Team KAF4 Members: Captain Kristen Fisher [kaf4]
Topic: Intelligent Learning Platform: ConceptView
Goal: Extract concepts taught from lectures in to build a “concept summary” for each lecture.

Final Report

`Courseradl` was used to batch download all lecture pdf and transcript text files from Coursera. (https://github.com/coursera-dl/coursera-dl). After that, the files were organized and renamed to match the lectures listed in `Concepts.xlsx` excel spreadsheet provided by the CS 410 staff. Some lectures span multiple slide decks, so it took a bit of time to get everything lined up. Of note, Lecture 33 is unaccounted for. Initially I examined just the n-grams from the transcripts. Then I used `pdftotext` to batch scrape text from each of the pdf files. This was more an exploration, as I soon realized it would be difficult to re-generate a coherent summary of concepts from n-grams or bag of words representations. Additionally, quick scraping pdf text resulted in very messy text files.

At this point, I considered I may need a more practical approach. As a student how would I generate an outline or summary in preparation for an exam? Typically, I start with the title of the lecture, and then use the headings on each slide to create such an outline. I began researching ways to mine text from PDF documents. I found `PyMuPDF` which parses a pdf in such a way that you can view not only the text itself but the font size and style for the text. Most slides contain a heading or title, which typically uses a larger and often bold font. The text retrieved from the larger fonts will be more likely to contain a key concept from the lecture. Using this idea, I wrote the code contained in `CV_demo.py` to mine the lectures for concepts.

In `CV_demo.py`, you’ll find the functions: `get_fonts`, `find_headers`, and `parse_doc`

- `get_fonts` retrieves all of the font styles in the pdf and counts how often they are used.
- `find_headers` uses a threshold at which it is suspected a font is a header containing important text. Figuring out a threshold for what would be considered paragraph text versus heading text was quite tricky. A common idea is that paragraph text is the most frequently used text in a document, so that should be the threshold. However, using a font size too small resulted in irrelevant concepts and too large excluded important concepts. I chose to keep the parameter (`p_size`) for thresholding header text larger and risk generating less concepts. A more complicated implementation would find a way to better tune this parameter.
- `parse_doc` scans the pdf again and only retrieves the text written in the fonts of interest.

Once a list of potential concepts was generated, `get_concepts` removes any stop_words or links from the concepts to generate shorter concepts for analysis. I started with a list of stopwords (lemur-stopwords) from one of our previous assignments. I added to this quite a few words along the way including items frequently used in lectures such as variables (*q, n,* etc.). Capitalization was used as an indicator that text was likely a heading/concept. During this process, the original (long) concepts before the removal of stopwords were saved.

`get grams` generates unigrams, bigrams, and trigrams for any given text string. I used these n-grams for ranking in `get_concept_score`. Ranking was difficult since I removed stop words and did not scrape all text from the pdf files. The sum of the counts of the n-grams were used for normalization. Higher weights were rewarded to bigrams and trigrams matches. Important concepts were sometimes lost due to the use of abbreviation only in the pdf files, such as LM for Language Model. To correct this, the counts of the unigrams (`pdf_uni`) from the concepts were including in ranking to avoid zero counts. It was kind of an arbitrary ranking system based on my observations with this dataset, it could use improvement.

Once the concepts were ranked, any concept with a rank greater than zero was selected in `choose_output` to be used in the summary. Very short concepts sometimes benefit from the inclusion of stopwords. For example, the short concept 'VSM' versus the original concept with stopwords: 'What VSM Doesn't Say.' For very short concepts, the original (long) version of the concept prior to the removal of stopwords was retrieved and used in the summary in to provide more clarity. Otherwise, the short version stripped of stopwords was sued unless the long concept was extraordinarily long.

The code contained in `CV_demo.py` is the same used to generate the concept summaries found in the `Concepts.xlsx` spreadsheet. The demo files provided are only from Lectures 1 through 5 and the output displays the "Selected Concepts" along with the `concepts` matrix which contains both the long and short concepts as well as the rank. All 42 lecture files from Part 1 of the CS 410 course were used to generate the summaries and only the "Selected Concepts" from the output was written to "CV Generated Concepts" column of `Concepts.xlsx`. Of note, I did not have a pdf file for Lecture 33.

`Concepts.xlsx` is a spreadsheet of the CS 410 Part 1 lectures containing my CV Generated Concepts alongside human generated concepts provide by the CS 410 staff for comparison. Slide decks were used across multiple lectures resulted in similar concept summaries being generated for multiple lectures, despite the fact certain concepts from the slides were covered in different lecture videos. An example of this is Lectures 33 to 35. The concepts are quite similar, however the order in which the concepts are output changes slightly due to ranking. I noticed my results tend to miss some of the more specific concepts. For example, in Lecture 11, my concepts contain "Integer Compression Methods" but not the specific method of "Unary Coding."

However, based on the amount of effort I was able to contribute as the sole team member working on this project, I am not completely disappointed with my concept summary results. My ranking function is a bit arbitrary. Further testing of the coverage of the generated concepts would be a better way to score concepts. Due to this, I favored stricter restrictions when considering potential concepts to limit the amount of junk in my summaries. Improvements could include loosening the restrictions for potential concepts to include additional text and find more relevant concepts with a better ranking function.