Heidelberg University NLP with Transformers

Data Science Group Winter Semester 2023/24

Prof. Dr. Michael Gertz

Ashish Chouhan, Satya Almasian, John Ziegler, Jayson Salazar, Nicolas Reuter

Group Project

General Notes

- The project is a team effort of **three to four** students; the same students who also work together on the assignments. You are not allowed to share project details among groups. You can ask general questions to the Moodle forum, but you are not allowed to share code, curated data etc. We will check completed projects for overlap, and if we detect a case of copy & paste, depending on the degree of overlap, we lower the grade for your project, or - in the worst case - assign a not passed for both your project and the one you copied from. If you have any questions regarding possible plagiarism issues, please get in contact with your project mentor.
- Each group will be assigned a project mentor in the week of October 30, 2023. The mentors will get in contact with their groups for further details. For further questions, please reach out via the Moodle forum, or contact your mentor or instructor.
- **The first milestone of the project has to be completed by January 11, 2024. The project report and final code has to be completed by March 4, 2024, at 2pm.**
- We do not offer any computing resources. We recommend that you use open-source or cloud-based systems. Google Colab is a good option for any computations that require GPU.
- Your final system should be functional, practical, and easy to interact with. It is **not** that important if you do not use all of the state-of-the-art and cutting-edge technology everywhere in your system.
- Over the course of the group project phase, we will post further information and details via the Moodle forum.

---

1. Project Topic and Data

In this project, you have to build a domain-specific question-answering (QA) system. As document corpus to be queries, you need to choose between

- the medical domain, more precisely, articles accessible via the PubMed portal (https://pubmed.ncbi.nlm.nih.gov/) or
- the legal domain, that is, articles from the EUR-Lex platform (https://eur-lex.europa.eu/).

Your system takes a question in natural language as input, e.g., “What are the requirements for refrigerated storage cabinets?” and should generate an answer in natural language by determining relevant documents from your corpus. An answer then might look like this: “The key requirements include Energy efficiency, Product labeling, and Technical documentation (extracted from Regulation (EU) 2015/1095)”. For building the QA system, we recommend the following workflow, which you can adjust to your needs.

Data Acquisition

The project starts with acquiring the relevant data. Choose the domain you want to work on and collect the data needed for the project from the respective source.

- **PubMed:** Abstracts of the articles published between the years 2013 to 2023 that contain the word “intelligence” in the abstract’s text.
- **EUR-Lex:** English articles present in different file formats for the “energy” domain.

Choose the data storage method carefully and in accordance with the retrieval method you want to apply.

Text Retrieval

In general, questions against a document corpus can be treated as queries against a document corpus, and based on these queries, potential candidates (documents or document fragments) are retrieved from the corpus for answer generation. Based on the final architecture your group develops, your system has to include an information retrieval (IR) component, which should at least include “free form queries” but also might include faceted search (e.g., for selecting documents published in a particular time range). Retrieval acts as a pre-filtering stage to retrieve relevant text data for the answer generation component. In this stage, it is important to identify which part of the questions should be used in what way for retrieval. For example, does the query contain named entities that should be treated in a special way? How do you handle fully specified questions that might contain stop words etc.? How to fully support both semantic search and lexicographical search?

Forms of Questions and Answers

Questions can come in different forms. Below are some common types of questions with examples. Your system should be able to generate answers to one or more of these question types. The more types your system can handle, the better.

- **Confirmation Questions [yes or no]:** Yes/No questions require an understanding of a given context and deciding a boolean value for an answer, e.g., “Is Paris the capital of France?”.
- **Factoid-type Questions [what, which, when, who, how]:** These usually begin with a “wh”-word. An answer then is commonly short and formulated as a single sentence. In some cases, returning a snippet of a document’s text already answers the question, e.g., “What is the capital of France?”, where a sentence from Wikipedia answers the question.
- **List-type Questions:** The answer is a list of items, e.g.,“Which cities have served as the capital of France throughout its history?”. Answering such questions rarely

requires any answering generation if the exact list is stored in some document in the corpus already.

