Interactive learning tool utilizing AI

PA026 Artificial intelligence project - spring 2024

authors:

Jan Franěk: 469204@mail.muni.cz Martin Kňažovič: 553635@mail.muni.cz

1.Introduction

1.1 Motivation

Learning is not an easy process and different people prefer various learning methods, speed and environment. Can AI be used to help people with learning anyhow? Either by unifying the study materials or by some automatization?

1.2 Goals of the project

Goal of this project is to utilize and explore capabilities of publicly available AI tools and use them for learning enhancement. Our idea is to focus on documents types most used by the students (videos and pdf) and make the whole studying process using these materials more efficient.

1.3 Similar tools

Even though we are not aware of any other tools with similar flow to what we wanted to achieve, there are tools which solve similar tasks.

PopAI

Pdf summarizer which can also create presentations from PDF and feeds ChatGPT with data from the source files so you can ask follow-up questions. Due to its non-free business model we had very limited time with it, but it at least seemed like an interesting alternative.

Adobe

Adobe viewer pdf summarizer. Although not as interesting in features as PopAI, we believe it might be more consistent in what it does.

Pictory

Authors of this tool claim this tool can be used for video summarizing, but it looks most people seem to be using a different feature of this tool which is text to video generator. From the demo you might find out why. This tool’s video summarizing takes a long time and you need a 20$/month subscription.

Harpa AI

The biggest contender to our tool. It's a web browser extension which can quickly summarize whole pages, videos, create watchdogs, custom extraction command and much more. It is a shame that it does not support viewing or working with PDFs. It's free to use and can do

much more than just learning enhancement. But at the same time it is not focused purely on the learning and we believe we could outperform it in this field.

2. Implementation

We've chosen the web application as the target platform for the project and that's why technologies, flow and features are all revolving around that decision. The web application serves as the frontend and also requires a backend server to solve some more computationally heavy tasks.

2.1 Technologies

Front-end

For the frontend we used mainly JavaScript framework Svelte together with Flowbite library, that provides premade UI components and tailwind as CSS framework.

Backend

- For the backend we are using Python with Flask as an API library for the backend - frontend communication.
- For the video-to-text extraction we are using Whisper (by OpenAI) and for PDF extraction we are using pyMuPDF.
- For LLM requests we use Openai API for keyword extraction and test generation.

2.2 Project structure

The project folder consists of two main folders: `kews` and `kews-fe`.

kews

This folder consists of the backend implementation in Python.

`kews`

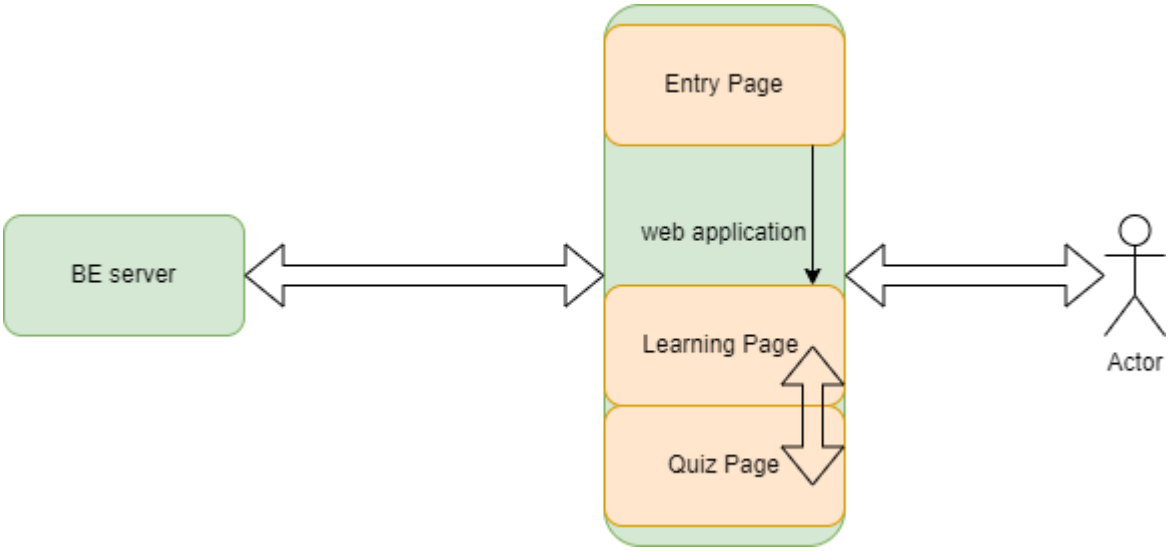
- `src`
  - `ExtractorInterface.py` - common interface for text Extractors
  - `Keywords.py` - keyword extraction functionality
  - `PdfExtractor.py` - PDF text extractor
  - `RequestResponse.py` - defines interface for server responses
  - `TestGenerator.py` - test generator component
  - `VideoExtractor.py` - Video text extractor
- `tests` - folder with unit tests
- `main.py` - main backend entry point
- `requirements.txt` - list of required Python libraries

