Final Mini Project Demonstration

Subject CODE

Project Title	:	Web App for Quiz Generation
Project Guide	:	Dr. S Natarajan
Project Team	:	Abdul Mannan- PES1UG19CS007
		Ishita Chaudhary- PES1UG19CS190
		Jason Carvalho- PES1UG19CS195

Project Abstract and Scope

Abstract-
Our project is essentially a web application which takes in a pdf as an input.
It then converts the pdf to text. Based upon this text, the user gets an automatically generated quiz which consists of two types of questions.
We also display the actual answers after the quiz.

Usability of our project-
Helps the student to test her/his knowledge of the topics.
Useful for self-study and reduce the workload of the teachers.
Keeping the student more interested while studying.

Scope:
The project is divided into three parts-
PDF to text conversion
Automatic Question Generation
Building the UI (website)

Literature Survey

[1]‘PDFminer’ parses all objects from a PDF document into Python objects. The layout analysis consist of three different stages: it groups characters into words and lines, then it groups lines into boxes and finally it groups textboxes hierarchically. The resulting output of the layout analysis is an ordered hierarchy of layout objects on a PDF page. Hence text is analysed and grouped in a human-readable way.

[2]Das and Majumdar came up with an interesting way of generating fill in the blank questions by using rule based approach which involves part of speech tagging and sentence constructors,etc. It doesn’t follow a template based approach like many other question generators. This was used for informative sentences only.

[3]Alsubait insists that the toughest part of generating MCQ’s is generating distractors. An ontology based approach was used to generate distractors.They used a ‘similarity measure’ to generate questions of varying difficulty by varying similarity between the key and distractors.

[4]Multipartite Rank is a graph based model similar to TextRank. In this model first a graph is built of the document on which then the ranking algorithm assigns each keyphrase a relevance score. Keyphrase candidates are selected from sequences of adjacent nouns with one or more preceding adjectives. The edges between the nodes (keyphrases) represent the co-occurrence relation between them, usually controlled by the distance between the word occurrences.. The multipartite graph for keyword extraction outperforms the baselines on most metrics.

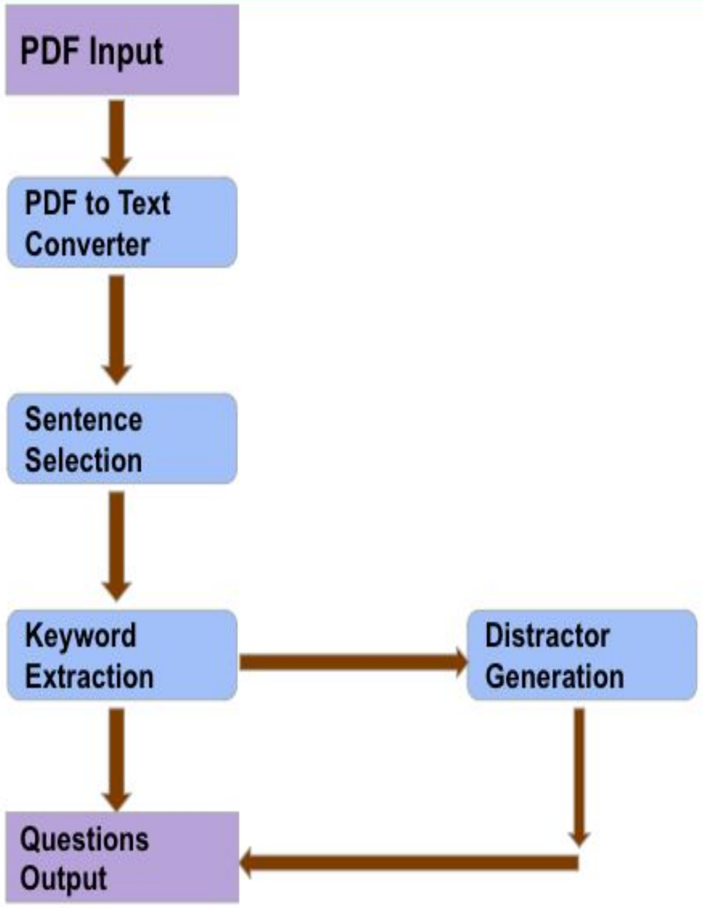
[5]An automatic, open-cloze question generation system of generating questions was put forward by Manish Agarwal. It is a rule based system for selecting informative sentences and identifying keywords in these sentences. The dataset used was News reports on Cricket matches. 22 questions (10+12) were generated and evaluated by three different evaluators. The overall accuracy of the system was 3.15 (Eval-1), 3.14 (Eval-2) and 3.26 (Eval-3) out of 4.

