Final Project : Documentation

MOOC Lecture Video and Content Integration with Associated Textbook

1. Team Information

NetID : jaecho2@illinois.edu Team Member Count : 1
Captain : jaecho2@illinois.edu

---

2. Theme of the Project

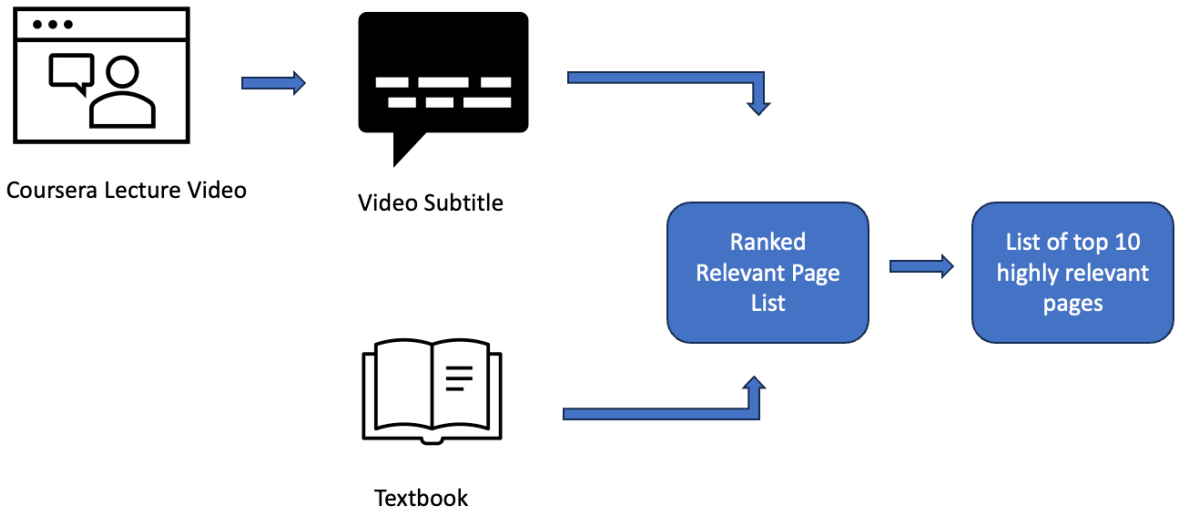
As students we all dream of a world where we can read the associated textbook cover to cover (or perhaps just me) so that each point mentioned in the lectures can be corelated with a specific section within the text. This has the benefit of reinforcing the learning experience.

The problem is that we rarely have the luxury of time to read a textbook cover to cover, which may lead us to overlook valuable information. This project is an attempt to assist students with a tool that will build the association between a textbook and MOOC lecture, so that the full power of a textbook can be utilized to enhance and expand the learning experience.

---

3. Architecture and Design

The project consists of the following input and output.
Input : Video Subtitle File selected by the user as the query list.
Input : Textbook for the course used as the document list.
Output : List of Top Ranked pages considered to be relevant to the current lecture video that the student is learning from.