kews-fe

This folder consists of the frontend implementation.
There are many folders from the Svelte framework so we are mainly highlighting our implementation parts. Note that all pages must start with `+page.svelte` because Svelte uses directory routing..
`kews-fe`

- `src`
  - `routes` - contains implementation of all project pages
    - `learning` - implementation of the main page
    - `quiz` - implementation of the quiz page
    - `+page.svelte` - entry page where user provides document source
    - `+layout.svelte` - definitions of common styles
    - `+page.server.js` - logic for validation of provided document source
  - `lib` - contains modified library of the quiz

2.3 Flow description



Let's start with the top-level overview. When an actor arrives at our application entrypoint he has to provide the URL resource of the document he wants to analyze. This resource should be a valid Youtube video or PDF link that can be then sent to backend information extraction. (Described in next section) When the backend collects all the necessary information, it's sent back to the frontend and the user can start the learning process. When the actor is finished with learning he can progress to the quiz page to test its knowledge.

Text extraction

Whether a user uses PDF or video as an input, the requested resource is then sent to the backend, downloaded and processed. Videos are processed as mentioned with Whisper

model and text is extracted using pyMuPDF library. This text is then processed and returned to the frontend where the actor can work with it.

Keywords extraction

We take the extracted text and use the Chat GPT API to extract keywords which the LLM deems are the most important. The user can see those words highlighted on the frontend and interact with them. This should help the user to realize which part of the requested resource is most important.

The API prompt is consisting of setting role to the system using the following prompt:

“You will be provided with a block of text, and your task is to extract a list of unique (no terms overlapping) keywords separated by commas”

and then we follow-up with feeding the API with the extracted text.

Quiz generation

App picks a subset of the selected keywords and sends a request to the LLM so that it creates a test from the given keywords. This is an optional part of the flow meant for the people who would like to test their newly acquired knowledge. We are aware there are a lot of apps online which let you create your own test out of given pictures/questions, but having the tests generated by AI has the advantage of avoiding overfitting on the already known small dataset of questions provided by f.e. your professor.

The API prompt is consisting of setting role to the system using the following prompt:

