**Possible Projects**

* It is recommended to start implementation from a simple pipeline and then gradually make it more complex (by fine-tuning (instruction fine-tuning), for example, or adding additional steps and pipelines)
* You can also add some specific details and focus on the project, feel free to discuss it with us!
* No more than two commands per project

| **Name** | **Description** | **Datasets / starting info** | **Note** |
| --- | --- | --- | --- |
| Teacher Assistant | The project is devoted to creating meaningful knowledge evaluation questions on documents provided by the user (teacher)  **Tasks/preferred features:**  - You should create a pipeline to generate various questions covering the provided data (presumably in zipped format). You should refer to Bloom taxonomy and types of education tasks  - User should be able to see the right answer on the question and/or a piece of context which was used to generate the question  - You should provide an ability for user to type his answer (or select from the options)  - user should get a final score on the test | - You can collect several open books / articles on generative learning (or on other topic), for example  - Look carefully at types of education tasks ([Learning assessment techniques.docx](https://docs.google.com/document/d/1YfIkYFjQlVch_sRcmdddhQsSBN2S1RnC/edit?usp=sharing&ouid=107288970882418970247&rtpof=true&sd=true)) and Bloom taxonomy (<https://en.wikipedia.org/wiki/Bloom%27s_taxonomy>)  - Refer to Ragas library and test set creation as an example (<https://docs.ragas.io/en/latest/getstarted/testset_generation.html>) - you are not required to implement optimization pipelines, but you can look at prompting there | **Simple pipeline:**  1. Parse and chunk your input data  2. Prompt LLM to choose “good” chunks  3. Prompt your LLM to generate question-answer pairs  4. Create a simple UI to show your results  **Advanced Pipeline:**  1. It is important to obtain a set of high quality questions  2. It will be very great if you’ll manage to add mechanisms for adaptation to individual learning rate (gradually increasing the task difficulty according to the student’s knowledge)  3. when a user made a mistake, the assistant may ask a pointing question (this question should not contain the direct answer to the initial question) to guide user to a correct answer |
| Visualize your docs | Create an assistant for historical docs processing. The assistant is responsible for identification of important dates and persons from the text and visualization of Events Timeline and relations between important persons.  **Tasks/preferred features:**  - Extract dates and persons and all related required information as a structured output.  - Dates should be accompanied with adequate and useful description  - Persons and relations between them should be accompanied with adequate and useful description  - Build data frames from extracted structured outputs  - Split the doc in parts if necessary.  - Use a function calling to generate visualizations with LIDA - Optionally: make visualizations interactive | - Wikipedia dataset  - You can also synthesize data | **Simple pipeline:**  1. Mine documents to extract important dates and persons  3. Rearrange them in correct time order  4. Use LIDA to generate control to draw data  **Advanced Pipeline:**  1. Make sure that you’ve extracted all the dates, add quality estimation pipeline - regenerate, or self-correct answers if needed  2. Fine-tune your model on real or synthetic examples if extraction quality is low |
| AI D&D game | Create a simple version of a text-based game driven by generative AI with elements of visual scene generation.  **Tasks/preferred features:**  - Your goal is to emulate a dungeon master who can also visualize an image of the current state of the game (location, npc, monster etc.)  - User should be able to freely write his current action and system can provide a textual response on the action and regenerate image if it should be changed | - may be implemented without dataset | **Simple pipeline:**  1. Prompt modell to write a basic game scenario (you can do it in one prompt, or, maybe you’ll need to do it iteratively)  2. Create a great prompt for your LLM model with clear instructions which role it has and how it should act  3. Provide scenario and role description to your visual language model (you can simplify task and do only text generation). Allow chat with user  **Advanced pipeline:**  1. Introduce game quality metrics that can be tracked during generation. Can you fix “bad” text and image generation? |
| Scenario to music | Create a GenAI-enabled service that can find appropriate soundtracks for scenes / situations described in the received scenario.  **Tasks/preferred features:**  - the service should analyze the scenario and find important scenes to be augmented with soundtracks.  - Soundtracks should be compiled into a database where the service can look for appropriate track  - the scenes should be matched with appropriate soundtracks from the database  - user may set additional restrictions and limitations for matching, e.g. only classic or not older than 2010 | - music datasets, e.g. <https://research.google.com/audioset/dataset/soundtrack_music.html> - in general you should create (or find) video/movie short description and music that is playing in this video | **Simple pipeline:**  1. Collect dataset  2. Play around with prompt engineering techniques to get descent audio generation results  3. Use the dataset to evaluate and/or finetune your audio language model  4. Let user send you the feedback and  **Advanced pipeline:**  1. Allow users to have some control over generation (set music genre / style / etc.)  2. Make regeneration pipeline based on user comments |
| The ultimate cheat sheet | Having a dataset compiled of cheat sheets on various topics, create a chat assistant that can support users by answering their questions using visually presented cheat sheets.  **Tasks/preferred features:**  - the chatbot should choose one or more appropriate cheat sheets to answer the question  - The chatbot should use VisualQA to answer questions using found cheat sheets | - dataset <https://www.kaggle.com/datasets/timoboz/data-science-cheat-sheets>  - you can collect a small set of cheat sheets, e.g. 1) <https://blogs.sas.com/content/subconsciousmusings/files/2017/04/machine-learning-cheet-sheet.png> 2) <https://learnsql.com/blog/sql-basics-cheat-sheet/sql-basics-cheat-sheet-a4.pdf?clckid=55c73d9f> | **Simple pipeline:**  - Collect dataset  - Check if visual language model understands the cheat sheet information correctly, use prompting techniques  - implement a multimodal rag pipeline to find the appropriate cheat sheets on the user’s question  **Advanced pipeline:**  - Ensure that answer that you provide to user is factually correct |
| Presentation Quality checker | Create a tool to evaluate the quality of slides deck. Important decisions in companies can be influenced by the right combination of facts, infographics and words!  **Tasks/preferred features:**  - User can load presentation in pptx / pdf format  - Each slide should be analyzed and described in detail, so user will be able to understand what should be fixed in order to satisfy the criteria  - System should give user some final quality score on the presentation | - available slide decks from <https://github.com/nttmdlab-nlp/SlideVQA>  -  - find presentation guidelines, e.g. (1) <https://slideworks.io/resources/how-mckinsey-consultants-make-presentations>, 2) <https://visualsculptors.com/structure-powerpointpresentation/>) | **Simple pipeline:**  - collect the dataset of different presentations  - Define a set of metrics to evaluate the quality of presentation  - Explain the metrics and criterias to visual language model to  **Advanced pipeline:**  - Add advanced metric on content and storytelling quality assessment  - Create pipelines to do fact checking of the information in presentation (if source link is presented) |