# Cal Poly Personal Assistant

Schuyler Fenton, Dani Lopez, Sarah Ellwein

## I. Division of Labor

The data sustainer was created by all three members, with each member focusing on parsing a different page category (course schedule, office hour table, course catalog). The main project script functions were, in contrast, worked on in various capacities by the whole team, with functions headed by those who were most involved. The query tokenizer was led by Dani, the query matcher by Sarah, and the output formatter by Schuyler.

## II. Data Sources

First, we develop a data sustainer to fetch and organize our needed data. In creating the data sustainer, we chose to use Requests and BeautifulSoups to pull and scrape the data from our available websites (refer to Appendix), then used MySQL to organize our data into databases. Our database schema is organized as shown in Figure 1 along with an ER diagram in Figure 2. We chose to organize our data in this manner in order to maintain normalization.

| |
|---|
| **course_info**(prefix*, coursenum*, course, descr) |
| **facilities**(buildname, buildnum*) |
| **faculty**(alias*, lastname, firstname, buildnum, dept, title, phone) |
| **term**(termid*, quarter, year) |
| **course_sections**(termid*, prefix*, coursenum*, section*, type, days, starttime, endtime, alias, buildnum, roomnum, wait, enrolled) |
| **office_hours**(alias*, days*, starttime*, endtime*, buildnum*, room*) |

**Figure 1.** Database Schemas (* indicates primary key)

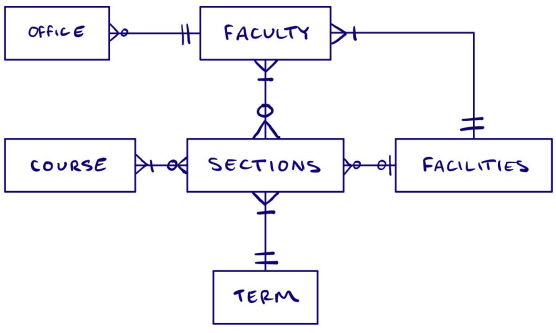


**Figure 2.** ER diagram

## III. Strategy Explanation and Road Blocks

Our strategy for this project followed the flow of Figure . After completing the data sustainer, we tokenized the user input. During tokenization, punctuation and stop words were removed, except for designated stopwords that we used as indicators, such as what, when, and where. Additionally, we made sure to remove any characters after apostrophes. As an example, let's use "When are Andrew Migler's office hours?". After tokenization, this would be ["when", "andrew", "migler", "office", "hours"]. This tokenized user input is then tagged on multiple different conditions.

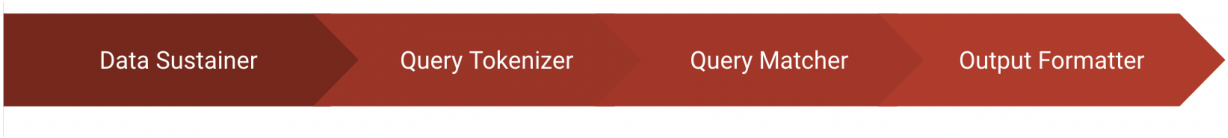


**Figure 3.** State Diagram after User Input

The tagging function uses keywords, syntax indicators, when appropriate, and cross references database information, to replace words with the appropriate tags. Two examples of these are quarters and professor names. Given the quarters are always the same (Fall, Winter, Spring, Summer), we easily matched words from the user input to those words. If there is a match, we can replace the word with the tag [QUARTER]. That's an example of how we used keywords. Then, we used that as a possible syntax indicator. We check if the number following it is a year, in case they are looking for prior years. Otherwise, we can assume they are indicating the current year. As for when we cross reference with the database, we can use professor names as an example. With the earlier example, ["when", "andrew", "migler", "office", "hours"], we check if any of the words exist in the faculty table. Since "andrew" and "migler" exist in the table, the string in which we duplicated the input would then become 'When [FACULTY] office hours' because we account for Andrew Migler being one name. This string with the labels is later used for our query matcher. Additionally however, this same function also saves the information that was replaced with the tag for the purpose of SQL queries. As an example, for "Who teaches CSC 101?", we would have {["[PREFIX]", "CSC"], ["[COURSENUM]", "101"]}.

With the input tokenized and labeled, we used that for our query matcher, which we used the Naive-Bayes approach for. This was a chosen strategy for a few reasons, starting with the fact that it was simpler, but also more intuitive to us. Additionally, it is simpler and faster than other algorithms. However, upon implementation, we recognized that our data was not accurate. This was due to the fact that for certain questions, there are multiple ways to ask that question, while others only have a couple ways that aren't awkward. Thus, the questions that have more ways of being asked increase the frequency of certain words, resulting in a higher importance. So in our model, queries that can be asked in less ways are given less importance than those that can be asked in multiple ways. With more time, we would have liked to implement other methods, such as SVM and decision trees, and compare how they perform with Naive-Bayes. We assume, based on our results, that SVM or decision trees would have been more accurate.

Once we have a query from our model that matches the user input, the final step is to substitute tags from sql query and english language templates associated with each class of response and query the database for the information. To do so, we replace whatever keywords appear in the templates, such as "[PREFIX]"

and “[COURSENUM]”, with the values they originally replaced in the user’s query. Based on the query, the resulting value is just substituted in for the string “%x” in the template, or if no rows were found in the database, a separate template is chosen.

## IV. Tools, Packages & Libraries

We primarily used NLTK functionality, for everything from tokenizing to our Naive-Bayes model. We found that NLTK was much easier to use than scikit-learn. However, when we briefly attempted to implement SVM after realizing that our Naive-Bayes model had complications, we did use scikit-learn. Additionally, we also did use BeautifulSoup and Pandas for the data sustainer.

## V. Testing, Performance & Results

For our results, as mentioned above, we did not have the most accurate model due to the nature of our data. Also some formats of building/room and course/section combos are unreliable, and times are currently untagged, with no extra features.

```
CALPASS QUERY (enter (q) to quit): Where is Khosmood's office?
CalPass: foaad's office location is 014 0204.

CALPASS QUERY (enter (q) to quit): When is Khosmood's office hours?
CalPass: foaad has office hours at F 16:10:00 17:00:00, M 16:10:00 17:00:00, T
16:10:00 17:00:00, W 16:10:00 17:00:00.

CALPASS QUERY (enter (q) to quit): What is Khosmood's title?
CalPass: foaad's title is Instr Fac AY.

CALPASS QUERY (enter (q) to quit): Is Khosmood available on Monday?
CalPass: foaad has office hours between 16:10:00 17:00:00 on M.
```

## VI. Appendix

Websites Used for Data Sustainer

| Website | URL |
|---|---|
| CSC Faculty Office Hours | http://frank.ored.calpoly.edu/CSSESpring2022.htm |
| STAT Faculty Office Hours | https://statistics.calpoly.edu/content/directory |
| Cal Poly Schedules:<br>CSC Faculty Information | https://schedules.calpoly.edu/depts_52-CENG_curr.htm |
| Cal Poly Schedules:<br>STAT Faculty Information | https://schedules.calpoly.edu/depts_76-CSM_curr.htm |
| Cal Poly Schedules:<br>CSC Course Information* | https://schedules.calpoly.edu/subject_CSC_curr.htm |
| Cal Poly Schedules:<br>STAT Course Information* | https://schedules.calpoly.edu/subject_STAT_curr.htm |

* Scraped data from Winter 2018 to Fall 2022