AI ENGINE  
Date:20/01/2024

# SUMMARIZATION FILE

**Libraries Used**:

* PyPDF2: For reading PDF files.
* cv2: OpenCV library for image processing.
* pytesseract: For optical character recognition (OCR) on images.
* fpdf: A library for creating PDF files.
* gtts: Google Text-to-Speech for converting text to speech.
* io: For handling byte streams.
* googletrans: Google Translate API for language translation.
* sumy: Text summarization library.
* docx: For reading Word documents.
* textwrap: For wrapping text to fit a specified width.
* nltk: Natural Language Toolkit for natural language processing tasks.
* spacy: An open-source library for advanced natural language processing.
* sklearn.feature\_extraction.text: For TF-IDF vectorization.
* spellchecker: Library for spell checking.
* language\_tool\_python: Interface to the LanguageTool API.

***Class: TextProcessor***

*extract\_text\_from\_image(scanned\_image):*

* Takes a scanned image as input.
* Converts the image to grayscale.
* Uses OCR to extract text from the image.
* Returns the extracted text.

*extract\_text\_from\_pdf(pdf\_path):*

* Takes the path to a PDF file as input.
* Reads the PDF file and extracts text from each page.
* Returns the concatenated text from all pages.

***extract\_text\_from\_text\_file(text\_file\_path):***

* Takes the path to a text file as input.
* Reads the text file and returns the content.

*extract\_text\_from\_word\_document(docx\_path):*

* Takes the path to a Word document as input.
* Reads the Word document and returns the text.

*get\_sentences\_count(text, summary\_length):*

* Takes text and a summary length as input.
* Calculates the number of sentences for summarization based on the specified length.
* Returns the calculated number of sentences.

*summarize\_text\_sumy(text, sentences\_count):*

* Takes text and the number of sentences for summarization as input.
* Uses LSA (Latent Semantic Analysis) for text summarization.
* Prints and returns the summary text.

*text\_to\_pdf(text, filename="summarized\_text.pdf"):*

* Takes text and an optional filename as input.
* Converts the text to a PDF file with specified formatting.
* Returns the filename.

*text\_in\_words\_and\_sentences(text):*

* Takes text as input.
* Counts the number of words and sentences in the text.
* Returns the counts.

*extracted\_text\_words(text, num\_keywords=5):*

* Takes text and the number of keywords as input.
* Processes the text, extracts keywords using TF-IDF, and returns the top keywords.

*summarize\_content(text, selected\_keywords):*

* Takes text and user-selected keywords as input.
* Extracts sentences containing the selected keywords and generates a summary.
* Returns the summary.

*calculate\_reduction\_percentages(extracted\_text\_words, extracted\_text\_sentences, summary\_text\_words, summary\_text\_sentences):*

* Takes counts of words and sentences from both the extracted text and the summary.
* Calculates reduction in words and sentences.
* Calculates reduction percentages.
* Returns information about the reduction.

*create\_bullet\_points(text, format\_choice):*

* Takes text and a format choice ("bullets" or "paragraphs") as input.
* Processes the text using spaCy and creates bullet points or paragraphs.
* Returns the formatted text

# Dictation Class

**Class: Dictation**

**Methods:**

*correct\_spelling\_and\_grammar(text):*

* Uses SpellChecker to correct spelling in the provided text.
* Utilizes LanguageTool for grammar correction.
* Returns the corrected text.

*translate\_text(text, target\_language='en'):*

* Uses Google Translate API to translate the provided text to the specified target language.
* Prints the original and translated texts.
* Returns the translated text.

*recognize\_and\_correct():*

* Utilizes the speech\_recognition library to recognize speech from the microphone.
* Translates the recognized speech to English.
* Corrects spelling and grammar of the translated text.
* Returns the corrected text.

# Translation\_File Class

**Class: translation\_file**

**Methods:**

*extract\_text\_from\_image(scanned\_image):*

* Takes a scanned image as input.
* Converts the image to grayscale.
* Uses OCR to extract text from the image.
* Returns the extracted text.

*extract\_text\_from\_pdf(pdf\_path):*

* Takes the path to a PDF file as input.
* Reads the PDF file and extracts text from each page.
* Returns the concatenated text from all pages.

***extract\_text\_from\_text\_file(text\_file\_path):***

* Takes the path to a text file as input.
* Reads the text file and returns the content.

*extract\_text\_from\_word\_document(docx\_path):*

* Takes the path to a Word document as input.
* Reads the Word document and returns the text.
* Reads a Word document and returns the text.
* Prints and returns the text.

*translate\_text(text, target\_language):*

* Translates the provided text to the specified target language using Google Translate API.
* (Google Translate API translate 5000 tokens only.)
* Prints and returns the translated text.

*text\_to\_pdf(text, filename="summarized\_text.pdf"):*

* Converts the given text to a PDF file using FPDF.
* Prints and returns the filename of the generated PDF.

# Audio Generation

*Class: audioprocess*

*Methods:*

*extract\_text\_from\_image(scanned\_image):*

* Uses OpenCV and Tesseract to extract text from a scanned image.
* Returns the extracted text.

*extract\_text\_from\_pdf(pdf\_path):*

* Reads a PDF file and extracts text from each page using PyPDF2.
* Returns the extracted text.

*extract\_text\_from\_text\_file(text\_file\_path):*

* Reads a text file and returns its content.
* Returns the text.

*extract\_text\_from\_word\_document(docx\_path):*

* Reads a Word document and returns the text.
* Returns the text.

*text\_to\_speech(text, language):*

* Uses gTTS to convert the given text to speech in the specified language.
* Returns the audio data.

detect\_language(text):

* Uses langid to detect the language of the given text.
* Returns the detected language.