AI-Powered Summarization and “Interaction” with Academic Papers

Ruslan Popov

March 27, 2024

Contents

1 Personal information

1.1 Personal details

- Name: Ruslan Popov
- Email: ruslanpopov1512@gmail.com, popov_ro@ffeks.dnu.edu.ua.
- Country: Ukraine.
- Timezone: GMT+2 (Eastern European Standard Time).
- University: Oles Honchar Dnipro National University.
- Major: Computer Science.

1.2 Background

I have been interested in programming and everything connected to computers and technology from my childhood (specifically, when I got my personal computer). I've participated in many activities at school, and was a member of several tech-related clubs. I've also participated in physics and chemistry olympiads (grand competitions) and got prize-winning place (II on regional level for physics, I on regional level for chemistry, *didn't participate in country level because of COVID*). Eventually, I decided to major in computer science.

During the governmental examination I got the highest grade on the math exam. I got the highest evaluation mark for entering the university (just a weighted sum of exam grades). Currently, I am a freshman at university, studying computer science. I have no problems with studying and have completed many subjects ahead of time.

Mostly, I learn on my own. I constantly learn new programming languages and technologies and read books. I try to write small projects based on what I learned. Here are the examples of the most completed (*I have a lot of uncompleted but very interesting projects, and those are quite old*):

1. CHIP-8 emulator written in C++ language. I have additionally written an assembler and disassembler for the CHIP-8 processor, and even tried to write a small game (like breakout).
2. Lisp and Lox interpreters written in C++ language.
3. An interpreter for my own programming language: Loop. It has many features from well-known programming languages. *I've abandoned it, so recent commits may not work.*

Currently, I have lost interest from low-level programming and now I'm trying to fill the gaps in my knowledge from other fields of programming. I haven't found interesting projects for programming languages, or they were too complex for me.

I have experience writing academic papers, using LaTeX, BibLaTeX. I will tell only about one of my papers. I'm participating in an international science competition called Black Sea Science (*warning, HTTP only*). In this competition students submit their paper for evaluation. I have written and explained the inner workings of a program that could automatically solve physics word problems. Here is the citation and link:

Popov, R., & Karpenko, N. (2024). Automatic solving of physics word problems.

This paper won the second place in the first tour and will be published in journal: "Automation technological and business - processes". Currently, I'm waiting for the final results.

For now, I'm interested in the AI field. My program was using rule-based techniques, and I want to learn the modern machine learning algorithms. That's where this GSoC project picked my attention.

I have no previous contributions besides this project.

2 Preliminary information

2.1 Commitments during GSoC

I have a month of exams in my university in June. Well, I think I may be off several days. But the pressure in my university is so low, so this is absolutely not a problem.

3 Project proposal

3.1 Synopsis

This project seeks to make research more efficient by adding powerful AI tools to JabRef. New features are:

- "Interaction" with papers (question answering mostly).
- Connection to different AI providers.
- Getting summaries of research papers.
- Interaction with library groups (QA and summarization).
- Support of local LLMs.
- Smart search for articles.
- Integration with SchoolarAI.

3.2 Related Work

Related work in reference managers:

- Zotero: currently it has no plans to integrate AI technologies. But there are several plugins:
  - kazgu/zotero-chatgpt: supports only interaction with one document.
  - MuiseDestiny/zotero-gp: seems like an interesting project, but it lacks documentation and tutorials. Chinese is extensively used, with practically no English support.
  - lifan0127/ai-research-assistant: has many interesting features. However, it relies on OpenAI's ChatGPT 4.
- Mendeley: no AI integration.
- Paperpile: no AI integration.
- EndNote: no AI integration.
- Citavi: no AI integration.

Related work in research assistants:

- Explainpaper: service that gives you ability to interact with papers. Though, it supports only one document and relies on OpenAI's ChatGPT. Moreover, the free version is very limited.

- Scispace: searches for articles and provides summaries. Let's you explain parts of papers. Again, it's fully online.
- AskYourPDF: another program, similar to others.

There are many programs like AskYourPDFs, but they provide the same set of features.

