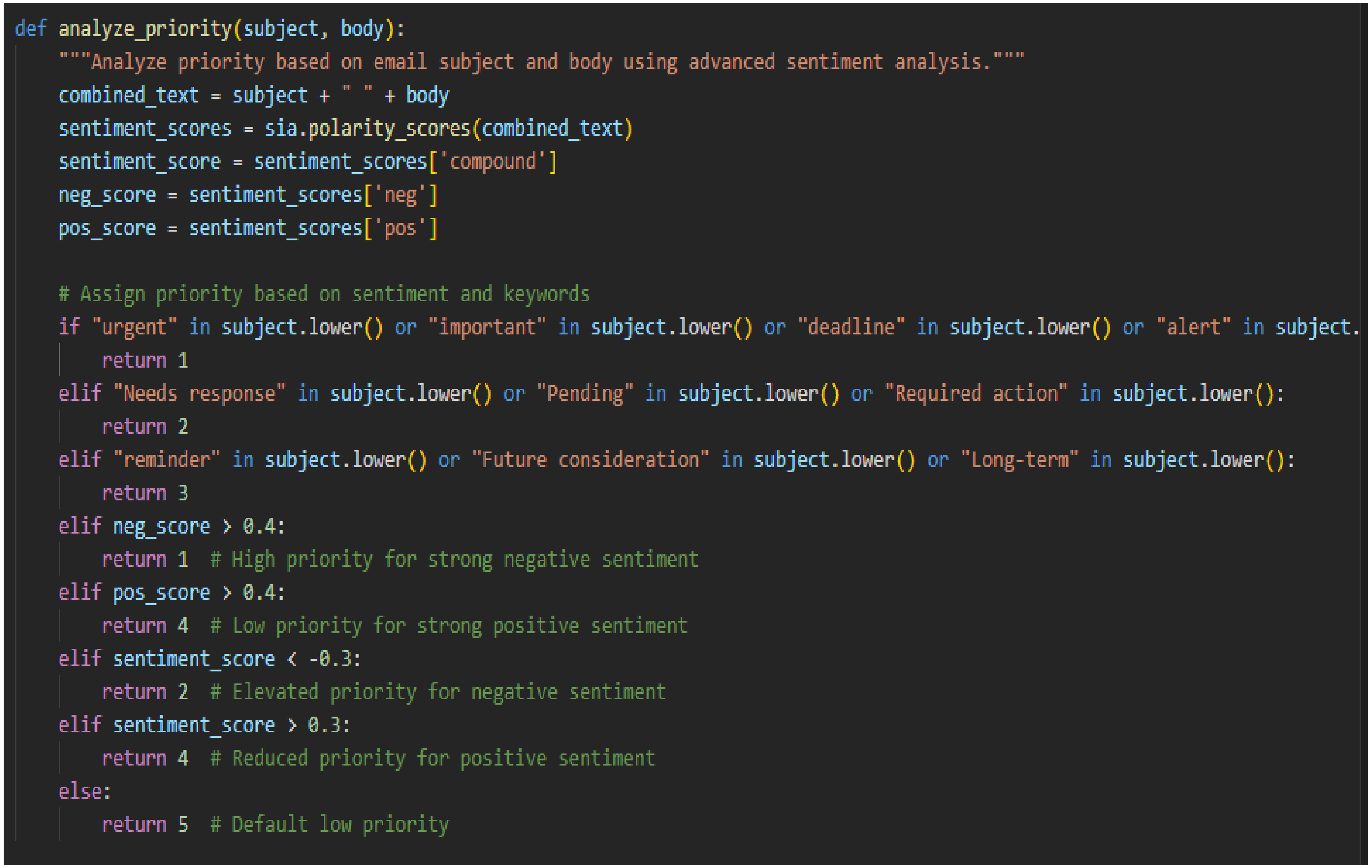
Implementation Details

Key Components:

1. Email Processing Module
2. NLP-Based Task Classifier
3. PDF Generator

Implementation Details: Key Components:

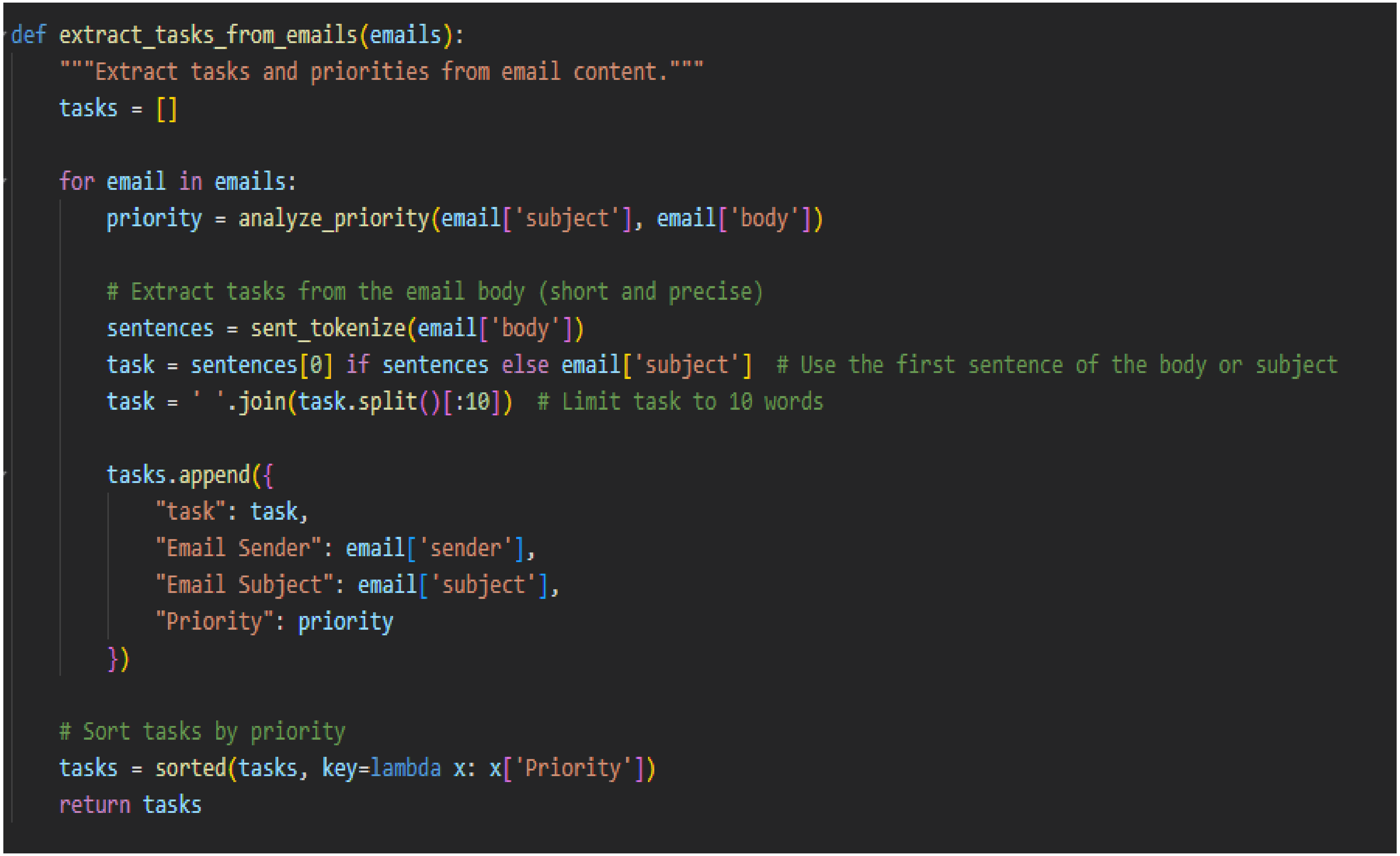
1. Email Processing Module



```python
def analyze_priority(subject, body):
    """Analyze priority based on email subject and body using advanced sentiment analysis."""
    combined_text = subject + " " + body
    sentiment_scores = sia.polarity_scores(combined_text)
    sentiment_score = sentiment_scores['compound']
    neg_score = sentiment_scores['neg']
    pos_score = sentiment_scores['pos']

    # Assign priority based on sentiment and keywords
    if "urgent" in subject.lower() or "important" in subject.lower() or "deadline" in subject.lower() or "alert" in subject.
        return 1
    elif "Needs response" in subject.lower() or "Pending" in subject.lower() or "Required action" in subject.lower():
        return 2
    elif "reminder" in subject.lower() or "Future consideration" in subject.lower() or "Long-term" in subject.lower():
        return 3
    elif neg_score > 0.4:
        return 1  # High priority for strong negative sentiment
    elif pos_score > 0.4:
        return 4  # Low priority for strong positive sentiment
    elif sentiment_score < -0.3:
        return 2  # Elevated priority for negative sentiment
    elif sentiment_score > 0.3:
        return 4  # Reduced priority for positive sentiment
    else:
        return 5  # Default low priority
```

