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

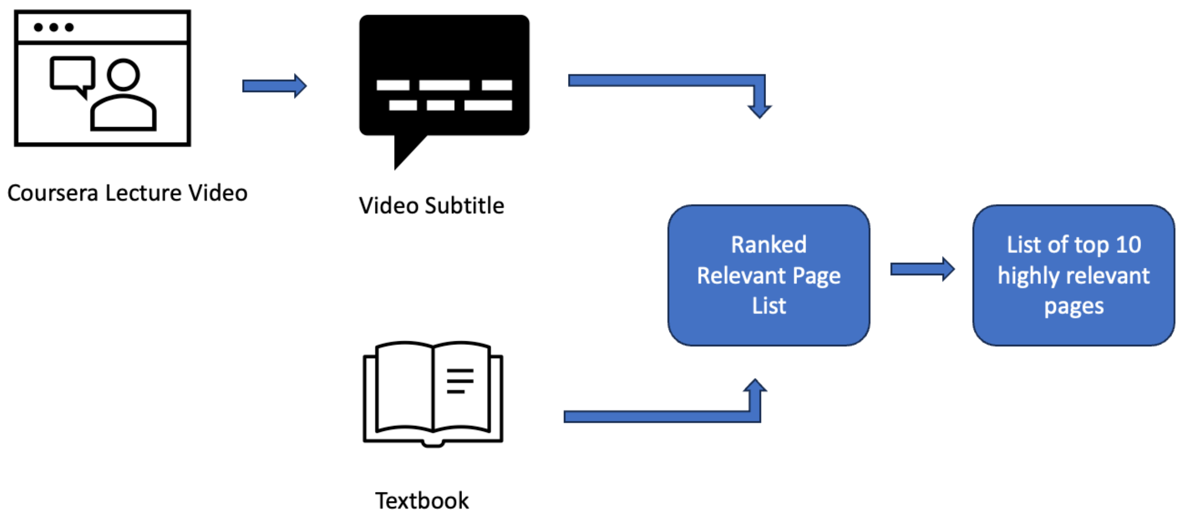
As students we all dream of a world where we can read the associated textbook cover to cover (or perhaps just me) so that each point mentioned in the lectures can be corelated with a specific section within the text. This has the benefit of reinforcing the learning experience.

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’.

```
MC & ChengXiang Zhai Sean Massung Text Data   Management   and Analysis A Practical Introduction
Text Data Management and Analysis
ACM Books Editor in Chief M. Tamer ¨Ozsu, University of Waterloo ACM Books is a new series of high-quality bo
Smarter than Their Machines: Oral Histories of Pioneers in Interactive Computing John Cullinane, Northeastern
Text Data Management and Analysis A Practical Introduction to Information Retrieval and Text Mining ChengXian
Copyright © 2016 by the Association for Computing Machinery and Morgan & Claypool Publishers All rights reser
To Mei and Alex To Kai
Contents Preface xv Acknowledgments xviii PART I OVERVIEW AND BACKGROUND 1 Chapter 1 Introduction 3 1.1 Funct
x Contents Chapter 4 META: A Unified Toolkit for Text Data Management and Analysis 57 4.1 Design Philosophy 5
Contents xi 8.4 Feedback Implementation 157 8.5 Compression 158 8.6 Caching 162 Bibliographic Notes and Furth
xii Contents Chapter 13 Word Association Mining 251 13.1 General idea of word association mining 252 13.2 Dis
Contents xiii 17.3 Mining One Topic from Text 340 17.4 Probabilistic Latent Semantic Analysis 368 17.5 Extens
xiv Contents A.4 The Dirichlet Distribution 461 A.5 Bayesian Estimate of Multinomial Parameters 463 A.6 Concl
Preface The growth of “big data” created unprecedented opportunities to leverage compu- tational and statisti
xvi Preface directly expressed in text data. For example, it is now the norm for people to tap into opinionat
Preface xvii and Ribeiro-Neto [2011], Information Retrieval: Implementing and Evaluating Search Engines by B
xviii Preface are expected to have basic knowledge about computer science, particularly data structures and p
Acknowledgments xix Laux, Laurel Muller, MaryEllen Oliver, and Jacqui Scarlott) for their great help with inc
xx Preface to thank all the people who have helped with these two MOOCs, especially TAs Hussein Hazimeh and A
I P A R T OVERVIEW AND BACKGROUND
1 Introduction In the last two decades, we have experienced an explosive growth of online infor- mation. Acco
4 Chapter 1 Introduction Text is the most expressive form of information in the sense that it can be used to
Chapter 1 Introduction 5 nificant progress has been made in this area in recent years, and specialized text m
6 Chapter 1 Introduction Once we obtain a small set of most relevant text data, we would need to further anal
1.1 Functions of Text Information Systems 7 retrieval and text mining to real-world text data and learn how t
8 Chapter 1 Introduction generally only does minimum analysis of text data sufficient for matching relevant
```

```
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`.

