Heidelberg University
Database Systems Research Group
Prof. Dr. Michael Gertz
Satya Almasian, Dennis Aumiller

November 02, 2020
Text Analytics
Winter Semester 2020/21

Milestone Guidelines
Milestone due on Friday, December, 18, 2020, 18:00, via GitHub

---

General Notes:

- The milestone is graded as part of the final code and report category, which (in total) makes up for 70% of your grade for the group project.
- **We will evaluate the last commit before the deadline in your GitHub repository**. Your milestone report should be realized **in the form of a `README.md` file** in your repository.
- You should have made sure that instructors are set as collaborators in the project by now. This is a reminder that we can only evaluate projects we have access to.
- For further questions, please reach out via the Moodle forums or contact an instructor.

---

1 General Submission Guidelines

State the following information **at the top** of your `README.md` file:

- **Title**: Title of your project (does not need to be final).
- **Team Members**: The name of all team members. The students must match a group handing in assignments and project proposal.
- **Mail Addresses**: The preferred email of each team member.
- **Existing Code Fragments**: In case you are choosing a project for which there exists code that you are using, you must clearly indicate those parts of your project and link to them.
- **Utilized libraries**: Provide a list of all required libraries to successfully run your system. Ideally, this comes in the form of a separate `requirements.txt` file.
- **Contributions**: Commit histories offer us a glimpse at the distribution of 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 that case, simply add the activity and participating member(s) of your team to the report in a separate section.
- **Uploading for other team members**: If you are pushing code for another team member, you must indicate their name in the commit message.

---

2 Project State

In the write-up, you should detail the current state of your project. This should both reflect on the status compared to the intended milestone achievements that you detailed in your proposal report, and simultaneously take this information to form an updated timeline for the remainder of the project. Specifically, we expect you to address the following points in the report:

- **Planning State**: As previously mentioned, briefly address the current state of the project, including tasks that you were not able to finish (yet).
- **Future Planning**: Detail a brief timeline including tasks that you want to complete in the second part of the project. These suggestions might come through new insights from the data/related work, input from your mentors, etc.
- **High-level Architecture Description**: Highlight different parts of your code project to give an initial understanding of your module structure. Keep in mind to also detail your processing pipeline (tokenization, lemmatization/stemming, NER, etc.).
- **Data Analysis**: See separate section.
- **Experiments**: In case that you already have some results from initial experiments, you may detail the results and implications. We strongly encourage you to already provide simple baselines.

---

3 Data Analysis

- **Data Sources**: Re-state all data sources that you are using in your acquired data collection.
- **Preprocessing**: Detail any preprocessing steps you have taken to ensure proper data quality. This can include unicode normalization, length normalization, text sanitizing, etc.
- **Basic Statistics**: List basic information, such as, number of samples, mean, median & standard deviation, etc. In the case of a classification problem, detail the class distribution. You may utilize plots similar to ones presented during the tutorials to support your description.
- **Examples**: Give at least one example of a data sample from your collection. You can additionally provide edge cases where additional assumptions about the data are necessary.

---

4 Current Code State

We will evaluate not only the write-up of your project state, but also consider a certain level of code quality and documentation within your project. Obviously, 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:

- **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, modularity of your project allows you to separate intents and divide up work sensibly.
- **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.

---

5 Grading Criteria

For grading, we will be looking at several criteria in your project state, among others:

- **Progress**: Are there clear attempts to realize the projected features from the proposal? We obviously do not expect that you fully implemented everything you were planning to, since timelines usually change, but nonetheless expect a certain progress towards the final product.
- **Reflection**: Based on the current state, we evaluate adjustments to future time schedules, including the potential shift in projected final states.
- **Code State**: As mentioned before, the project should exhibit a minimal level of code quality.
- **Workload Distribution**: See general submission guideline section.
- **Responsiveness to Feedback**: Did you take into consideration feedback from your mentor? Did new material from the class influence design choices (for relevant parts of your project)?