4. Use Case

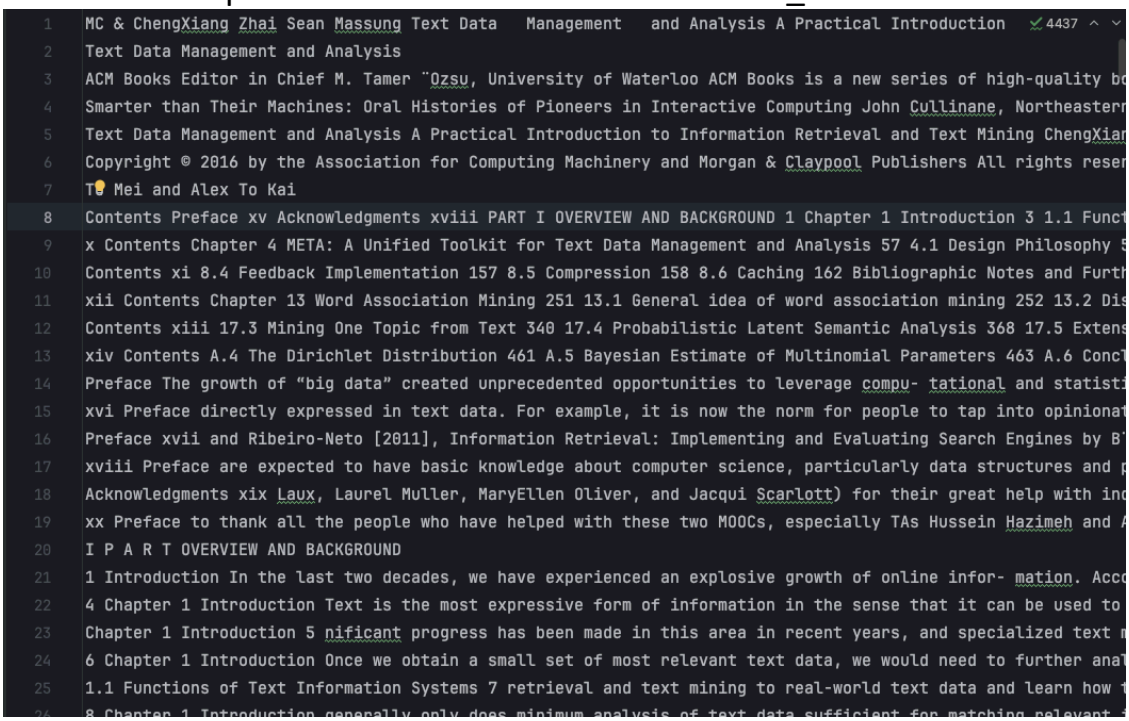
[STUDENT] Student watches a Lecture video that is provided in Coursera.
After completing the Video student want to also review relevant section in the associated textbook to gain additional understanding and clarity.
[STUDENT] Scans the textbook to determine where the relevant section is that covers all the terms and concepts that the professor mentioned in class.
[STUDENT] Student can scan and pickup a couple of relevant sections but is afraid that he/she might have missed something that is relevant.
[STUDENT]Student submits the textbook and subtitles for the lecture that was just completed.
[SYSTEM]System takes this information and creates a ranked list of most probable pages associated with the video.

5. Code Documentation

```
def extract_and_save_text(pdf_path, output_dir):
```

This function takes a path to a textbook as an input and stores the information in a file called ‘text_data.dat’. Each row in ‘text_data.dat’ pertains to a single page in the textbook.

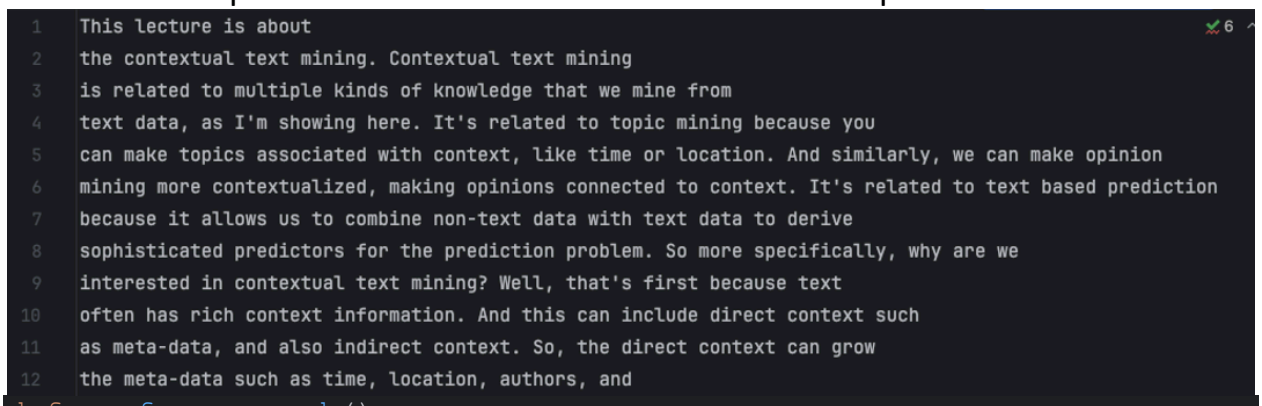
Below is a sample screenshot of the content of ‘text_data.dat’.




```
def load_subtitle():
```