```
This lecture is about
the contextual text mining. Contextual text mining
is related to multiple kinds of knowledge that we mine from
text data, as I'm showing here. It's related to topic mining because you
can make topics associated with context, like time or location. And similarly, we can make opinion
mining more contextualized, making opinions connected to context. It's related to text based prediction
because it allows us to combine non-text data with text data to derive
sophisticated predictors for the prediction problem. So more specifically, why are we
interested in contextual text mining? Well, that's first because text
often has rich context information. And this can include direct context such
as meta-data, and also indirect context. So, the direct context can grow
the meta-data such as time, location, authors, and
```

```
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.

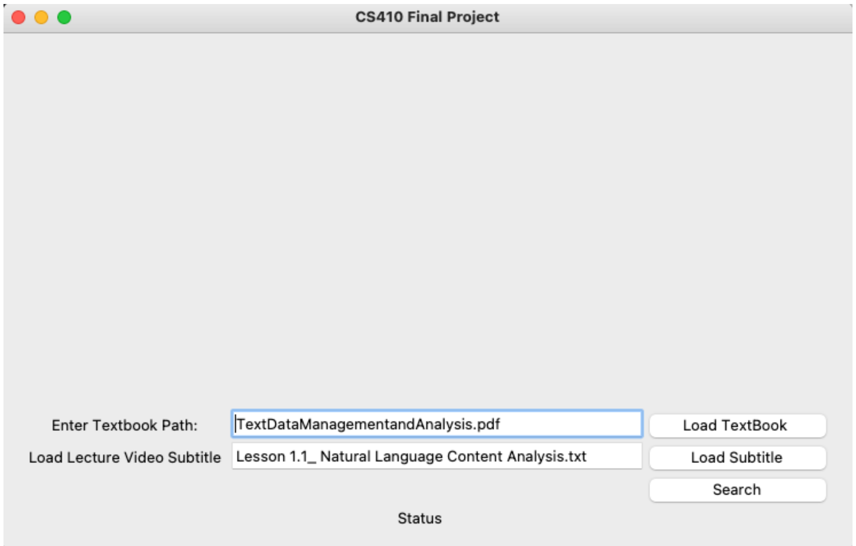
The logic to decide relevant document is as below.

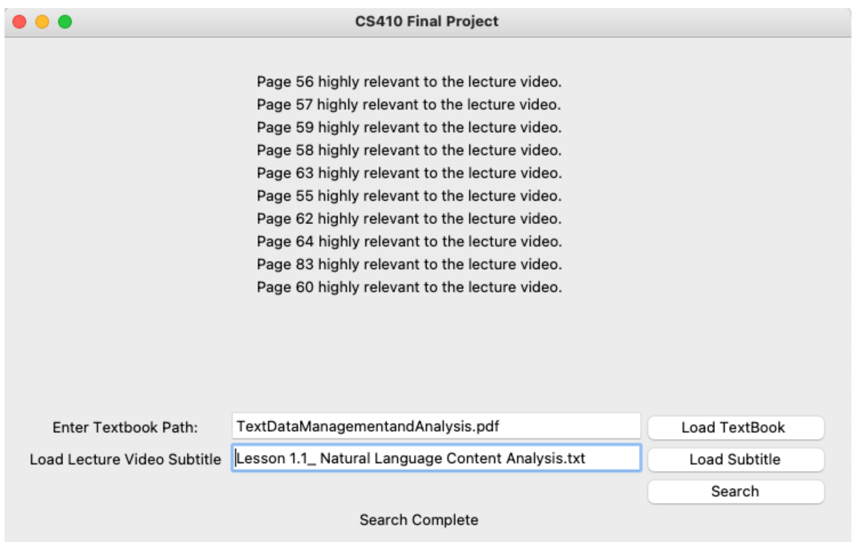
- Go through each query topic in subtitle-queries.txt.
- Pic the top 5 ranked documents from this list
- This will be stored in a list called *ranked_doc_ids*
- Pick the 10 most common documents in *ranked_doc_ids*
- Present the top 10 documents as the relevant documents that students should review for 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

