Proposal

1. Team members and captain

Name	NetID
Captain: Joey Gaysek	jgaysek2
Naina Balepur	nainab2
Brian Betancourt	brianib2

2. What topic have you chosen? Why is it a problem? How does it relate to the theme and to the class?

We have chosen to create a browser extension to fuzzy match Coursera subtitles. This extension would be helpful because search is presently limited to exact match, which may limit some relevant results for users. This relates to Intelligent Browsing because we are improving an existing browser (Chrome, Firefox, etc) as well as Coursera. It relates to the theme of the class because we are using our knowledge from the Text Retrieval section to retrieve relevant documents (subtitles and lecture segments) based on queries.

3. Briefly describe any datasets, algorithms or techniques you plan to use

We plan to use the Coursera API to collect the subtitle chunks and video timestamps. If we encounter limitations with the Coursera API, we will continue our work on data pulled from coursera-dl instead. We will then construct a corpus with these documents. To implement fuzzy matching, or approximate string matching, we will explore several methods such as: hamming distance, levenshtein distance, ngram string match, and bk tree. For testing purposes, we will use the CS 410 course as our data source for videos and subtitles.

4. How will you demonstrate that your approach will work as expected?

We will compare our match results to current algorithms. If we are able to increase the matched results while still keeping them relevant, our approach has worked as expected. For instance, searching “dirichlet” in cs 410 returns 16 matches, however a mistyped query of “direchlet” returns zero matches. This is a contrived example, but fuzzy matching can be beneficial beyond simple user errors, as subtitles can have inaccuracies and inconsistencies, making it more likely for a student to miss pertinent information with the current search implementation.

5. Which programming language do you plan to use?

Work will be done primarily in Javascript, but we may work with Python as well serverside. That said, should a different language become a better choice, we will pivot.

6. Please justify that the workload of your topic is at least 20*N hours, N being the total number of students in your team. You may list the main tasks to be completed, and the estimated time cost for each task.

   20*N => 20*3 => **60 hours**

   - Team organization / proposal refinement / task assignment (5 hours)
   - Research limitations of api (4 hours)
   - Research browser extension creation for Chrome/Firefox (6 hours)
   - Research current state of search (4 hours)
   - Model Selection / Prototype
       - Assembling timestamp chunks of subtitles into corpus (6 hours)
       - Displaying results effectively (8 hours)
       Some or all of the following:
           - implement/test hamming distance (4 hours)
           - implement/test levenshtein distance (6 hours)
           - implement/test ngram string search (6 hours)
           - implement/test bk tree (8 hours)
   - Creating browser extension, integrating queries, search logic and display (20 hours)
   - Testing (10 hours)
   - Evaluation/comparison to alternatives (6 hours)

   => **~70-90 hours**