- **Causal Questions [why or how]:** Causal questions seek reasons, explanations, and elaborations on particular objects or events, e.g., “Why did Paris become the capital of France?” Causal questions have descriptive answers that can range from a few sentences to whole paragraphs.
- **Hypothetical Questions:** These questions describe a hypothetical scenario and usually start with “what would happen if”, e.g., “What would happen if Paris airport closes for a day?”. The reliability and accuracy of answers to these questions are typically low in most application settings.
- **Complex Questions:** Sometimes a question requires multi-part reasoning by understanding the semantics of multiple text snippets to generate a correct answer, e.g., “What cultural and historical factors contributed to the development of the Louvre Museum as a world-renowned art institution?”, which requires inferring information from multiple documents to generate an answer.

User Interface

You are also encouraged to build a user-friendly interface to interact with your system. It can be a website, mobile app, or desktop application. In its simplest form, the command line interface acts as a question-answering chatbot. While designing your interface, focus on usability and convenience; templates and coloring are less important. Would it be useful to point the user to the excerpt in the text where the answer was generated? Would it be useful to provide follow-up questions or clarification for user engagement? How easy is it for the user to specify further constraints on a search, e.g., in the form of a faceted search.

2. Evaluation

An important objective of the project ist that you become familiar with a real-world problem scenario, where frequently a common benchmark dataset (ground truth) and complete framework for an extensive evaluation is not present. Such cases are rarely considered in academia, and typically only frameworks are adopted where proper evaluation datasets are publicly available. Nonetheless, in the real world, evaluation methods need to be tailored to specific use cases or to unfamiliar domains that do not have publicly available testdata.
For this project, you are required to evaluate your system and show the effectiveness of your approach by coming up with novel and creative ways for evaluation. Some suggestions are:

- **Annotate Data:** Create a small set of questions and answers from your dataset and use these for evaluation.
- **Automatically Generate Test Data:** Use an “oracle system” to generate question and answer pairs and treat those as ground truth.
- **Experimental Evaluation:** Check the sanity and coherence of your outputs by looking at some examples from the system and documenting the performance.

Your evaluation setup is not only established to prove the superiority of your system but also to determine its shortcomings and weaknesses. This also includes tests for edge cases, where data is scarcely available, typos, and other sorts of perturbations of the input.

3. First Milestone

Your mentor will set up a meeting with you regarding the first milestone, which is due on January 11, 2024. In this meeting, you will be asked about the progress of your project. By this time you should have acquired the data necessary and set up a search system to find relevant documents for a given question or, more generally, query. Keep in mind the following points for your first milestone:

- **Come prepared:** Discuss among the teammates how you plan to present your progress to the mentor. You can use slides, show a functioning prototype system, or describe the progress in words.
- **Engage in the meeting:** All teammates are required to attend the meeting, answer questions, and show engagement. Avoid delegating the task of speaking to a single person.
- **Ask questions:** The first milestone is also an opportunity for you to ask questions and clarification on future steps (well, you are always allowed to ask your mentor questions). Don't hesitate to ask for feedback. The best course of action is to send some questions to your mentor before the meeting, so we can provide a well-thought-through answer.

4. Project Report

The final project report for this class should detail what you have achieved and show the results of your work. The report needs to be submitted as a **DOCUMENTATION.md** markdown file to the GitHub repository of your project. It should be written in a way that a fellow classmate can understand your report, so you cannot assume any knowledge about your specific application, but the reader should be familiar with the basic concepts you learned during class. The following sections might already give you a rough structure for the report, but this is not set in stone and can be altered if it serves your needs. Keep the report short, concise, and to the point. This is not a seminar report that extends over 16-20 pages when printed (using 11pt or 12pt font).

Key Information

Your report needs to give the following information:

- **Title:** The final title of your project.
- **Team Members:** The name of all team members. The participating students must match a group that is handing in assignments together. Please add your matriculation number and course of study as well.
- **Email Addresses:** The preferred email of each team member.

- **Member Contribution**: We also expect you to detail what each member did for the project. Each team member needs to contribute some text in which they describe what their contribution was, and which challenges they faced. If you wish, you can also provide a table like structure detailing the contributions to the different parts of your project.
- **Advisor:** Give the name of your advisor.
- **Anti-plagiarism Confirmation:** A short affirmation that you did the project's work on your own as already indicated in the template. We will provide you with such a statement at the beginning of next year.