This function takes a path to a subtitle file as an input and stores it as a list of queries in a file called `subtitle-queries.txt`. The subtitle file must be in a txt file format.

Below is a sample screenshot of the content of ‘`subtitle-queries.txt`.




```
def perform_search():
```

This function does the heavy lifting by indexing the textbook file ‘text_data.dat’ and ranking them based on the queries stored in ‘subtitle-queries.txt’

From the list of rankers available in MeTA, Okapi BM25 was used.

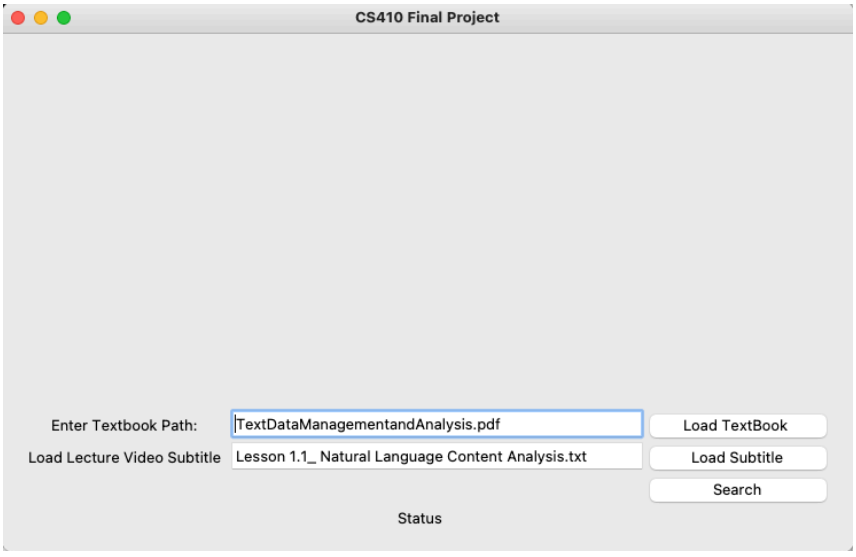
The logic to decide relevant document is as below.

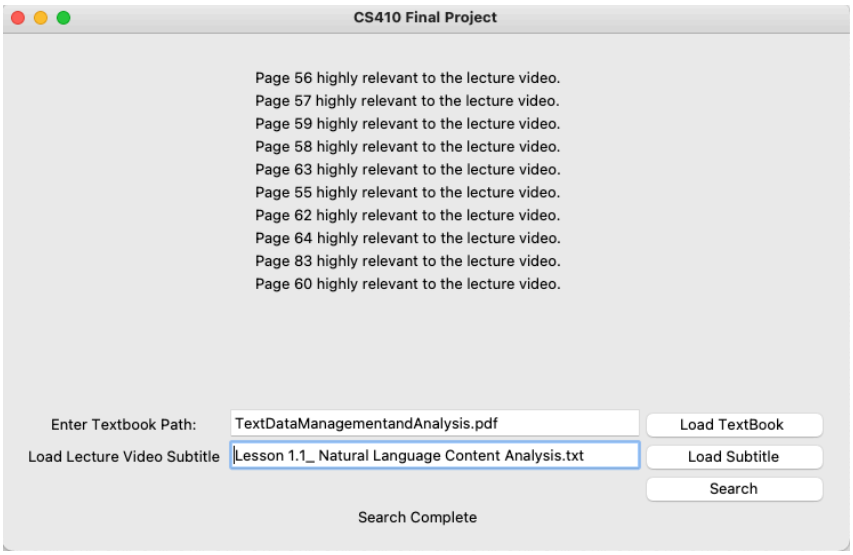
- Go through each query topic in subtitle-queries.txt.
- Pic the top 5 ranked documents from this list
- This will be stored in a list called *ranked_doc_ids*
- Pick the 15 most common documents in *ranked_doc_ids*
- Present the top 15 documents as the relevant documents that students should review for the video lecture.