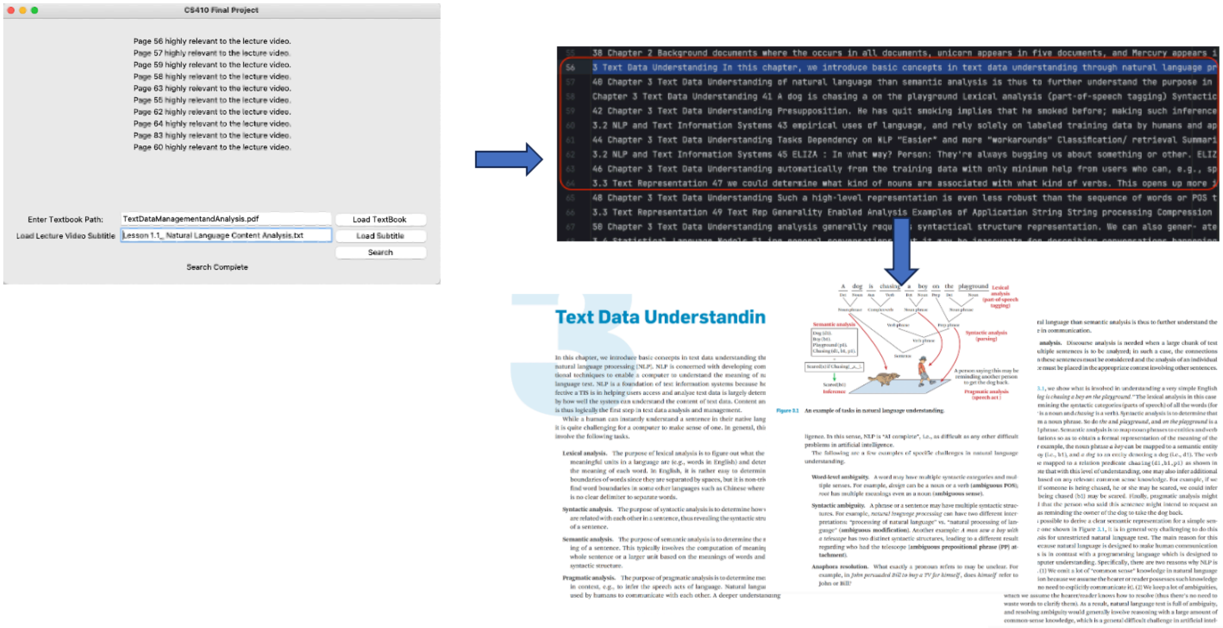
Input :

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.



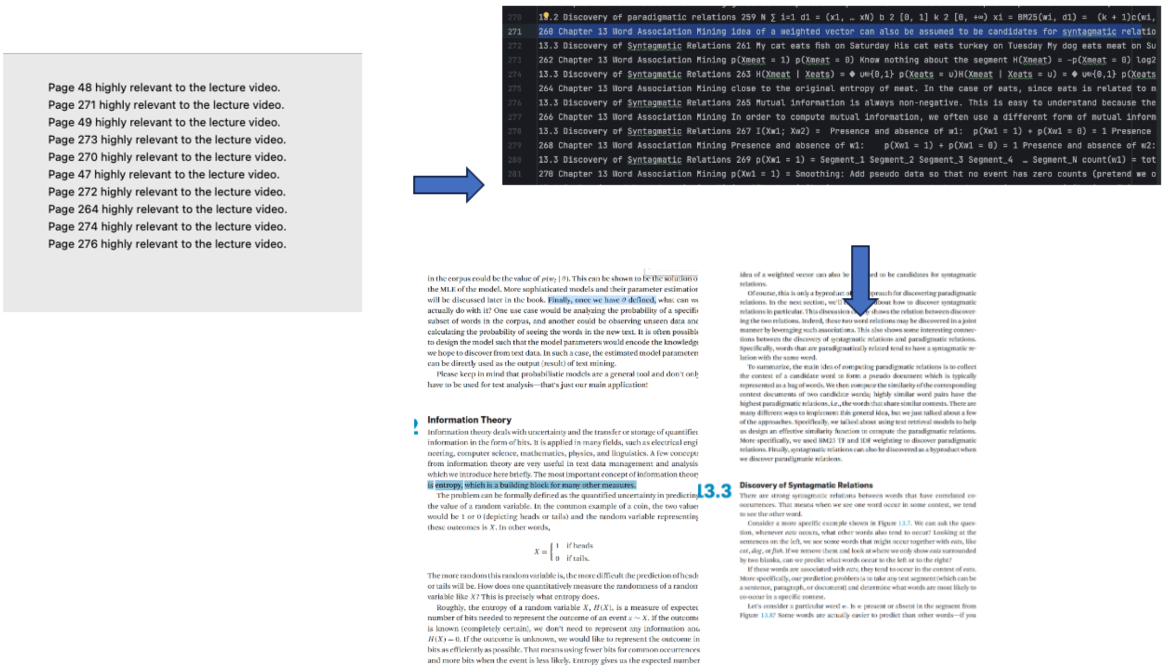
Test Case 2

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.



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