Introduction
The first section should explain what your project is all about, why the project and underlying problems are interesting and relevant, and it should outline the rest of your report. The introduction also outlines the key ideas of your approach without going into the details in terms of realization and implementation. It is also possible to give an outlook of the results the reader can expect to see in the later sections.

Related Work
This section of your report covers core prior work related to your project, and puts your work into the context of current research. This includes papers that you used as a basis for your approach or papers that used the same techniques as you did but applied them to a different problem. In this section, you should emphasize how your work differs from previous work, e.g., by outlining the limitations of previous work, or why your application domain is different from those other researchers have already investigated. However, you should not go into all the details about any of the work you cite, instead, you should outline how the major points of previous work relate to your project.

Methods/Approach
This section gives all the conceptual details of the system you developed and is intended for readers who want to fully understand your system, its functionality, and its components. This includes the (data) processing pipelines you realized, the algorithms you developed, and all other key methods you designed during the project.Here's some advice for this section:.

- Be specific about your methods. In this section, you can show your detailed understanding of the matter. Include equations where necessary and show figures, such that the reader can better follow and understand your approach.
- If some part of your work is original or especially creative, make it clear, such that the reader (e.g., your mentor) can understand the novelty of your approach. You do not need to show any false modesty in this part of your work. If you use any existing methods, however, you need to recognize and properly cite them. Be consistent throughout your report and stick to a fixed vocabulary before writing. This includes mathematical terms and notations, method names, names for data sets, and the like. Only a few things are more irritating than changing naming and writing conventions during a report.

- If you have any baseline approaches, they should also be described. This can be detailed if you created the baseline yourself or just briefly if there is an external source from which you took the baseline.

Experimental Setup and Results

This section needs to cover the following information:

- **Data:** Describe the data you used for the project. Outline where, when, and how you have gathered the dataset. Providing a clear understanding of the data can be very helpful for readers to follow your analysis. If you have any interesting or insightful metrics of your data, this is the place to show them.
- **Evaluation Method:** Be clear about how you are going to evaluate your project. Do you use an existing quantitative metric? Or did you define your evaluation metric? If you use your method, be sure to motivate what you try to achieve with it, and what it does reflect. In any case, define your evaluation metric, be it a qualitative or quantitative one, automatic or human-defined.
- **Experimental Details:** If your methods rely on configurable parameters, specify them, such that your results are replicable. Explain why you chose the parameters in that way (e.g., did you do some grid search?).
- **Results:** The results section of your report highly depends on your project. If you have a baseline to which you can compare your approach, you obviously should do this. In such cases, tables and plots are suitable means to present your achievements. In any case, use the evaluation metrics defined before. Moreover, comment on your results: are they in line with your expectations? Or are they better or worse? What are the potential reasons for this?
- **Analysis:** You can include some qualitative analysis in this section. Does your system work as expected? Are there cases in which your approach consistently succeeds in the task, or surprisingly fails? If you have a baseline, you can also compare if they succeed and fail in the same contexts or if your approach may be suitable for other applications than the baseline. Underpin your points using examples or metrics instead of stating unproven claims**.**

Conclusion and Future Work

Recap the main contributions of your project. Highlight your achievements, but also reflect on the limitations of your work. Potentially, you can also outline how your model can be extended or improved in the future, or you can briefly give insights into what you have learned during the project.

References

List your references at the end of your report. You can add them as a list at the end of your report and refer to the author's name and year in the text, e.g., [Gertz et al. 2020](#link-to-the-bib-section). Make sure you follow proper citation methods and include all author's names, the name of the journal or conference, the full title of the paper, and the publication year.