A sample output is below

```
Number of documents : 514
Number of unique terms 4689:
Average document length 180.577819824:
Total corpus terms : 92817
Running queries
[((57, 1), 25), ((56, 1), 24), ((56, 2), 20), ((56, 3), 17), ((57, 3), 15), ((59, 1), 13), ((63, 2), 12), ((55, 1), 11), ((57, 2), 10)
```

Also added a nice GUI so that user can easily change the input files.





6. Challenges

4.1 Package unable to parse PDF properly.

The biggest initial challenge was in using the correct library that can parse the textbook which is in pdf format.

First attempt was to use *PyPDF2,* but faced problems because each word was not parsed properly with space in between. Some of the words were but majority of the words were all appended together without any spaces.

Second attempt was to try a different library called *pdfPlumber*, this provided marginally better results. Even after trying to parse the text with regular expression did not work well.

Third attempt was to experiment with a library called FITZ. Based on a number of testing this worked well.

4.2 Combining all documents into a single .dat file.

Initially I tried to create a separate .dat file for each page in the textbook. This would mean I had a total of 514 pages. So each page would be a document within the data source. I had a lot of trouble working with the ranking function and trying to combine the results based on each individual document. To overcome this I combined all of the documents into a single text_data.dat file and had each row represent a page in the textbook.

4.3 Using legacy Python version 2.7 for compatibility issues with MeTA

Using anaconda, I created 2 environment one based on Python 3.11 and another based on 2.7. I wanted to create nice bar charts to present the user with the ranking results. I found matplotlib does not work well in Python 2.7 and MeTA does not work at all in Python 3.11. The only option I had was to stick with Python 2.7 and use text based results.

7. Validation

For validation of the rank results and relevancy, human intervention was required. Through the 3 test scenarios that were run I rated the system on Low, Medium, and High.

- Low : 20% accuracy of the listed pages on relevancy to the lecture video.
- Medium : 20% ~ 80% accuracy of the listed pages on relevancy to the lecture video.
- High : over 80% accuracy of the listed pages on relevancy to the lecture video.

Test Case 1

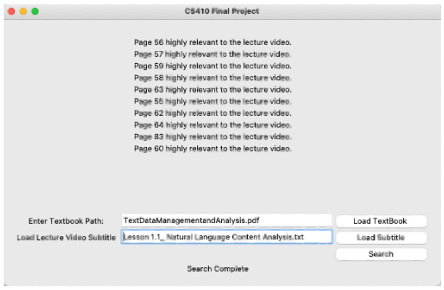
Input :

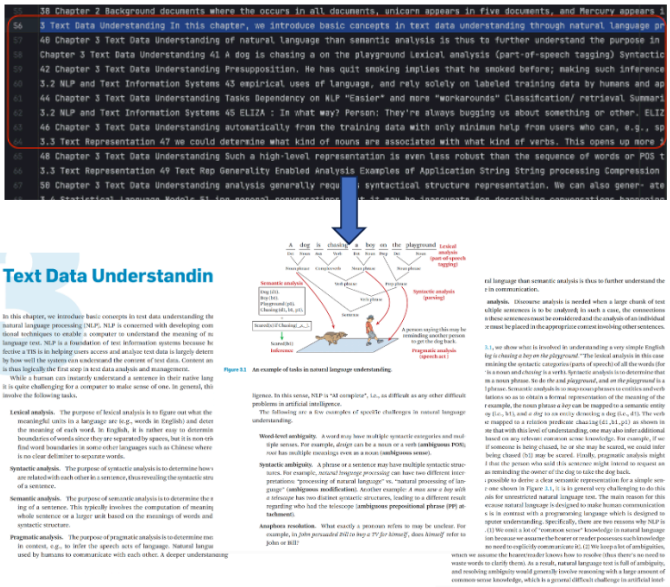
Document : Text Data Management and Analysis.pdf (The Textbook)

Query : Lesson 1.1_ Natural Language Content Analysis.txt (The subtitles for the video)

Expected Output : The first lecture video and subtitles were chosen as the input. Naturally since this is the first lecture, student would expect that the relevant textbook content should be in the beginning chapters.