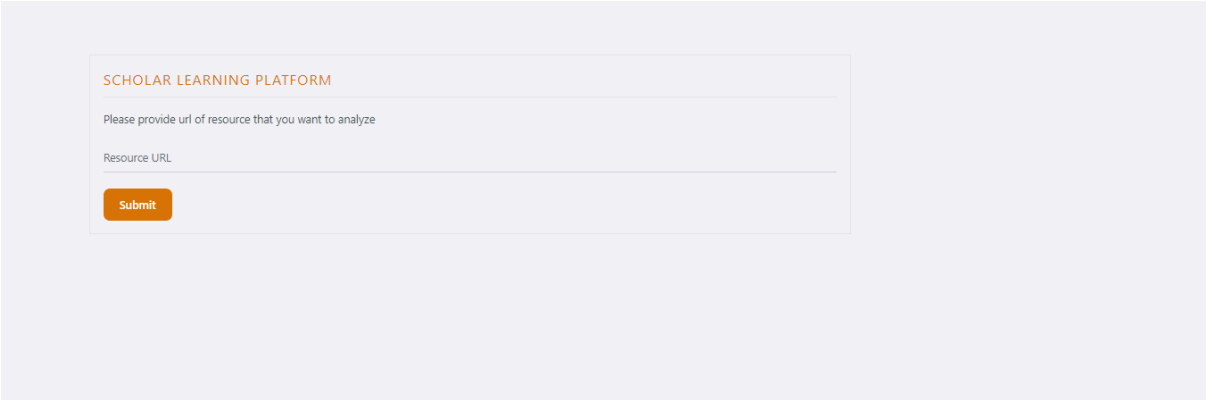
"Generate test with {self.num_questions} questions with {self.num_answers} options per question asking about the correct definitions for the keywords given by the user. No keyword must be present in the options. After you generate the questions, you must also print the list of the right answers. For more context about the given keywords, there are the related keywords to the topic: {self.keywords}”
and then we give him few random extracted keywords to generate the quiz for.

We use a random subset of keywords (if small amount of keywords if extracted, we use all) to give model more context and also to seduce the token count.

2.4 UX description

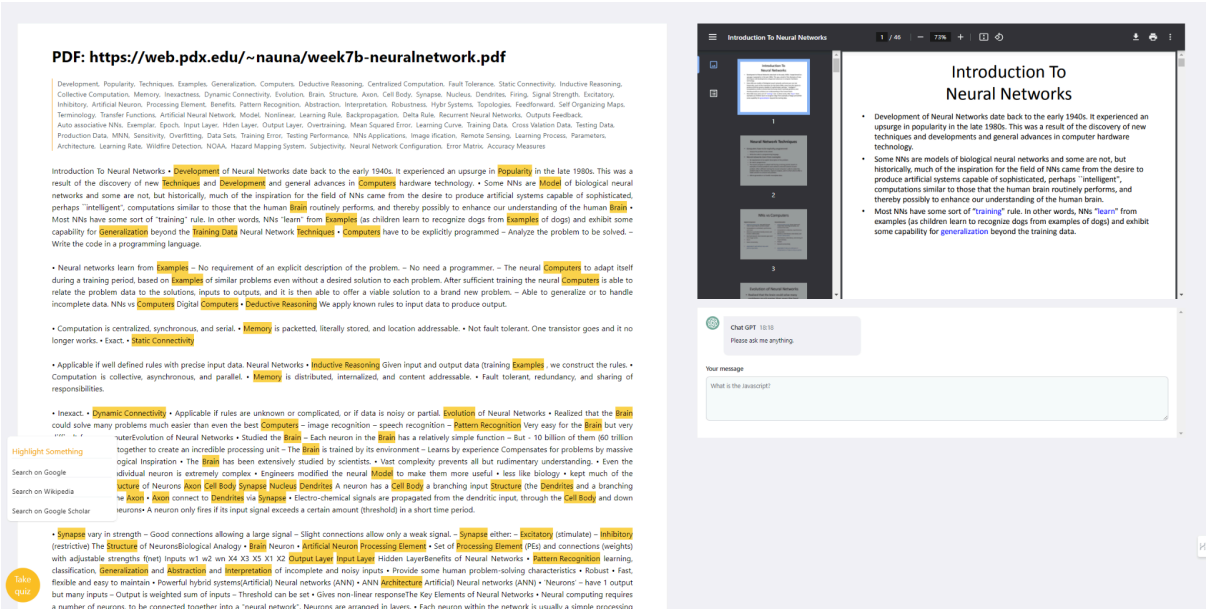
Entry page

Serves as an input form for the user. Only publicly accessible links can be used for now. Local files are not yet supported to reduce server storage costs. For video resources we support all videos accessible on Youtube.Otherwise the provided resource must be a valid PDF link. By clicking submit, the user can proceed with selected document to the main project page.