[6] TextRank is a graph-based ranking model mainly utilised for keyword and sentence extraction. If a sentence has similarities (calculated by using the number of common tokens of the sentences) to another sentence, it gets upvoted by that sentence. The sentences are then ranked in the order of their votes.

[7] RAKE is an unsupervised method for extracting keywords from text data. Stop words and phrase delimiters are used to break down/ segregate the data. Each word is given a score based on word co-occurrence and highest scored words are chosen as keywords. As the number of words in the document increases, RAKE outperforms TextRank.

[8]This paper uses a mix of syntax and semantic based approach to generate wh- type questions from both simple and complex sentences. It makes use of the Stanford Tagger for POS tagging. It then uses NER over the sentence for identifying the Subject, Object, etc. Questions on complex sentences are determined by the discourse connective(conjunctions). It also extracts the auxiliary verb (if present) to help figure out the tense of the sentence, which is important in proper question generation.

Design Description/UI Design



UI Design

(Add final website screenshots)

Technologies Used

For PDF to text conversion-

- PyPDF2
- Pdfminer.six

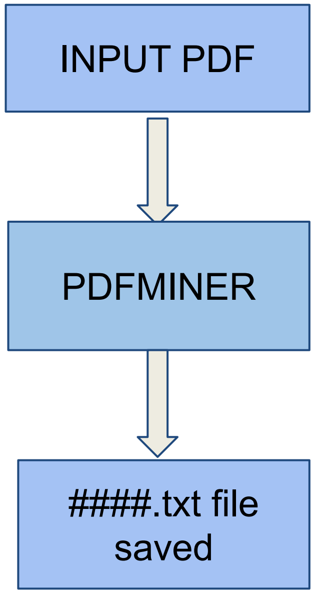
For Question Generation-

- PKE
- NLTK python package
- Wordnet
- ConceptNet

For Website creation-

- HTML
- CSS
- Flask

Implementation and why we used specific technologies



PDF Conversion to Text:
The input pdf is taken using a form. The **PDFminer library** is used to convert the pdf to text file which is present in same directory which is then fed to the question generation function. We used PDFminer because it gave very accurate conversions of the test PDFs.

PDFminer.six parses all objects from a PDF document into Python objects. First we do the layout analysis. This consist of three different stages: it groups characters into words and lines, then it groups lines into boxes and finally it groups textboxes hierarchically. The resulting output of the layout analysis is an ordered hierarchy of layout objects on a PDF page. Hence, text is grouped appropriately and then we store this text into a text file.

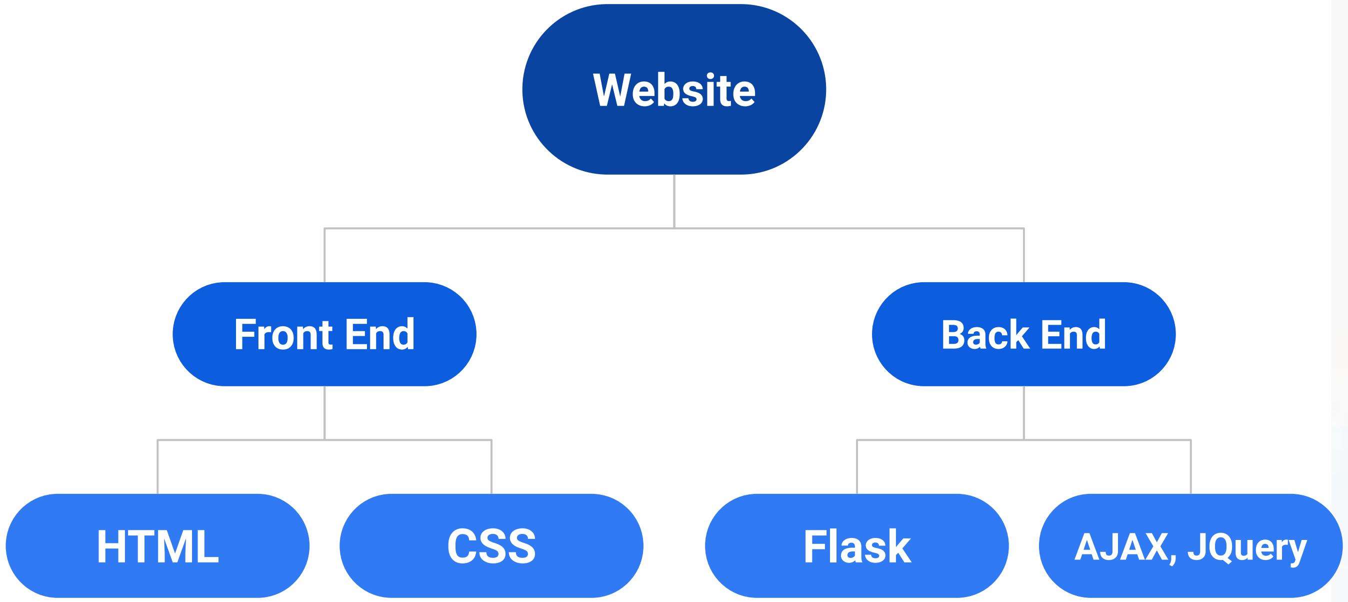
Question Generation:

We use a very simple algorithm to generate questions from some text. We first pre-process the text to remove stop words and punctuations, basically forming clean sentences that can be used to generate a summary for the text . We then find the keywords of the whole text using **Multipartite** graphs. A check is then done to see if the keyword exists in the summarized text, if it does then that keyword is chosen and all sentences that it appears in is mapped to it.

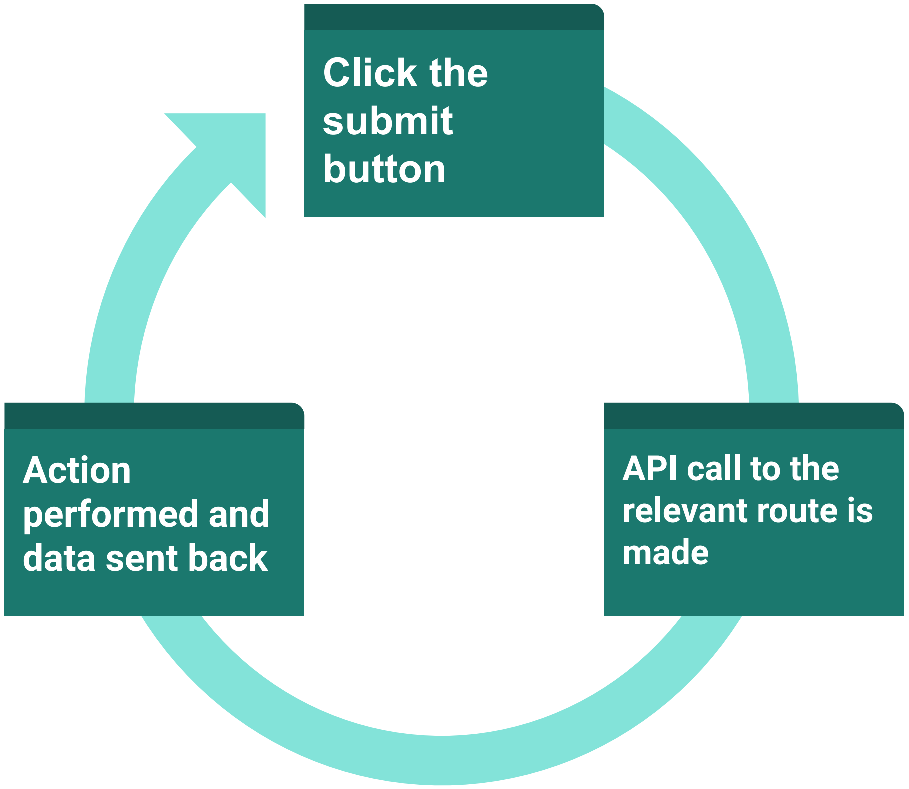
We then use the keywords to form a distractor words list using functions from wordnet and conceptnet. Both provide easy to use API's to get similar words.

Random sentences are chosen and then the keyword in it is replaced with a blank. These sentences are then appended to a dictionary. Same thing is done for fill-in-the-blank questions and both these are returned as a tuple to the function call.

For the website, we used:



We used **Flask** to create a server as the pdf converter and question generator are implemented using python.


Click the submit button
API call to the relevant route is made
Action performed and data sent back

PDF to text file:

- When we press the submit button, we make an API call to the route that converts PDF to a text file by calling the conversion algorithm.

Generating questions:

- Similarly, when we press the ‘Generate Questions’ button, we’re making an API call to question generation route. We receive the data from the Question generation in the form of a tuple, containing FIB and MCQ questions.

Display Quiz:

- We then display the data in the form of a quiz using **flask templates.**

Project Demo

Prepare for a demo of the project:

Need to show the working of the project.

Thank You!
:)