Output : Result is High : The outcome show that most of the contents on the listed pages are indeed about the content presented in the video.





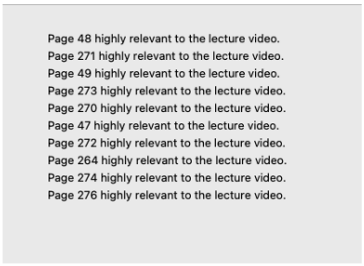
Test Case 2

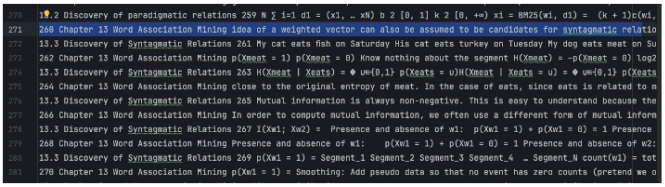
Input :

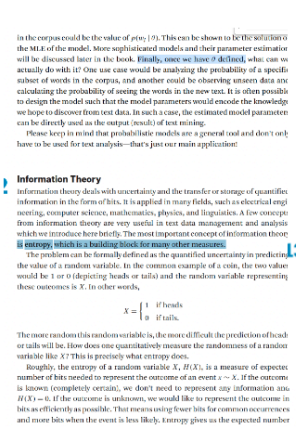
Document : Text Data Management and Analysis.pdf (The Textbook)

Query : Lesson 8.1 Syntagmatic Relation Discovery_ Entropy.txt (The subtitles for the video)

Output : Result is High : The outcome show that most of the contents on the listed pages are indeed about the content presented in the video.







in the corpus could be the value of $p(w_i|\theta)$. This can be shown to be the solution to the MLE of the model. More sophisticated models and their parameter estimation will be discussed later in the book. Finally, once we have θ defined, what can we actually do with it? One use case would be analyzing the probability of a specific subset of words in the corpus, and another could be observing unseen data and calculating the probability of seeing the words in the new text. It is often possible to design the model such that the model parameters would encode the knowledge we hope to discover from text data. In such a case, the estimated model parameters can be directly used as the output (result) of text mining.

Please keep in mind that probabilistic models are a general tool and don't only have to be used for text analysis—that's just our main application!

Information Theory

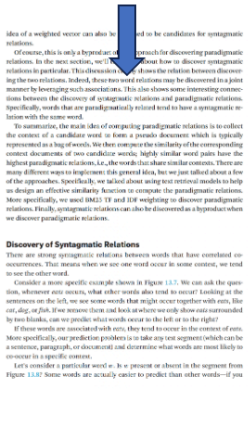
Information theory deals with uncertainty and the transfer or storage of quantified information in the form of bits. It is applied in many fields, such as electrical engineering, computer science, mathematics, physics, and linguistics. A few concepts from information theory are very useful in text data management and analysis which we introduce here briefly. The most important concept of information theory is entropy, which is a building block for many other measures.

The problem can be formally defined as the quantified uncertainty in predicting the value of a random variable. In the common example of a coin, the two values would be 1 or 0 (depicting heads or tails) and the random variable representing these outcomes is X. In other words,

$$X = \begin{cases} 1 & \text{if heads} \\ 0 & \text{if tails.} \end{cases}$$

The more random this random variable is, the more difficult the prediction of heads or tails will be. How does one quantitatively measure the randomness of a random variable like X? This is precisely what entropy does.