3.3 Benefits to Community

Our app offers valuable features for researchers. It provides summarized analyses of research papers, saving time and effort. Enhanced literature discovery ensures staying updated with the latest research, while seamless integration with multiple AI providers offers flexibility and access to cutting-edge technology. These features streamline workflow, save time, and deepen understanding, accelerating research and aiding informed decisions and discoveries.

Our program offers distinct advantages: integration with diverse AI providers, support for local LLMs, and extensibility. It provides access to a variety of tools, improves accuracy with tailored language models, and easily adapts to future needs.

3.4 Some work that I have done

In order to investigate the source code of JabRef and prove the ability to do programming I've tried to resolve several issues:

- Support .lnk files for TeXworks #11065. *Merged!*
- Fix html content handling when fetching IEEE papers #11091. *Co-author.*

I want to thank you for this opportunity. It's my first (*very small*) contribution to open source, and I still need to learn a lot.

Also, to understand the available instruments for making AI apps, I've made a prototype app in which users can interact with PDF files (either a single file or directory with several files). This was a very important experience that showed the challenges for making such an app.

While experimenting with `langchain4j` library, I came across a bug, I have made an issue and even solved it.

3.5 Comments on timeline

- At the end of each week we will have a complete feature and a new version of JabRef.

- There are two kinds of tests: basic tests which involve UI functionality, storing functionality, connection tests, etc. The other type of tests are LLM tests. We can't automatize them, so a human check is needed. And also an API key is needed for testing. In each week there is a sub-goal "Test" that means the first types of testing.
- In each week there are a sub-goal "Receive and process the feedback". It usually means "Receive and process the feedback of previous feature". So that we work in parallel on new feature and improving the previous. We want to make this project "iterative", meaning the users will have access to all new features right in the code period, not just at the end of GSoC.

3.6 Timeline

3.6.1 Week 1 – Chat with one document

Goals:

- Make UI tab
- Establish a connection to LLM
- Experiment with embedding model and prompt templates
- Test the feature
- Document the feature
- Receive and process the feedback

Results of this week: the first addition to JabRef! Users will be able to interact with one library entry (if it has a full-text attached).

UI changes or additions:

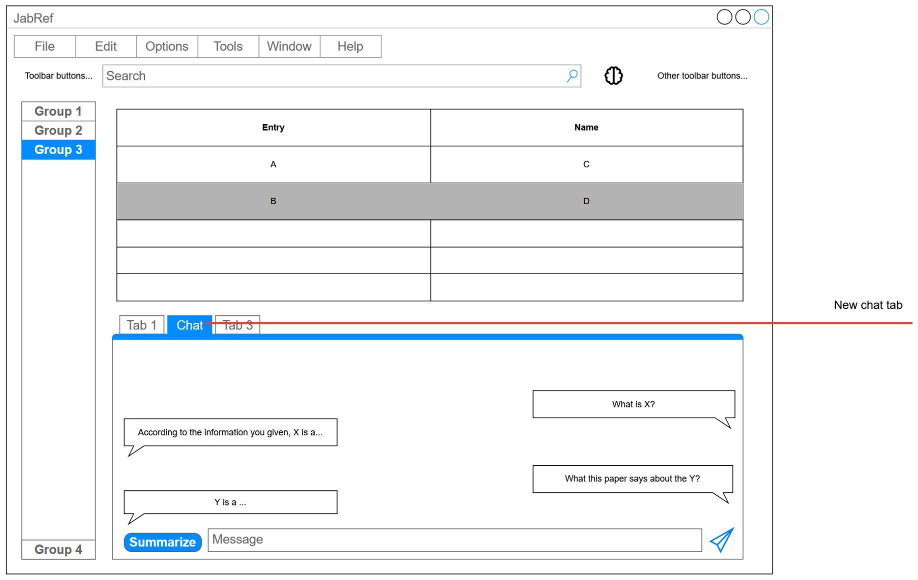


Figure 1: New UI - chat with entry (new tab)

Implementation details: we need to make a new UI components, then integrate the `langchain4j` library.

Possible challenges: while I've experimented with `langchain4j`, I don't have experience with JavaFX. But I hope I will learn along the way.

Useful links:

- `langchain4j` library repo.
- Chat with documents example.

