1. Computer Science
2. UCI
3. Cristina Lopes
4. David Eppstein
5. Engineering
6. What events are happening this week?
7. What majors are there at UCI?
8. What is a linked list?
9. What classes are there at UCI?
10. When is the science fair?
11. What classes does David Eppstein teach?
12. What is a stop word?
13. Aldrich Park
14. Anteater recreation center
15. What is the UCI mascot?
16. Parking
17. Thisworddoesnotexist
18. Thisisaquerythathasnoresults
19. What kind of sports teams does UCI have?
20. At least half of those queries should be chosen because they do poorly on one or both criteria; the other half should do well. Then change your code to make it work better for the queries that perform poorly, while preserving the good performance of the other ones, and while being as general as possible.

- One problem we had with our code was that queries longer than about 2 words would take over 300ms while shorter ones didn't. In order to solve this problem we created a function that helped us parse the postings lists instead of relying on the built in python eval function. This made it so that even the longer queries remained under 300ms.
- Another change we made was that since the function scales with the number of tokens in the query, we limited the number of words to include if the token length was over 32.
- Another problem we fixed was when the query resulted in no matches. Before, the program would just crash, but we implemented a check for the results and returned no result if there were no postings found.