Learning page

On the left side of the page you can see an extracted transcript from the input data with highlighted keywords. You can left-click keywords for additional options (mostly automatic search on platforms like Wikipedia, Google) or use integrated ChatGPT directly in the right-bottom page. Right-top part serves as a viewer for your input data. Context menu on the bottom left part provides the same functionality as the keywords menu but works for the user highlighted text (no need to copy and search for the text manually). For fast text searching, users can press “s” + “d” at the same time to set search text and “s” + “f” for looping found text. (“s” + “d” could be skipped altogether and same search prompt should open when no text was set with “s” + “d”)
For PDF the native PDF viewer is used and videos are directly loaded from the youtube. By clicking on the left-most-bottom button you can access the quiz page.

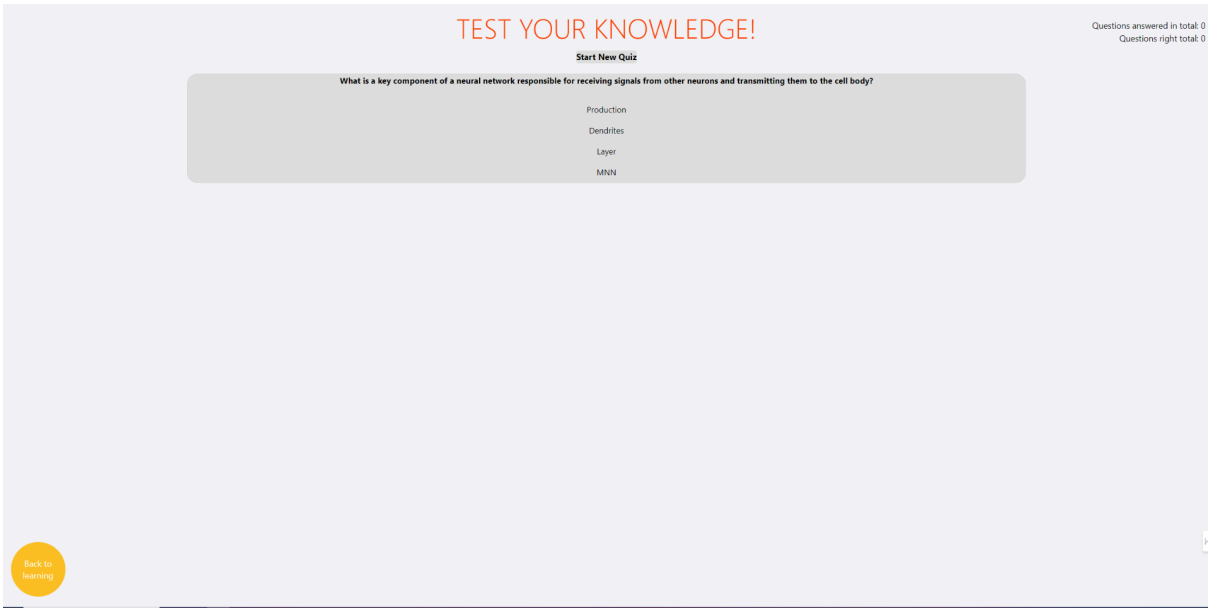


Quiz page

By clicking on the button “start new quiz” and waiting a few seconds you can see an AI created test based on your previous data given. For now it is hardwired to 5 ABCD

questions, but you can create a new test as many times you want. After each sequence of questions you can view the correct solution or continue with another quiz.

By clicking on the left-most-bottom button you can return to the main page.
Beware that doing so will trigger the extraction flow again due to missing cookies that would keep all the information.