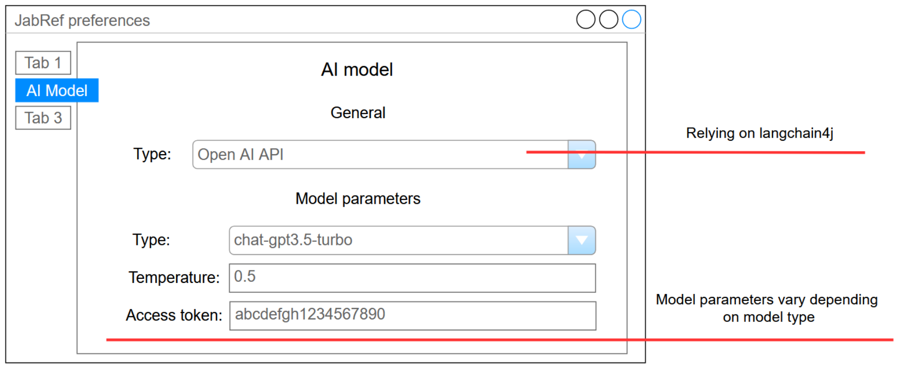
3.6.2 Week 2 – Different AI providers

Goals:

- Make preferences model
- Make a tab in settings window
- Test the feature
- Document the feature
- Receive and process the feedback

Results of this week: users will have the ability to use the AI provider they have (have paid) or they like the most. This feature also gives the ability to experiment and evaluate effectiveness of different AI models.

UI changes or additions:



Type: Open AI API

Type: chat-gpt3.5-turbo

Temperature: 0.5

Access token: abcdefgh1234567890

Figure 2: New UI - new tab in preferences

Implementation details: `langchain4j` provides many integration with modern AI providers, so we can easily reuse them.

Possible challenges: probably no. But I only have some money on OpenAI API, so I can't test others. Also, Google AI APIs are not available in Ukraine.

Useful links:

- List of `langchain4j` integrations.

3.6.3 Week 3 – Summarize one document

Goals:

- Make UI button
- Implement summarization algorithms
- Experiment with algorithms parameters and prompt templates
- Test the feature
- Document the feature
- Receive and process the feedback

Results of this week: users can now ask AI to summarize documents.

UI changes or additions: add button "Summarize" next to user message field.

Implementation details: summarization should be implemented separately from QA because they use different algorithms. If the user will type "Can you make a brief summary of the document, please?" in message field then it will have very bad consequences (test in the prototype app I made).

Possible challenges: `langchain4j` doesn't have algorithms for summarization, and there is no library for Java with those algorithms. We have to rewrite them in Java. That's not a problem because this topic is actively studied and is implemented in parent Python library `langchain` (which `langchain4j` is derived from).

Useful links:

- Summarization methods 1 (Google).
- Summarization methods 2.
- Summarization in `langchain` (Python).

3.6.4 Week 4 – Chat with a group of documents

Goals:

- Make UI window for chat with groups
- Instruct LLM to add references where it has taken information
- Test the feature
- Document the feature
- Receive and process the feedback

Results of this week: a very useful feature for QA over several documents. If we even provide links to entries in the AI response that would be super cool.

UI changes or additions:

In the first draft of this proposal, I have made it as a button that allows to talk with the library. However, we have changed it to talk not to the library, but to a group of entries. We can implement this either as a button or a menu entry