5. Code

The source code needs to be made available via a **private** GitHub. Create a repo with a meaningful name of the pattern {project-name}-INLPT-WS2023 and all the members. You should ensure that your project advisor is set as a collaborator. This is a reminder that we can only evaluate projects we have access to.
We will require a certain level of code quality and documentation within your project. There is no way to expect a fully polished project according to industry-level standards, but we want to see clear efforts to make the code maintainable and understandable from an external perspective. Specifically, we point out the following tips to make your code more readable:

- **Detailed README.md file:** Describe the installation guide and any additional data sources needed to successfully run the project in the README.md file. Make sure you describe the code structure and if sequential runs are required to produce an output, the steps should be explained. A download link to the data used in the project or description to create the same dataset should be included.
- **Include member data:** The README.md file should start with the name, matriculation number, GitHub username, and email addresses of all teammates. Include the name of your advisor as soon as you are assigned one.
- **Dependencies:** Provide a list of all required libraries to successfully run your system. The dependencies of your systems should be clearly defined and easily reproducible. Take advantage of a requirement.txt file or conda environments to make sure one can run your code with minimal effort.
- **Self-explanatory variables:** Most importantly, make use of sensible naming conventions for variables.
- **Comments:** In places where self-explanatory variables cannot fully describe what is happening anymore, comments should be used to concisely (!) state the functionality of your code.
- **Docstrings:** Similar to comments, docstrings make it easier to understand the workings of an entire class/function/module. Parameters can be explained in more detail as well, including their expected type.
- **Module structure**: A modular code base is not only easier to understand but easier to maintain as well. Especially when working in parallel with multiple people on the same code base, the modularity of your project allows you to separate intents and divide up work sensibly. Try to structure the code so that the individual modules of your system can be found quickly and intuitively in the repository.
- **Code consistency:** Ideally, you define and adhere to coding guidelines from the beginning of the project, and configure git hooks (or your local IDE) to check for style consistency. If you haven't read it, PEP-8 is the go-to style guide for Python code.
- **Indicate "Hacks":** Keywords like TODO or FIXME allow you to highlight a special type of comment that draws the attention of your future self (and us instructors) to relevant passages that may need more attention in the future, or are otherwise in a fragile state. Many IDEs support this via specifically colored comment lines.

- **Existing code fragments**: In case you already know that you will be using a third-party or prior project, you must indicate those parts of your project (respective parts, of course, should also be mentioned in the report).
- **Code clean up:** At the end of your project, you should merge all your contributions into the main branch and delete all other (unnecessary) branches. When reviewing the code, your supervisor will only look at the main branch. Also, make sure to remove old, unused, or duplicate files **(i.e., do a garbage collection)**

6. Meetings

Create a folder named **Meetings** in your GitHub repo to keep meeting notes. You should document any meetings, discussions, additional reading material, and communications with your mentor in a markdown file in this folder. Notes should include the date and subject of the meeting. For example, "28th November 2023: Pair programming". Include the members who were present in the discussion and the important points discussed. Your mentor will regularly check this folder to assess the progress of your project. For meeting with a mentor, contact the assigned mentor directly via email at {lastname}@informatik.uni-heidelberg.de.

7. Grading

We will evaluate the **last commit before the deadline** in your GitHub repository. Your project report will be graded by taking into account multiple criteria. This includes primarily the originality or creativity of your approach, the performance of your methods, your scientific writing, how detailed you presented your results, your in-depth understanding of the matter, etc. Note that each student gets an individual grade, i.e., in cases where team members contributed unequally to the project, grades can differ between members. Several criteria to keep in mind:

- **Code state:** The project should exhibit a minimal level of code quality.
- **Responsiveness to feedback**: Did you take into consideration feedback from your mentor? Did new material from the class influence your design choices (for relevant parts of your project)?
- **Workload distribution:** Everyone should contribute to the project and contribution should be reflected in the commit history and meeting journal. Commit histories offer a glimpse at the distribution of the workload. We neither want a solo project state (i.e., only a single person doing everything), nor slackers (i.e., members without any clear activity). You may indicate pair programming efforts or contributions that are otherwise not clear from the git history in the meeting journal.
- **Grade distribution:** Milestone (10%), project report and evaluation (40%), and code and running system (50%).
- **Creativity:** An additional bonus is given for creative approaches. Did you find a smart way to acquire new samples or test data for your project? Is there domain knowledge that you can incorporate? Did you make a user-friendly interface?