Roughly, the entropy of a random variable X, $H(X)$, is a measure of expected number of bits needed to represent the outcome of an event $x \sim X$. If the outcome is known (completely certain), we don't need to represent any information and $H(X) = 0$. If the outcome is unknown, we would like to represent the outcome in bits as efficiently as possible. That means using fewer bits for common occurrences and more bits when the event is less likely. Entropy gives us the expected number



idea of a weighted vector can also be assumed to be candidates for syntagmatic relations.

Of course, this is only a hypothetical approach for discovering paradigmatic relations. In the next section, we'll discuss how to discover syntagmatic relations in particular. This discussion will also show the relation between discovering the two relations. Indeed, these two word relations may be discovered in a joint manner by leveraging such associations. This also shows some interesting connections between the discovery of syntagmatic relations and paradigmatic relations. Specifically, words that are paradigmatically related tend to have a syntagmatic relation with the same word.

To summarize, the main idea of computing paradigmatic relations is to collect the context of a candidate word to form a pseudo document which is typically represented as a bag of words. We then compare the similarity of the corresponding context documents of two candidate words; highly similar word pairs have the highest paradigmatic relations, i.e., the words that share similar contexts. There are many different ways to implement this general idea, but we just talked about a few of the approaches. Specifically, we talked about using text retrieval models to help us design an effective similarity function to compute the paradigmatic relations. More specifically, we used BM25 TF and IDF weighting to discover paradigmatic relations. Finally, syntagmatic relations can also be discovered as a byproduct when we discover paradigmatic relations.

13.3 Discovery of Syntagmatic Relations

There are strong syntagmatic relations between words that have correlated co-occurrences. That means when we see one word occur in some context, we tend to see the other word.

Consider a more specific example shown in Figure 13.7. We can ask the question, whenever *eats* occurs, what other words also tend to occur? Looking at the sentences on the left, we see some words that might occur together with *eats*, like *cat*, *dog*, or *fish*. If we remove them and look at where we only show *eats* surrounded by two blanks, can we predict what words occur to the left or to the right?

If these words are associated with *eats*, they tend to occur in the context of *eats*. More specifically, our prediction problem is to take any text segment (which can be a sentence, paragraph, or document) and characterize what words are most likely to co-occur in a specific context.

Let's consider a particular word *w*. It *w* present or absent in the segment from Figure 13.8? Some words are actually easier to predict than other words—if you

Test Case 3
Input :
Document : Text Data Management and Analysis.pdf (The Textbook)
Query : Lesson 12.7 Contextual Text Mining_ Mining Causal Topics with Time Series Supervision.txt (The subtitles for the video)

Output : Result is High : The outcome show that most of the contents on the listed pages are indeed about the content presented in the video.

8. Future Improvements

The logic behind the ranking system seems to work very well based on validation results. What remains is mostly GUI improvements to show a nice bar chart on the ranked documents.
Also need to figure out a better mapping between the page index in text_data.dat vs the actual PDF file.
Another future improvement would also involve taking into consideration queries that user would enter. This would be considered as an additional query with a higher weight than what is derived from the subtitle list.

9. Timeline

Overall, this project much more time than the initial estimated 20 hours per student. This was due to facing a lot of hurdles and research required during the initial implementation.

Timeline
W8 - Proposal writeup, initial design
W9 - Research into algorithms and approach enhancements
W10 – Version 0.1 : basic features
W11 – Associated Technology Review
W12 – Version 0.2 : work on UI
W13 – Version 0.3 : start work on information retrieval algorithm, progress report
W14 – Function Test
W15,16 – Additional user test and submit

10. References

PyPDF2 : https://pypi.org/project/PyPDF2/
Pdfplumber : https://pypi.org/project/pdfplumber/0.1.2/
FITZ : https://pymupdf.readthedocs.io/en/latest/
Matplotlib : https://matplotlib.org/stable/
Metapy : https://github.com/meta-toolkit/metapy