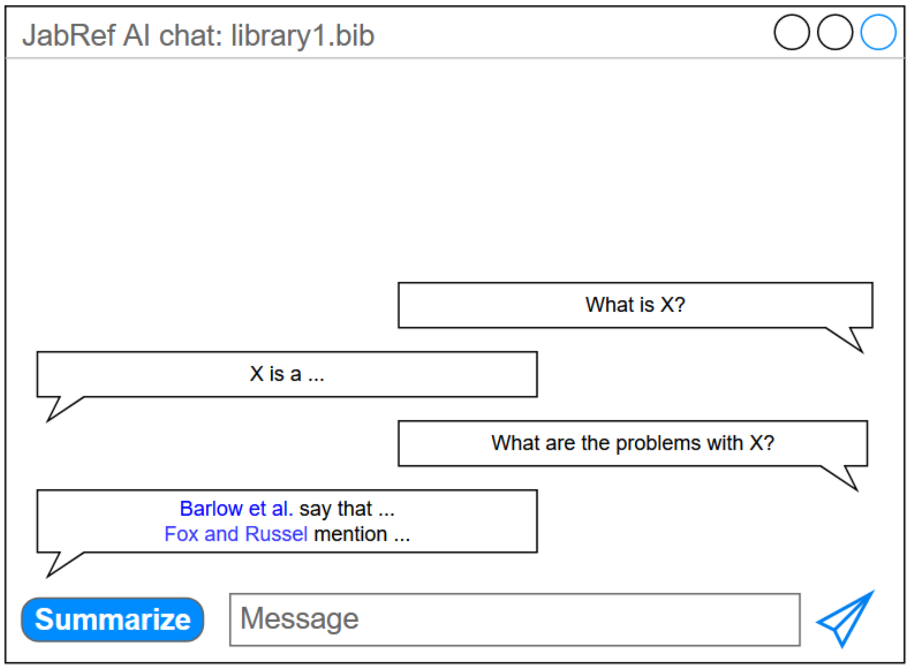

JabRef AI chat: library1.bib
What is X?
X is a ...
What are the problems with X?
Barlow et al. say that ...
Fox and Russel mention ...
Summarize
Message

Figure 3: New UI - chat with library window

Implementation details: the `langchain4j` tutorial for "Chat with documents" has all the code we need. There are practically no changes compared with a single document.

Possible challenges: the biggest challenge here is to make LLM add citations from which it has taken the relevant information for the question. `langchain4j` doesn't have such feature. We can also implement it on another week, if it turns out too hard. *But I would really like to have this feature in JabRef. Does any service has a feature like this?*

3.6.5 Week 5 – Summarize a group of documents

Goals:

- Make UI button
- Reuse or expand the available summarization algorithms
- Test the feature
- Document the feature

- Receive and process the feedback

Results of this week: a brief summary of the topic of the library.

UI changes or additions: just a button in chat window.

Implementation details: there should not be much a problem if we reuse the algorithms for summarization of one paper.

Possible challenges: maybe there are better ways to summarize several documents, so a little investigation is needed. And again `langchain4j` doesn't have those algorithms.

3.6.6 Milestone 1 – First integration with remote LLMs done

Now user can:

- Interact with documents
- Summarize documents
- Use their favorite AI providers

3.6.7 Week 6 – Local LLM provider

Goals:

- Investigate the available libraries or methods for running a local LLM
- Add the Ollama server variant in model preferences tab
- Write clear and simple tutorials about how to start and use Ollama
- Test the feature
- Document the feature
- Receive and process the feedback

Results of this week: users an use LLM safely, confidentially without relying on third-party services.

UI changes or additions: add a variant in model preferences tab.

Implementation details: there are two main libraries for running local LLMs: LocalAI and Ollama. They act as a server that have compatible API as in OpenAI API. Moreover, they have integration with `langchain4j`. LocalAI can support many models from Hugging Face in `.gguf` and `.ggml` formats. Also, it has more features than Ollama (text to audio, text to image, etc.). For now it's much simpler to use Ollama. It has simpler configuration and supports only

LLM. They also have a curated list of LLM models, so users don't have to worry too much about choices.

Possible challenges: I hope there will be no problems with running LLMs.

Useful links:

- LocalAI.
- Ollama.
- Ollama LLM support.

3.6.8 Week 7 – Article search

Goals:

- Investigate the available libraries or frameworks
- Make UI for article search
- Test the feature
- Document the feature
- Receive and process the feedback

Results of this week: the user can provide an LLM with a query and then the LLM will find the relevant articles.

UI changes or additions:

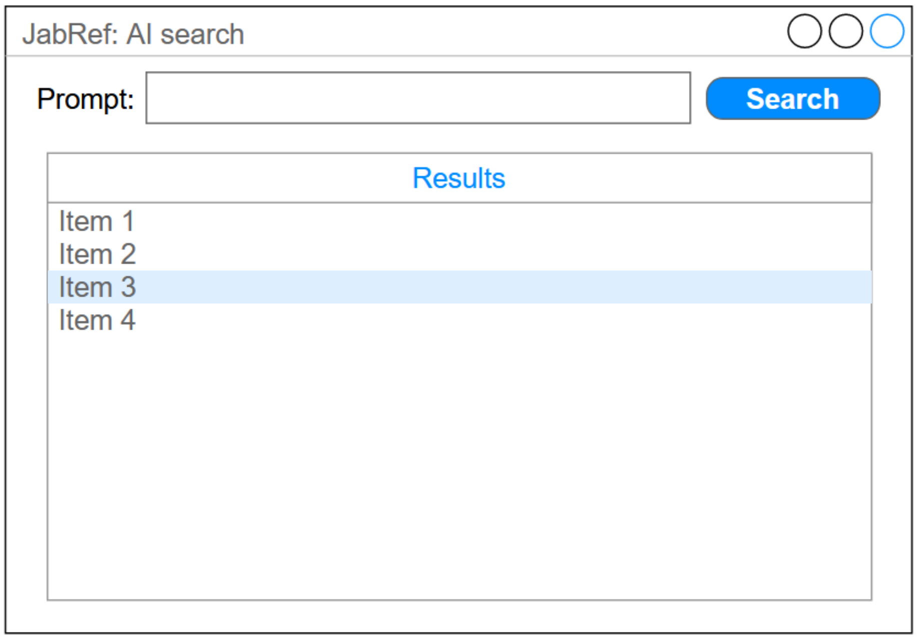


Figure 4: New UI - window for article search

Implementation details: we need to investigate (*I've probably used this word too many times*) the available libraries for this task.

3.6.9 Milestone 2 – First integration with local LLMs done

Now user can:

- Use a local LLM model while interacting with documents
- Use smart search for papers with AI help.

3.6.10 Week 8 – ScholarAI integration

Goals:

- Investigate methods for integration, retrieving information, authorizing, etc.
- Synchronization with ScholarAI (PDFs and library entries)
- Chat with ScholarAI

- Test the feature
- Document the feature
- Receive and process the feedback

Results of this week: integration with ScholarAI.

UI changes or additions: may include some window for chat and buttons (or menu entries) for synchronization. Also a window for login.

Possible challenges: SchoolarAI doesn't have a public API.

3.6.11 Week 9 – Kernel Memory

Goals:

- Investigate features provided with Kernel Memory
- Experiment with and integrate Kernel Memory to JabRef.
- Test the feature
- Document the feature
- Receive and process the feedback

Results of this week: a more powerful AI integration.

Useful links:

- Kernel Memory GitHub.

3.6.12 Milestone 3 – Integration with local and remote LLMs done

Now user can:

- Work with ScholarAI.
- Enhance their work with Kernel Memory services.

3.6.13 Week 10 – Semantic Kernel

Goals:

- Investigate features provided with Semantic Kernel
- Experiment with and integrate Semantic Kernel to JabRef.
- Test the feature

- Document the feature
- Receive and process the feedback

Results of this week: a more powerful AI integration.

Useful links:

- Semantic Kernel GitHub.
- Semantic Kernel documentation.

3.6.14 Week 11 – LLM tests

Goals:

- Collect a library of documents
- Make questions for these documents
- Test different LLMs for QA
- Test different LLMs for summarization
- Check the speed and cost for this
- Additionally include local LLMs for testing

Results of this week: a research on effectiveness of different LLMs for QA and summarization in academic papers.

UI changes or additions: no.

Implementation details: we need to somehow automatize this, maybe make a script that generates a table. With the help of Oliver Kopp we have a dataset of articles from TUGboat.

Possible challenges: we need to spend some time to make the questions and evaluate the answers. Also, it will take some time for the LLM to produce answer. Moreover we should be careful, so that it won't cost too much. (*OpenAI says "pay as you go", and the price for a token is not small, so I hope this won't be a problem*).

Useful links:

- TUGboat mirror.

3.6.15 Week 12 – Final polishing

Includes:

- Writing documentation
- Writing tutorial
- Writing blog posts
- Fixing bugs

3.6.16 Milestone 4 – Alternatives evaluated and documented. User-facing documentation done.

- The final milestone that represents the whole work. We successfully made the interaction with papers, AI providers integration, smart article search, and more.
- We also conducted testing of different LLM models which is a very useful result.
- Users will have nice and clear tutorials about how to use the new AI technologies the best way.