NATURAL LANGUAGE PROCESSING ASSIGNMENT

Topic – Spelling and Grammer Checking

By,

Name	J. Babin Joe
Reg. No	42612008
Branch & Department	B.E. – CSE with AI & Robotics
Batch	2022 - 2026

SPELLING AND GRAMMER CHECKING

AIM :

To develop a Python-based application that extracts text from PDF, Word, and Text files, and checks for grammar and spelling errors, displaying the detected mistakes along with possible corrections.

PROCUDURE :

1. Import Required Libraries
   - tkinter – for creating a file upload dialog.
   - docx – to read .docx Word documents.
   - fitz (PyMuPDF) – to extract text from PDF files.
   - pytesseract and PIL – to extract text from scanned PDF images using OCR.
   - language_tool_python – to detect grammar and spelling errors.
   - io – to handle in-memory binary image data.
   - Command for installing the libraries

     "pip install python-docx PyMuPDF language-tool-python pytesseract Pillow"
2. File Selection
   - Use tkinter.filedialog to allow the user to select a file (.pdf, .docx, or .txt).
3. Text Extraction
   - PDF: Extract text directly using PyMuPDF. If no text is found (scanned PDF), convert pages to images and apply Tesseract OCR.
   - Word: Read all paragraphs using python-docx.
   - Text: Directly read the file contents.

4. Grammar and Spell Check
   - Use language_tool_python with the en-US model to detect issues in the text.
   - For each error found, extract the sentence, highlight the incorrect word, and suggest corrections.
5. Display Results
   - Print each result in the format:

     Sentence : <sentence containing the error>

     Error : <incorrect word or phrase>

     Suggestion : <possible corrections>
   - If no errors are found, display:

     No errors found. Everything is alright ✅

Note : While running this program for the first time a python wrapper called "language_tool_python" will get downloaded which is nearly 200MB.

CODING :

```
import tkinter as tk
from tkinter import filedialog
import docx
import fitz
import language_tool_python
import pytesseract
from PIL import Image
import io
```

```
# Optional: Set Tesseract path for Windows users
# pytesseract.pytesseract.tesseract_cmd = r"C:\Program Files\Tesseract-OCR\tesseract.exe"

def extract_text_from_pdf(file_path):
    text = ""
    pdf_document = fitz.open(file_path)

    for page_num in range(len(pdf_document)):
        page = pdf_document[page_num]
        page_text = page.get_text()

        if page_text.strip():  # Normal text extraction
            text += page_text
        else:
            # OCR for scanned PDF pages
            pix = page.get_pixmap()
            img = Image.open(io.BytesIO(pix.tobytes("png")))
            text += pytesseract.image_to_string(img)

    return text

def extract_text_from_word(file_path):
    doc = docx.Document(file_path)
    return "\n".join([para.text for para in doc.paragraphs])

def extract_text_from_txt(file_path):
    with open(file_path, "r", encoding="utf-8") as f:
```

```
        return f.read()

def check_grammar_and_spelling(text):
    tool = language_tool_python.LanguageTool('en-US')
    matches = tool.check(text)

    if not matches:
        print("No errors found. Everything is alright ✅ ")
        return

    for match in matches:
        sentence = match.context.strip()
        error_text = match.context[match.offset:match.offset + match.errorLength]
        suggestions = ", ".join(match.replacements) if match.replacements else "No suggestions"

        print(f"Sentence : {sentence}")
        print(f"Error    : {error_text}")
        print(f"Suggestion : {suggestions}")
        print("-" * 50)

def main():
    root = tk.Tk()
    root.withdraw()
    file_path = filedialog.askopenfilename(
        title="Select PDF, Word, or Text file",
        filetypes=[
            ("All supported files", "*.pdf *.docx *.txt"),
            ("PDF files", "*.pdf"),
```

```
            ("Word files", "*.docx"),
            ("Text files", "*.txt")
        ]
    )

    if not file_path:
        print("No file selected.")
        return

    if file_path.lower().endswith(".pdf"):
        text = extract_text_from_pdf(file_path)
    elif file_path.lower().endswith(".docx"):
        text = extract_text_from_word(file_path)
    elif file_path.lower().endswith(".txt"):
        text = extract_text_from_txt(file_path)
    else:
        print("Unsupported file format.")
        return

    check_grammar_and_spelling(text)

if __name__ == "__main__":
    main()
```

INPUT AND OUTPUT :

Input : Uploaded file “TEST 1.txt”

Output :
PS C:\Users\Babin Joe\Downloads\NLP Assignment> & C:/Python313/python.exe "c:/Users/Babin Joe/Downloads/NLP Assignment/Assignment.py"

Sentence : Hello, My name is John. I is doing Computer Science.

Error : is

Suggestion : am

SUMMARY :

This project demonstrates Natural Language Processing (NLP) techniques to automatically detect grammar and spelling errors in documents.

Concept:

- Text extraction is performed differently depending on file type:
  - PDFs are handled by PyMuPDF, with OCR (Tesseract) for scanned documents.
  - Word files are processed with python-docx.
  - Text files are read directly.
- Grammar & spelling checking is achieved using language_tool_python, which provides rule-based error detection and correction suggestions.

How it works:

1. User uploads a document.
2. The system extracts text from the file.
3. The text is sent to the grammar checking tool.

4. Detected issues are printed with context, error, and suggestions.

This approach avoids printing unnecessary text, focusing only on the results of the check.

Software Tools Used:

- Python – Core programming language.
- PyMuPDF (fitz) – PDF text extraction.
- python-docx – Word file processing.
- pytesseract & PIL – OCR for scanned PDFs.
- language_tool_python – Grammar and spelling checking.
- tkinter – GUI for file upload dialog.

APPLICATIONS :

- Automated proofreading of academic assignments, reports, and official documents.
- Grammar checking in publishing and content creation.
- Assisting non-native speakers in improving writing quality.
- Pre-submission checking for research papers.
- Integration into e-learning platforms for instant feedback.