3. Installation

3.1 Prerequisites

- Python 3.9+
  - Python libraries required are located in /kews/requirements.txt
  - `pip install -r requirements.txt`
- JavaScript installed via npm
  - you will also need to install JS packages using npm install (node is required)
- For video processing you will also need ffmpeg installed
  - https://ffmpeg.org/download.html
  - easiest option is to run choco install ffmpeg if you have installed chocolatey
- ChatGPT API key with sufficient credit
  - get here: openai web
  - set env var GPT_TOKEN to be equal to your key

3.2 Running the web app

- Start first by running the backend server by simply calling python kews/main.py
- Then start the web application by running npm run dev in kews-fe folder
- Now open your browser and visit your local server (see frontend console for address)

- link may differ for you so check printouts from the previous command

4. Evaluation

Since both LLMs and humans are behaving non-deterministically it is really hard to do 100% objective evaluation on specific tasks. We believe this is one of such tasks and we at least try to do our best to measure the quality of the tool by different methods from unit tests and benchmarks to human evaluation. Human evaluation is slow, but building a natural language processing testing environment, vision sensors and brains to learn would be even slower.

4.1 PDF text extraction

method: unittest/benchmark

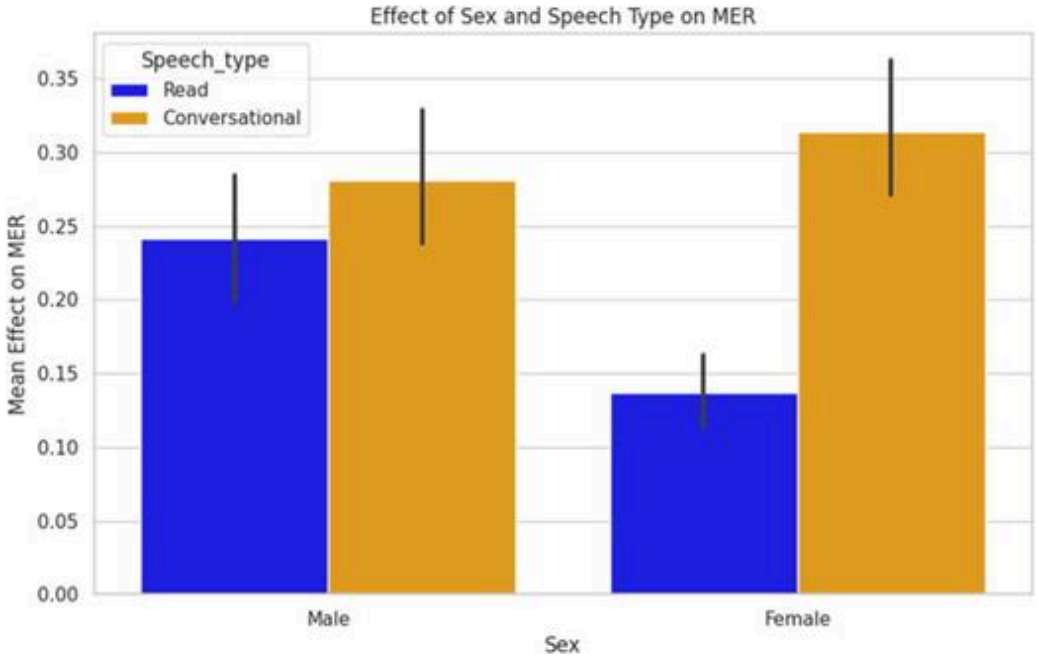
We prepared a dataset of 6 PDFs to test out if the tool correctly extracts the text as humans would do.
These are the results:

- Text similarity for the dataset simple_text: 99,47%
- Text similarity for the dataset code_in_text: 98, 61%
- Text similarity for the dataset recipe: 98, 88%
- Text similarity for the dataset biology_text : 98, 59%
- Text similarity for the dataset statistics: 97,67%
- Text similarity for the dataset presentation_cnn: 71,16%

After evaluation of the results we concluded that in general the text extraction is good enough for this project, but can be really unreliable for presentations with hidden text, a lot of picture data, repeated info on slides (due to transitions), math formulas and also complex hierarchy.