Implementation Details: Key Components:

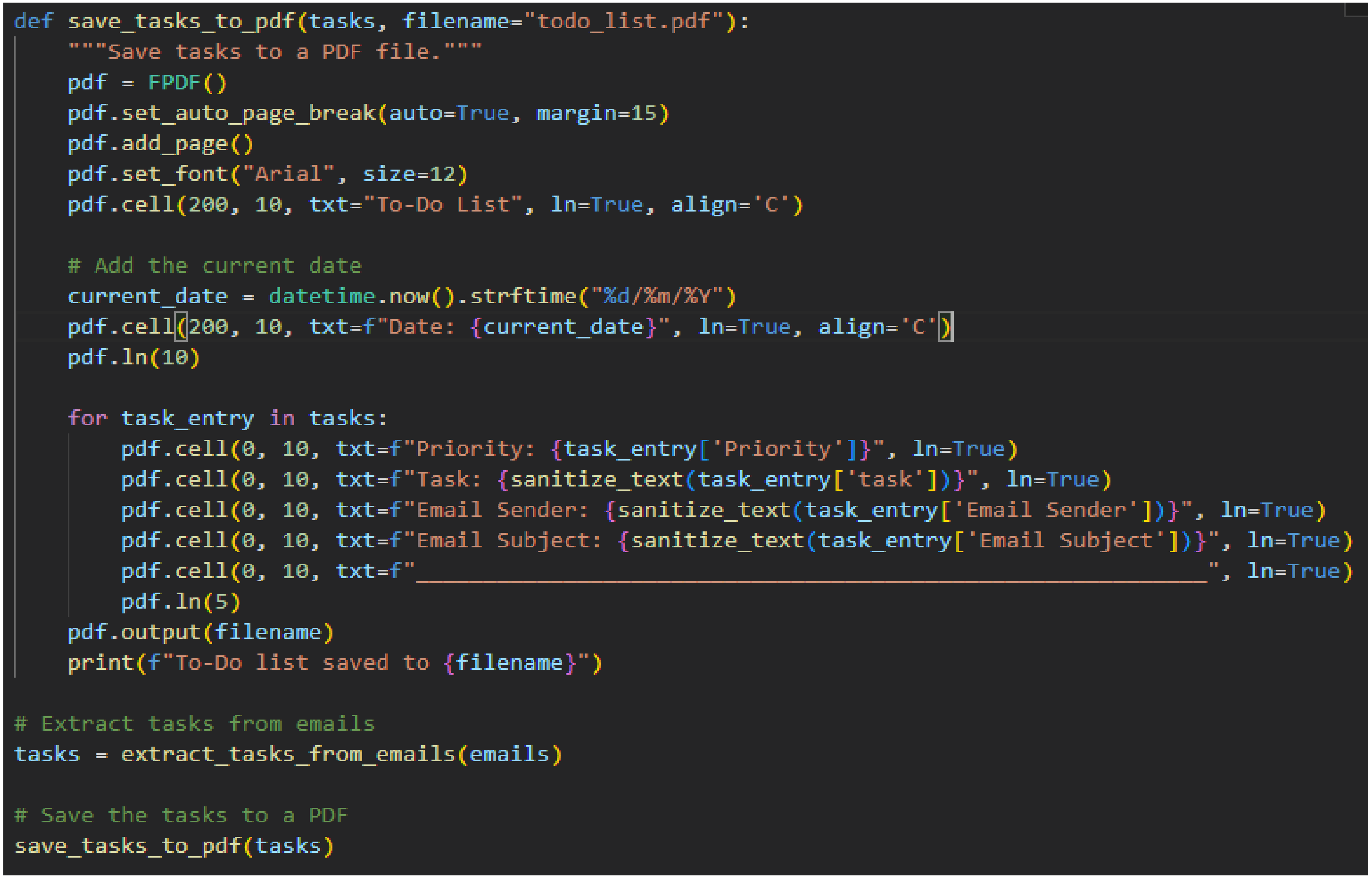
2. NLP-Based Task Classifier



```python
def extract_tasks_from_emails(emails):
    """Extract tasks and priorities from email content."""
    tasks = []

    for email in emails:
        priority = analyze_priority(email['subject'], email['body'])

        # Extract tasks from the email body (short and precise)
        sentences = sent_tokenize(email['body'])
        task = sentences[0] if sentences else email['subject']  # Use the first sentence of the body or subject
        task = ' '.join(task.split()[:10])  # Limit task to 10 words

        tasks.append({
            "task": task,
            "Email Sender": email['sender'],
            "Email Subject": email['subject'],
            "Priority": priority
        })

    # Sort tasks by priority
    tasks = sorted(tasks, key=lambda x: x['Priority'])
    return tasks
```

Implementation Details: Key Components:

3. PDF Generator



```python
def save_tasks_to_pdf(tasks, filename="todo_list.pdf"):
    """Save tasks to a PDF file."""
    pdf = FPDF()
    pdf.set_auto_page_break(auto=True, margin=15)
    pdf.add_page()
    pdf.set_font("Arial", size=12)
    pdf.cell(200, 10, txt="To-Do List", ln=True, align='C')

    # Add the current date
    current_date = datetime.now().strftime("%d/%m/%Y")
    pdf.cell(200, 10, txt=f"Date: {current_date}", ln=True, align='C')
    pdf.ln(10)

    for task_entry in tasks:
        pdf.cell(0, 10, txt=f"Priority: {task_entry['Priority']}", ln=True)
        pdf.cell(0, 10, txt=f"Task: {sanitize_text(task_entry['task'])}", ln=True)
        pdf.cell(0, 10, txt=f"Email Sender: {sanitize_text(task_entry['Email Sender'])}", ln=True)
        pdf.cell(0, 10, txt=f"Email Subject: {sanitize_text(task_entry['Email Subject'])}", ln=True)
        pdf.cell(0, 10, txt=f"___________________________________________________________", ln=True)
        pdf.ln(5)
    pdf.output(filename)
    print(f"To-Do list saved to {filename}")

# Extract tasks from emails
tasks = extract_tasks_from_emails(emails)

# Save the tasks to a PDF
save_tasks_to_pdf(tasks)
```

Expected Impact & Benefits

- Reduces Manual Work
- Enhances Productivity
- Minimizes Oversight
- Improves Task Visibility

Future Enhancements

- Multi-Language Support
- AI-Based Summarization
- User Customization Features

Conclusion

- Bridging email overload with efficient task management.
- Future improvements will enhance usability & integration.