4.2 Video text extraction

Since Whisper is widely known tool we took an advantage of an existing more rigorous evaluation [1]. The studied paper evaluates Whisper text from voice capabilities using MER (Match Error Rate = percentage of correctly guessed words) and tries to compare different variables of it like gender of the speaker, accent and the speakers native language.



4.3 Keyword extraction

method: handmade benchmark
We transcribed several videos from Youtube and selected one talking about elections [2] and let third person manually extract the keywords from it - according to his best intentions without any additional instructions. We then generated keywords and compared results from multiple models using cosine similarity and got the following numbers:

- ChatGPT 3.5 (0.894)
- KeyBERT (0.891)
- Rakun2 (0.736)

We concluded that ChatGPT 3.5 is not only the most straightforward tool to use, but also seems to be the most reliable one. Disclaimer: scope of this benchmark was rather small because we didn't have enough human resources for the manual annotation.

4.4 Quiz generation

method: human evaluation

The quiz suffers from a lot of issues and the most severe ones observed are described here.

LLM Hallucinating

One of the most frequent buzzwords of the year 2023 and one of the biggest AI weaknesses. Some tests were generated with blatantly wrong questions/answers even though the "creativity" parameter is set to low value.

Repetitive patterns

The generated answers are usually one word answers mostly being the given keywords. From time to time there is a question of type “what is X" or “what is the function of Y " but a higher ratio of such questions is desired . The user would definitely benefit from more diverse answers.

Hints

Since the LLM has access to all keywords given at once as it is ineffective to send them one by one, it can hint the answers indirectly to the user. It does so by using keywords from the different questions as an answer to the current one. I believe this could be a workaround by giving more specific requests while generating the tests.

Context

Keywords are just a short part of the data context and it may happen that Quiz is generated with correct keywords but used in the wrong context. For example some of the keywords from random Convolutional Neural Networks presentation could be used also in the context of biology. We noticed this issue and added more hidden keywords that just the ones we want to generate tests for. This way the LLM can derive more context, but still this method is not bulletproof. We might add one more pipeline to rank the important keywords to fight that.

4.5 User experience

We let several people try our application and gathered their feedback for this part.

There were in total 6 respondents which are anonymously presented in this table:

Sex	**Field of studies**
F	Business Valuation
F	Architecture
M	IT
M	IT
M	IT
M	Electrical engineering

The feedback is separated into positive and negative notes. Both categories contain only notes which at least 2 respondents said independently of each other. The evaluation method was a brief introduction to the tool, a short period of observation of the respondent and his actions and then a small series of open questions about his experience.

positive notes

- The app interface looks nice (4x)
- I liked I can immediately text my knowledge through the quiz (4x)
- I prefer reading the text over listening to whole video (2x)
- I would use this tool (2x)

negative notes

- The keywords extracted are wrong/unreliable(4x)
- Context of the quiz questions is somehow lacking (3x)
- I would like to have pictures as an input (2x)
- Its taking too long (2x)

We are very happy about the positive notes and for the negative notes we would mostly need to improve our usage of the LLM APIs or rather hope for a better LLM version in the future.

5. Summary

Despite initial uncertainty about the goal of the project, we managed to create a working app which is in the same cases really helpful even in the current version. We could experience the current AI limitations by our own hands and also discovered how much more the AI can be utilized even with a small project wrapping around it like this one.

We are well aware that in the current state, our tool still has many steps to become easy to use and more reliable, but the potential which AI enhanced learning has is already noticeable and will only grow bigger over the next year.

We hope that you find this project and documentation helpful and it inspires you to try to do your own project.

6. Sources

[1] Graham, C., & Roll, N. (2024, February 23). *Evaluating openai's whisper ASR: Performance analysis across diverse accents and speaker traits*. AIP Publishing. https://pubs.aip.org/asa/jel/article/4/2/025206/3267247/Evaluating-OpenAI-s-Whisper-ASR-Performance

[2] Why US elections only give you two choices. (n.d.). [Video]. YouTube. Retrieved May 24, 2024, from https://www.youtube.com/watch?v=bqWwV3xk9Qk